वेब डेवलपर

WEB DEVELOPER

(कार्य भूमिका)

(योग्यता पैक– Ref. Id SSC/Q0503)

क्षेत्र — सूचना प्रौद्योगिकी–सूचना प्रौद्योगिकी सक्षम सेवाएँ (IT–ITeS)

कक्षा 11 के लिए मॉड्यूल

पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान

(भारत सरकार के शिक्षा मंत्रालय के अधीन रा.शै.अ.प्र.प. की घटक मॉड्यूल)

श्यामला हिल्स, भोपाल— 462002, मध्य प्रदेश, भारत

http//www.psscive.ac.in

आमुख

व्यावसायिक शिक्षा एक गतिशील और विकासशील क्षेत्र है और यह सुनिश्चित करना अत्यंत महत्वपूर्ण है कि प्रत्येक विद्यार्थी के पास गुणवत्तापूर्ण शिक्षण सामग्री उपलब्ध हो। पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान (पी.एस.एस.सी.आई.वी.ई.) की व्यापक और समावेशी अध्ययन सामग्री तैयार करने की यात्रा कठिन और समय लेने वाली है जिसके लिए गहन शोध, विशेषज्ञ परामर्श और राष्ट्रीय शैक्षिक अनुसंधान और प्रशिक्षण परिषद् (रा.शै.अ.प्र.प.) द्वारा प्रकाशन की आवश्यकता है। हालाँकि, अंतिम अध्ययन सामग्री की अनुपस्थिति हमारे विद्यार्थियों की शैक्षिक प्रगति में बाधा नहीं बननी चाहिए। इस आवश्यकता को देखते हुए हम प्रारूप अध्ययन सामग्री प्रस्तुत करते हैं, जो एक अनंतिम लेकिन व्यापक मार्गदर्शिका है, जिसे शिक्षण और सीखने के बीच का अंतर दूर करने के लिए डिजाइन किया गया है, जब तक कि अध्ययन सामग्री का आधिकारिक संस्करण रा.शै.अ.प्र.प. द्वारा उपलब्ध नहीं करा दिया जाता। प्रारूप अध्ययन सामग्री शिक्षकों और विद्यार्थियों के लिए अंतरिम अवधि में उपयोग करने के लिए सामग्री का एक संरचित और सुलभ सेट प्रदान करती है। सामग्री को निर्धारित पाठ्यक्रम के साथ संरेखित किया गया है ताकि यह सुनिश्चित किया जा सके कि विद्यार्थी अपने सीखने के उद्देश्यों के साथ सही रास्ते पर बने रहें।

मॉड्यूल की विषयवस्तु शिक्षा में निरंतरता बनाए रखने और व्यावसायिक शिक्षा में शिक्षण-अधिगम की गति को बनाए रखने के लिए तैयार की गई है। इसमें पाठ्यक्रम और शैक्षिक मानकों के अनुरूप आवश्यक अवधारणाएँ और कौशल शामिल हैं। हम उन शिक्षाविदों, व्यावसायिक शिक्षकों, विषय विशेषज्ञों, उद्योग विशेषज्ञों, शैक्षणिक सलाहकारों और अन्य सभी लोगों के प्रति आभार व्यक्त करते हैं जिन्होंने इस प्रारूप अध्ययन सामग्री के निर्माण में अपनी विशेषज्ञता और अंतर्दृष्टि प्रदान की।

शिक्षकों को अध्ययन सामग्री के प्रारूप मॉड्यूल को एक मार्गदर्शक के रूप में उपयोग करने और अपने शिक्षण को अतिरिक्त संसाधनों और गतिविधियों से पूरक बनाने के लिए प्रोत्साहन दिया जाता है जो उनके विद्यार्थियों की विशिष्ट शिक्षण शैलियों और आवश्यकताओं को पूरा करते हैं। सहयोग और प्रतिक्रिया महत्वपूर्ण हैं; इसलिए, हम अध्ययन सामग्री की विषय-वस्तु में सुधार के लिए विशेष रूप से शिक्षकों द्वारा, सुझावों का स्वागत करते हैं।

भोपाल
अगस्त 2025

दीपक पालीवाल
संयुक्त निदेशक
पं.सुं.श. केंद्रीय व्यावसायिक शिक्षा संस्थान (पी.एस.एस.सी.आई.वी.ई.)
राष्ट्रीय शैक्षिक अनुसंधान और प्रशिक्षण परिषद्

पाठ्यपुस्तक विकास समिति

सदस्य

1. प्रो. के. वी. आर्य, एबीवी-भारतीय सूचना प्रौद्योगिकी एवं प्रबंधन संस्थान, ग्वालियर (म.प्र.)
2. प्रो. विशाल गोयल, निदेशक, आईक्यूएसी, जीएलए विश्वविद्यालय, मथुरा
3. श्री विजय गोस्वामी, संस्थापक एवं निदेशक, एट्रिक्स टेक्नोलॉजीज़, आगरा (उ.प्र.)
4. श्री अंकित श्रीवास्तव, एनआईटी अगरतला

सदस्य समन्वयक

डॉ. मुनेश चंद्र, प्रोफेसर (कंप्यूटर विज्ञान एवं अभियांत्रिकी), इंजीनियरिंग और प्रौद्योगिकी विभाग, पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान, रा.शै.अ.प्र.प., श्यामला हिल्स, भोपाल, मध्यप्रदेश।

अनुवाद, संपादन एवं समीक्षा

- डॉ. कृष्ण कुमार दीवान, रजिस्ट्रार, नितरा, गाजियाबाद उत्तर प्रदेश
- मंयक के. भूषण, व्यख्याता, कम्प्यूटर साइंस, जी.वी.एस.एस., गाजियाबाद, उत्तर प्रदेश
- विजेंद्र बोरबन, वरिष्ठ संपादक, पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान, रा.शै.अ.प्र.प., श्यामला हिल्स, भोपाल, मध्यप्रदेश।
- अवंतिका त्रिपाठी (मुख्य कार्यकारी), किरपी (कार्यकारी सहयोगी), कविता (कार्यकारी सहयोगी), अनन्या एडु-टेक कंसल्टेंसी सर्विसेज़, नई दिल्ली
- राजीव पाल, डी.टी.पी. ऑपरेटर, पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान, रा.शै.अ.प्र.प., श्यामला हिल्स, भोपाल, मध्यप्रदेश।

कार्यक्रम समन्वयक

रजनीश, सहायक पुस्तकालयाध्यक्ष, पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान, रा.शै.अ.प्र.प., श्यामला हिल्स, भोपाल, मध्यप्रदेश।

राज्य समन्वयक

विपिन कुमार जैन, सह प्राध्यापक एवं विभागाध्यक्ष, मानविकी, विज्ञान, शिक्षा और अनुसंधान, पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान, रा.शै.अ.प्र.प., श्यामला हिल्स, भोपाल, मध्यप्रदेश।

कार्यक्रम निदेशक

दीपक पालीवाल, संयुक्त निदेशक, पंडित सुंदरलाल शर्मा केंद्रीय व्यावसायिक शिक्षा संस्थान, रा.शै.अ.प्र.प., श्यामला हिल्स, भोपाल, मध्यप्रदेश।

विषयसूची

मॉड्यूल 1

वेब विकास की मूल बातें

भूमिका

इस मॉड्यूल में आप सबसे पहले "सूचना प्रौद्योगिकी (Information Technology - IT) तथा सूचना प्रौद्योगिकी सक्षम सेवाएँ (Information Technology Enabled Services - ITeS)" के बारे में सीखेंगे। IT तथा ITeS उद्योग वैश्विक अर्थव्यवस्था में एक केंद्रीय भूमिका निभाते हैं और विभिन्न क्षेत्रों में नवाचार, उत्पादकता एवं संपर्क क्षमता को सशक्त बनाते हैं। इसके पश्चात हम "वेब डिज़ाइन और विकास" की ओर बढ़ेंगे, जहाँ आप वेब डिज़ाइन और विकास, वेब डिज़ाइन की सीमाएँ, वेबसाइट एवं इंटरफ़ेस निर्माण हेतु मुख्य पूर्वानुमानों तथा वेब डिज़ाइनिंग प्रक्रिया के इंटरफ़ेस के बारे में जानेंगे। इसके बाद हम "वेब डिज़ाइन एवं विकास उपकरण" पर चर्चा करेंगे, जिसमें वेब विकास मानक, वेब विकास उपकरण, वेबसाइटों के प्रकार तथा वेबसाइट के कोडिंग और प्रोग्रामिंग को शामिल किया गया है। फिर हम "आईटी के एप्लीकेशन विकास मॉडल" पर चर्चा करेंगे, जिसमें आप एप्लीकेशन विकास की प्रक्रिया, एप्लीकेशन विकास में SDLC मॉडल्स (सॉफ्टवेयर विकास जीवन चक्र) के महत्व, वॉटरफॉल मॉडल, पुनरावृत्ति मॉडल तथा एजाइल मॉडल के बारे में समझेंगे।

इसके पश्चात "वेब डिज़ाइनिंग विनिर्देश" को समझाया जाएगा, जिसमें वेब डिज़ाइन विनिर्देश, व्यापारिक आवश्यकताओं के विनिर्देश (Business Requirement Specifications), व्यापारिक आवश्यकता दस्तावेज़ के घटक, उसके लाभ, इसे बनाने की प्रक्रिया तथा सॉफ़्टवेयर आवश्यकताओं के विनिर्देशों को कवर किया गया है। इसके बाद हम "प्रोग्रामिंग के लिए निम्न-स्तरीय और उच्च-स्तरीय डिज़ाइन" को कवर करेंगे, जिसमें निम्न-स्तरीय डिज़ाइन, उच्च-स्तरीय डिज़ाइन, वस्तु-उन्मुख डिज़ाइन (Object-Oriented Design), डेटाबेस डिज़ाइन, एपीआई (Application Programming Interface) डिज़ाइन तथा उच्च-स्तरीय डिज़ाइन का उद्देश्य सम्मिलित है। इसके पश्चात "डिज़ाइन दोषों का परीक्षण एवं पहचान" नामक खंड आएगा, जिसमें डिज़ाइन दोष, दोषों के प्रकार, डिज़ाइन दोषों के सामान्य कारण एवं सॉफ़्टवेयर विकास जीवन चक्र में दोषों के विभिन्न चरणों पर चर्चा की जाएगी। अंत में हम "डिज़ाइन दोषों का समाधान" शीर्षक खंड में डिज़ाइन दोषों का समाधान, भविष्य की डिज़ाइन के लिए सुधार की संभावनाएँ, कोडिंग उपकरण, तथा डिज़ाइन दोषों का अभिलेखन एवं प्रलेखन कवर करेंगे। इस मॉड्यूल में आपको IT/ITeS क्षेत्र में आवश्यक ज्ञान और कौशल से सुसज्जित किया जाएगा।

सत्र 1

सूचना प्रौद्योगिकी (IT) और सूचना प्रौद्योगिकी सक्षम सेवाएँ (ITeS)

माया, टेक्नो विल की एक उत्साही सॉफ्टवेयर डेवलपर है और IT/ITeS उद्योग के केंद्र में कार्य करते हुए सफलता की ऊँचाइयों को छू रही थी। उसने एक अत्याधुनिक परियोजना पर नियुक्त होने के बाद IT की समन्वयात्मक प्रकृति को समझा, जिसमें वह हार्डवेयर, नेटवर्किंग और साइबर सुरक्षा विशेषज्ञों के साथ कार्य कर रही थी। एक आईटी कंसल्टिंग संगोष्ठी से प्रेरित होकर उसने बिजनेस प्रोसेस आउटसोर्सिंग (बीपीओ), डिजिटल मार्केटिंग और स्वास्थ्य देखभाल आईटी (Healthcare IT) के क्षेत्रों में गहराई से समझा और उनके सामाजिक प्रभाव के बारे में जाना। माया की यह यात्रा भारत के "डिजिटल इंडिया" अभियान तक पहुँचती है, जहाँ उसने भविष्य के लिए एक डिजिटल विरासत का निर्माण किया — जैसा कि चित्र 1.1 में दर्शाया गया है।

चित्र 1.1— सॉफ्टवेयर डेवलपर के रूप में कार्य करती माया

इस अध्याय में, सूचना प्रौद्योगिकी (IT) और सूचना प्रौद्योगिकी सक्षम सेवाएँ (ITeS) का संक्षिप्त परिचय, उनके खंड तथा प्रासंगिकता पर चर्चा की जाएगी। सूचना प्रौद्योगिकी (IT) और सूचना प्रौद्योगिकी सक्षम सेवाएँ (ITeS) उद्योग वैश्विक अर्थव्यवस्था में एक केंद्रीय भूमिका निभाते हैं, जो नवाचार, उत्पादकता एवं विभिन्न क्षेत्रों में संपर्कक्षमता को बढ़ावा देते हैं। यह उद्योग प्रौद्योगिकी, डेटा प्रबंधन एवं डिजिटल सेवाओं से संबंधित विविध गतिविधियों को सम्मिलित करता है।

1.1 सूचना प्रौद्योगिकी (IT)

सूचना प्रौद्योगिकी का तात्पर्य कंप्यूटर, सॉफ़्टवेयर, नेटवर्क एवं अन्य तकनीकी उपकरणों के उपयोग से है, जो डेटा को संसाधित, स्टोर, प्रेषित एवं नियंत्रित करने के कार्यों में प्रयुक्त होते हैं। इसमें हार्डवेयर और सॉफ़्टवेयर विकास, आईटी समर्थन, तथा सिस्टम एडमिनिस्ट्रेशन (System Administration) सम्मिलित हैं।

सूचना प्रौद्योगिकी सक्षम सेवाएँ (ITeS) — ITeS, जिसे व्यावसाय प्रक्रिया आउटसोर्सिंग (Business Process Outsourcing - BPO) के नाम से भी जाना जाता है, के अंतर्गत व्यावसायिक प्रक्रियाओं को तृतीय-पक्ष सेवा प्रदाताओं को सौंपा जाता है, जो तकनीक का उपयोग करके ग्राहक सहायता, डेटा एंट्री, वित्तीय लेखा-जोखा आदि जैसी सेवाएँ प्रदान करते हैं।

1.1.1 प्रमुख घटक

हार्डवेयर — इसमें कंप्यूटर, सर्वर, संग्रहण उपकरण, नेटवर्किंग उपकरण और आईटी संचालन में प्रयुक्त सहायक यंत्र शामिल हैं।

सॉफ़्टवेयर — सूचना प्रौद्योगिकी मुख्यतः सॉफ़्टवेयर अनुप्रयोगों और ऑपरेटिंग सिस्टम्स पर आधारित होती है, जिनका उपयोग डेटा प्रसंस्करण, अनुप्रयोगों को चलाने और हार्डवेयर संसाधनों के प्रबंधन में होता है।

नेटवर्किंग — सूचना प्रौद्योगिकी उद्योग सशक्त नेटवर्क एवं दूरसंचार अवसंरचना पर आधारित है, जो उपकरणों को जोड़ने और डेटा के आदान-प्रदान को संभव बनाता है।

सेवाएँ — ITeS सेवा प्रदाता ग्राहक सहायता, तकनीकी सहायता, डेटा एंट्री, बैक-ऑफ़िस संचालन आदि जैसी सेवाएँ प्रदान करते हैं।

1.1.2 महत्व

आर्थिक योगदान — सूचना प्रौद्योगिकी एवं ITeS उद्योग वैश्विक अर्थव्यवस्था में महत्वपूर्ण योगदान देते हैं, जिससे विशाल राजस्व और रोजगार के अवसर उत्पन्न होते हैं।

नवाचार — IT तकनीकी नवाचारों की अग्रिम पंक्ति में है, जो कृत्रिम बुद्धिमत्ता, क्लाउड कंप्यूटिंग, साइबर सुरक्षा आदि क्षेत्रों में विकास को बढ़ावा देती है।

वैश्विक संपर्क — यह वैश्विक स्तर पर निर्बाध संचार और डेटा साझा करने को संभव बनाता है, जिससे अंतर्राष्ट्रीय व्यापार और सहयोग को गति मिलती है।

दक्षता — स्वचालन एवं डिजिटलीकरण के माध्यम से IT विभिन्न उद्योगों में संचालन दक्षता और उत्पादकता को बढ़ाता है।

1.1.3 प्रमुख प्रवृत्तियाँ (Key Trends)

क्लाउड कंप्यूटिंग (Cloud Computing) — क्लाउड-आधारित सेवाओं की ओर झुकाव संगठनों को अपनी सूचना प्रौद्योगिकी (आईटी) अवसंरचना को लचीले व लागत-दक्ष तरीके से विस्तार करने की सुविधा मिलती है।

कृत्रिम बुद्धिमत्ता और मशीन लर्निंग (Artificial Intelligence and Machine Learning) — एआई (AI) और एमएल (ML) डेटा विश्लेषण, स्वचालन और निर्णय-निर्माण की प्रक्रियाओं में क्रांति ला रहे हैं।

साइबर सुरक्षा (Cyber Security) — डिजिटलीकरण में वृद्धि के साथ, डेटा और प्रणालियों की रक्षा के लिए मजबूत साइबर सुरक्षा उपायों की आवश्यकता भी बढ़ रही है।

ऑनलाइन कार्य (दूरस्थ कार्य/Remote Work) — कोविड-19 महामारी ने दूरस्थ कार्य और डिजिटल सहयोग उपकरणों को अपनाने की गति को तीव्र किया।

बिग डेटा और एनालिटिक्स (Big Data and Analytics) — बड़े डेटा सेटों का विश्लेषण संगठनों को ग्राहक व्यवहार और बाजार प्रवृत्तियों को समझते हुए डेटा-आधारित निर्णय लेने में मदद करता है।

1.2 आईटी/आईटीईएस उद्योग के खंड

सूचना प्रौद्योगिकी (आईटी) और सूचना प्रौद्योगिकी-सक्षम सेवाएँ (आईटीईएस) उद्योग एक विशाल और विविध क्षेत्र है, जिसमें कई उपखंड शामिल हैं, जिनमें से प्रत्येक की अपनी विशिष्ट भूमिका और कार्य हैं। निम्नलिखित आईटी/आईटीईएस उद्योग के कुछ प्रमुख खंड हैं—

सॉफ़्टवेयर विकास (Software Development) — यह आईटी उद्योग के सबसे महत्वपूर्ण खंडों में से एक है। इसमें वे कंपनियाँ और पेशेवर शामिल होते हैं जो सॉफ़्टवेयर अनुप्रयोगों और प्रणालियों को डिज़ाइन, विकसित और बनाए रखते हैं। इसमें वेब विकास, मोबाइल ऐप विकास, एंटरप्राइज़ सॉफ़्टवेयर विकास आदि जैसी कई तकनीकों और प्लेटफ़ॉर्म्स को शामिल किया जाता है।

हार्डवेयर निर्माण (Hardware Manufacturing) — इस खंड में वे कंपनियाँ आती हैं जो प्रोसेसर, मेमोरी चिप्स, मदरबोर्ड और संग्रहण उपकरण जैसे कंप्यूटर हार्डवेयर घटकों के डिज़ाइन, निर्माण और वितरण से जुड़ी होती हैं। यह खंड कीबोर्ड, मॉनिटर और प्रिंटर जैसे कंप्यूटर परिधीय उपकरणों के उत्पादन को भी शामिल करता है।

आईटी परामर्श और सेवाएँ (IT Consulting and Services) — आईटी परामर्श फर्म व्यावसायों को तकनीकी अपनाने और कार्यान्वयन से संबंधित अहम निर्णय लेने में मार्गदर्शन प्रदान करती हैं। इनमें आईटी कार्यनीति विकास, प्रणाली एकीकरण और तकनीकी अनुकूलन जैसी सेवाएँ शामिल हो सकती हैं।

क्लाउड कंप्यूटिंग और होस्टिंग (Cloud Computing and Hosting) — इस खंड में क्लाउड सेवा प्रदाता शामिल हैं जो अधोसंरचना (IaaS), प्लेटफॉर्म (PaaS) और सॉफ़्टवेयर (SaaS) के रूप में सेवाएँ प्रदान करते हैं। ये सेवाएँ व्यावसायों को अपने

अनुप्रयोगों और डेटा को क्लाउड में होस्ट और प्रबंधित करने की सुविधा देती हैं, जिससे ऑन-प्रिमाइसेज़ अवसंरचना की आवश्यकता कम होती है।

साइबर सुरक्षा (Cyber Security) — साइबर सुरक्षा कंपनियाँ संगठनों की डिजिटल परिसंपत्तियों को साइबर खतरों जैसे वायरस, मैलवेयर, डेटा उल्लंघन और हैकिंग प्रयासों से सुरक्षा प्रदान करती हैं। ये नेटवर्क सुरक्षा, पहचान और प्रवेश प्रबंधन, और सुरक्षा परामर्श जैसी सेवाएँ प्रदान करती हैं।

बिजनेस प्रोसेस आउटसोर्सिंग (BPO) — बीपीओ कंपनियाँ व्यावसायों को विभिन्न बैक-ऑफ़िस और फ्रंट-ऑफ़िस सेवाएँ प्रदान करती हैं, जैसे ग्राहक सेवा, डेटा प्रविष्टि, मानव संसाधन, वित्त और लेखा। बीपीओ को आगे वॉइस-आधारित (कॉल सेंटर) और नॉन-वॉइस-आधारित सेवाओं में विभाजित किया जा सकता है।

डिजिटल विपणन और विज्ञापन (Digital Marketing and Advertising) — यह खंड उन एजेंसियों और पेशेवरों से संबंधित है जो सर्च इंजन ऑप्टिमाइज़ेशन (SEO), सोशल मीडिया मार्केटिंग, कंटेंट मार्केटिंग और ऑनलाइन विज्ञापन जैसी सेवाएँ प्रदान करते हैं। वे व्यावसायों को अपने उत्पादों और सेवाओं को ऑनलाइन बढ़ावा देने में मदद करते हैं।

ई-कॉमर्स और ऑनलाइन खुदरा व्यापार (E-Commerce and Online Retail) — ई-कॉमर्स कंपनियाँ ऑनलाइन मार्केटप्लेस का संचालन करती हैं जहाँ उपभोक्ता उत्पाद और सेवाएँ खरीद सकते हैं। इस खंड में ऑनलाइन रिटेलर, भुगतान गेटवे, लॉजिस्टिक्स प्रदाता और ई-कॉमर्स तकनीकी कंपनियाँ शामिल हैं।

कृत्रिम बुद्धिमत्ता (AI) और मशीन लर्निंग (ML) — एआई और एमएल कंपनियाँ ऐसे एल्गोरिद्म और प्रणालियाँ विकसित करती हैं जो डेटा का विश्लेषण कर सकती हैं, पूर्वानुमान कर सकती हैं और कार्यों को स्वचालित कर सकती हैं। इस खंड का उपयोग स्वास्थ्य सेवा, वित्त, और स्वायत्त वाहनों जैसे क्षेत्रों में होता है।

दूरसंचार (Telecommunications) — दूरसंचार कंपनियाँ आवाज़ और डेटा संप्रेषण की सुविधा के लिए अवसंरचना और सेवाएँ प्रदान करती हैं। इसमें इंटरनेट सेवा प्रदाता (ISP), मोबाइल नेटवर्क ऑपरेटर और 5G तकनीक के विकास से जुड़ी कंपनियाँ शामिल हैं।

डेटा विश्लेषण और व्यावसायिक बुद्धिमत्ता (Data Analytics and Business Intelligence) — यह खंड उन कंपनियों से संबंधित है जो डेटा संग्रह, विश्लेषण और व्याख्या में विशेषज्ञ हैं, जिससे व्यावसाय डेटा-आधारित निर्णय ले सकें। इसमें डेटा विश्लेषण उपकरण, डेटा वेयरहाउसिंग और डेटा विज़ुअलाइज़ेशन समाधान शामिल होते हैं।

गेमिंग और मनोरंजन (Gaming and Entertainment) — इस खंड की कंपनियाँ वीडियो गेम, इंटरैक्टिव एंटरटेनमेंट और डिजिटल मीडिया सामग्री का विकास और प्रकाशन करती हैं। इसमें ई-स्पोर्ट्स संगठन और स्ट्रीमिंग प्लेटफ़ॉर्म भी शामिल हैं।

हेल्थकेयर आईटी (Healthcare IT) — हेल्थकेयर आईटी कंपनियाँ स्वास्थ्य क्षेत्र के लिए प्रौद्योगिकी समाधान प्रदान करती हैं, जिनमें इलेक्ट्रॉनिक हेल्थ रिकॉर्ड्स (EHR), टेलीमेडिसिन प्लेटफ़ॉर्म, मेडिकल इमेजिंग सॉफ़्टवेयर और हेल्थकेयर एनालिटिक्स शामिल हैं।

फिनटेक (FinTech) — फिनटेक कंपनियाँ वित्तीय सेवाओं को तकनीक के माध्यम से उपलब्ध कराती हैं, जैसे डिजिटल बैंकिंग, भुगतान प्रसंस्करण, व्यक्ति से व्यक्ति ऋण (peer-to-peer lending), और ब्लॉकचेन आधारित समाधान जैसे क्रिप्टोकरेंसी।

एडटेक (EdTech) — एडटेक कंपनियाँ शिक्षा और प्रशिक्षण हेतु तकनीकी समाधान प्रदान करती हैं, जिनमें ऑनलाइन पाठ्यक्रम, ई-लर्निंग प्लेटफ़ॉर्म और शैक्षणिक सॉफ़्टवेयर शामिल हैं।

आईटी/आईटीईएस उद्योग में अनेक खंड हैं और प्रत्येक खंड डिजिटल परिदृश्य को आकार देने तथा विभिन्न आर्थिक क्षेत्रों में नवाचार को प्रेरित करने में महत्वपूर्ण भूमिका निभाता है।

अभ्यास 1.1

- IT/ITeS के प्रमुख घटकों की सूची बनाएँ।
- IT/ITeS की प्रमुख प्रवृत्तियों (ट्रेंड्स) की सूची बनाएँ।
- IT/ITeS उद्योग के प्रमुख खंडों की सूची बनाएँ।

आईटी/आईटीईएस उद्योग की प्रासंगिकता

सूचना प्रौद्योगिकी (आईटी) और सूचना प्रौद्योगिकी-सक्षम सेवाएँ (आईटीईएस) उद्योग आज की दुनिया में अत्यंत प्रासंगिक बना हुआ है, और हाल के वर्षों में इसकी महत्ता और अधिक बढ़ी है। नीचे दिए गए बिंदु बताते हैं कि आईटी/आईटीईएस उद्योग क्यों आज भी अत्यंत महत्वपूर्ण है—

डिजिटल रूपांतरण (Digital Transformation) — आईटी/आईटीईएस उद्योग व्यावसायों और संगठनों को डिजिटल रूपांतरण की प्रक्रिया अपनाने में मदद करता है। इसमें क्लाउड कंप्यूटिंग, बिग डेटा एनालिटिक्स, कृत्रिम बुद्धिमत्ता (AI), इंटरनेट ऑफ थिंग्स (IoT) जैसी तकनीकों को अपनाना शामिल है। ये तकनीकें कंपनियों को अधिक दक्ष, प्रतिस्पर्धी और ग्राहक-केन्द्रित बनाती हैं।

वैश्विक संपर्क (Global Connectivity) — आईटी/आईटीईएस वैश्विक संपर्क और सहयोग को संभव बनाता है। संप्रेषण उपकरणों, सॉफ़्टवेयर और अवसंरचना के विकास के माध्यम से व्यावसाय अब वैश्विक स्तर पर संचालन कर सकते हैं, नए बाजारों तक पहुँच सकते हैं और व्यापक ग्राहक आधार बना सकते हैं।

रोजगार सृजन (Job Creation) — यह उद्योग वैश्विक स्तर पर रोजगार का एक प्रमुख स्रोत है। यह न केवल सॉफ़्टवेयर विकास और आईटी सहायता में, बल्कि डिजिटल विपणन, डेटा विश्लेषण, साइबर सुरक्षा और परियोजना प्रबंधन जैसे क्षेत्रों में भी रोजगार उत्पन्न करता है। यह रोजगार विभिन्न कौशल स्तरों को समेटता है—प्रवेश स्तर से लेकर अत्यधिक विशेषज्ञ भूमिकाओं तक।

नवाचार (Innovation) — आईटी/आईटीईएस नवाचार के अग्रिम पंक्ति में है। तकनीकी कंपनियाँ लगातार संभावनाओं की बॉर्डर्स को आगे बढ़ा रही हैं, जिससे हार्डवेयर, सॉफ़्टवेयर और सेवाओं में प्रगति हो रही है। 5G तकनीक, क्वांटम कंप्यूटिंग और स्वायत्त वाहनों जैसे नवाचारों की जड़ें आईटी क्षेत्र में हैं।

आर्थिक विकास (Economic Growth) — यह उद्योग वैश्विक अर्थव्यवस्था में महत्वपूर्ण योगदान करता है। यह निवेश आकर्षित कर, राजस्व उत्पन्न कर और उद्यमिता को प्रोत्साहित कर आर्थिक विकास को गति देता है। कई देश इसे एक कार्यनीतिक क्षेत्र मानते हुए इसे बढ़ावा देते हैं।

ई-गवर्नेंस (E-Government) — विश्वभर में सरकारों ने सार्वजनिक सेवाओं के वितरण और शासन में सुधार के लिए आईटी/आईटीईएस को अपनाया है। इससे पारदर्शिता, दक्षता और नागरिक सहभागिता में वृद्धि हुई है।

शिक्षा और प्रशिक्षण (Education and Training) — आईटी/आईटीईएस तकनीकी शिक्षा, प्रशिक्षण और स्किल डेवलपमेंट में महत्वपूर्ण भूमिका निभाता है, जिससे युवाओं को भविष्य की डिजिटल दुनिया के लिए तैयार किया जा सके।

साइबर सुरक्षा (Cyber Security) — डिजिटल तकनीकों पर बढ़ती निर्भरता के कारण साइबर सुरक्षा अत्यावश्यक हो गई है। आईटी/आईटीईएस उद्योग व्यावसायों, व्यक्तियों और सरकारों को साइबर खतरों से सुरक्षित रखने हेतु सुरक्षा उपायों के विकास और कार्यान्वयन में महत्वपूर्ण भूमिका निभाता है।

स्वास्थ्य सेवा और जीवन विज्ञान (Health Care and Life Sciences) — इस उद्योग से हेल्थकेयर सूचना प्रणाली, टेलीमेडिसिन समाधान, जीनोमिक्स अनुसंधान और औषधि खोज तकनीकों के विकास के माध्यम से स्वास्थ्य सेवा और जीवन विज्ञान क्षेत्रों में महत्वपूर्ण योगदान दिया गया है।

पर्यावरण में स्थिरता (Environmental Sustainability) — आईटी/आईटीईएस दक्ष संसाधन प्रबंधन, दूरस्थ कार्य को बढ़ावा देने और हरित तकनीकों के विकास के माध्यम से पर्यावरण की स्थिरता में भी योगदान कर सकता है।

आईटी/आईटीईएस उद्योग डिजिटल परिवर्तन को आगे बढ़ाने, नवाचार को प्रोत्साहित करने, रोजगार सृजन, आर्थिक विकास में योगदान देने और सामाजिक चुनौतियों का समाधान प्रस्तुत करने के कारण अत्यंत प्रासंगिक बना हुआ है। जैसे-जैसे तकनीक आगे बढ़ेगी, इस उद्योग का महत्व और भी अधिक बढ़ेगा और इसका भविष्य में गहराई से प्रभाव होगा।

1.3 भारत में आईटी अनुप्रयोग विकास उद्योग

आईटी अनुप्रयोग विकास भारत में एक फलता-फूलता हुआ उद्योग है और पिछले कई दशकों से देश की आर्थिक वृद्धि का एक महत्त्वपूर्ण प्रेरक रहा है। भारत का आईटी क्षेत्र, जिसे प्रायः **भारतीय आईटी उद्योग** कहा जाता है, अपने **सॉफ्टवेयर विकास, आईटी सेवाओं** और **बिज़नेस प्रोसेस आउटसोर्सिंग (BPO)** क्षमताओं के लिए प्रसिद्ध है। भारत में आईटी अनुप्रयोग विकास उद्योग के कुछ प्रमुख पहलू निम्नलिखित हैं—

सॉफ्टवेयर सेवाएँ — भारतीय आईटी कंपनियाँ सॉफ्टवेयर विकास में अपने कौशल के लिए जानी जाती हैं, जिसमें वेब और मोबाइल अनुप्रयोग विकास भी शामिल है। वे अनुकूलित सॉफ्टवेयर विकास से लेकर अनुप्रयोग अनुरक्षण एवं सहायता तक की सेवाएँ प्रदान करती हैं।

वैश्विक उपस्थिति — भारतीय आईटी कंपनियाँ वैश्विक स्तर पर सशक्त उपस्थिति रखती हैं और विभिन्न उद्योगों के अंतरराष्ट्रीय ग्राहकों को सेवा देती हैं। उन्होंने अनेक देशों में विकास केंद्र स्थापित किए हैं और विविध तकनीकी मंचों पर आधारित परियोजनाओं पर कार्य करती हैं।

प्रौद्योगिकी स्टैक — भारत का आईटी उद्योग Java, Python, JavaScript, .NET आदि सहित विभिन्न प्रोग्रामिंग भाषाओं और प्रौद्योगिकियों में दक्ष है। इसके अतिरिक्त, वे कृत्रिम बुद्धिमत्ता (AI), ब्लॉकचेन तथा इंटरनेट ऑफ थिंग्स (IoT) जैसी उभरती प्रौद्योगिकियों में भी विशेषज्ञता रखते हैं।

आउटसोर्सिंग और ऑफशोरिंग — लागत प्रभावशीलता, दक्ष प्रतिभा और गुणवत्तापूर्ण परिणामों के कारण अनेक अंतरराष्ट्रीय कंपनियाँ अपने आईटी अनुप्रयोग विकास कार्य भारतीय कंपनियों को आउटसोर्स करती हैं। भारत में एक विशाल प्रशिक्षित सॉफ्टवेयर इंजीनियरों और डेवलपरों का समूह उपलब्ध है, जो वैश्विक मानकों पर खरा उतरता है।

स्टार्टअप्स और नवाचार — भारत में स्टार्टअप पारिस्थितिकी तंत्र तेज़ी से विकसित हो रहा है, जहाँ कई नई कंपनियाँ अनुप्रयोग विकास और नवाचारी तकनीकों पर केंद्रित हैं। बेंगलुरु, हैदराबाद और पुणे जैसे शहर तकनीकी स्टार्टअप्स के प्रमुख केंद्र बन चुके हैं।

शिक्षा और प्रशिक्षण — भारत की शिक्षा प्रणाली मजबूत है, जो प्रतिवर्ष बड़ी संख्या में अभियंत्रण एवं कंप्यूटर विज्ञान के स्नातक तैयार करती है। इन स्नातकों में से कई सॉफ्टवेयर विकास में प्रशिक्षण प्राप्त कर आईटी उद्योग में प्रवेश करते हैं।

सरकारी पहलें — भारत सरकार ने आईटी उद्योग को प्रोत्साहित करने हेतु अनेक पहलें शुरू की हैं, जैसे कि *"डिजिटल इंडिया"* अभियान, जिसका उद्देश्य सरकारी सेवाओं का डिजिटलीकरण करना और देशभर में प्रौद्योगिकी को बढ़ावा देना है।

चुनौतियाँ — अपनी सफलता के बावजूद, भारतीय आईटी उद्योग को बढ़ती प्रतिस्पर्धा, श्रम लागत में वृद्धि, डेटा सुरक्षा संबंधी चिंताएँ, और तकनीकी उन्नयन की निरंतर आवश्यकता जैसी चुनौतियों का सामना करना पड़ता है।

गुणवत्ता मानक — वैश्विक प्रतिष्ठा को बनाए रखने के लिए भारतीय आईटी कंपनियाँ प्रायः अंतरराष्ट्रीय गुणवत्ता मानकों और प्रमाणनों जैसे ISO तथा CMMI (Capability Maturity Model Integration) का पालन करती हैं।

वैश्विक प्रभाव — भारतीय आईटी कंपनियों ने विश्वभर में व्यावसायों को डिजिटल रूपांतरण की ओर अग्रसर करने में महत्त्वपूर्ण भूमिका निभाई है। इन्होंने वित्त, स्वास्थ्य, ई-कॉमर्स और विनिर्माण जैसे क्षेत्रों के लिए अनुप्रयोग विकसित कर उन्हें सक्षम और गतिशील बनाया है।

आईटी अनुप्रयोग विकास भारत में एक **गतिशील और प्रभावशाली क्षेत्र** है, जो न केवल देश की अर्थव्यवस्था में महत्वपूर्ण योगदान देता है बल्कि भारत को एक **वैश्विक प्रौद्योगिकी केंद्र** के रूप में स्थापित करता है। यह क्षेत्र तकनीक की प्रगति और अंतरराष्ट्रीय मांग के साथ निरंतर विकसित हो रहा है।

1.4 आईटी अनुप्रयोग विकास के उपक्षेत्र

आईटी अनुप्रयोग विकास एक व्यापक क्षेत्र है, जिसमें कई उपक्षेत्र सम्मिलित हैं, जो सॉफ़्टवेयर अनुप्रयोगों के निर्माण के विशिष्ट पहलुओं पर केंद्रित होते हैं। आईटी अनुप्रयोग विकास के कुछ सामान्य उपक्षेत्र निम्नलिखित हैं—

वेब विकास — वेब डेवलपर्स वेबसाइटों और वेब अनुप्रयोगों के निर्माण में विशेषज्ञ होते हैं। वे HTML, CSS, JavaScript और विभिन्न वेब विकास फ्रेमवर्क्स का उपयोग कर इंटरैक्टिव और उत्तरदायी वेब अनुप्रयोगों का निर्माण करते हैं।

मोबाइल ऐप विकास — मोबाइल ऐप डेवलपर्स स्मार्टफोन और टैबलेट के लिए अनुप्रयोग विकसित करते हैं। इसमें iOS ऐप विकास (Swift या Objective-C का उपयोग) और Android ऐप विकास (Java या Kotlin का उपयोग) शामिल होता है।

डेस्कटॉप अनुप्रयोग विकास — इस उपक्षेत्र के डेवलपर्स ऐसे अनुप्रयोग विकसित करते हैं जो Windows, macOS या Linux जैसे डेस्कटॉप ऑपरेटिंग सिस्टम पर चलते हैं। इनमें सामान्यतः C++, C# या Java जैसी भाषाओं का उपयोग किया जाता है।

गेम विकास — गेम डेवलपर्स कंसोल, पीसी और मोबाइल उपकरणों के लिए वीडियो गेम बनाते हैं। इसमें गेम डिज़ाइन, ग्राफिक्स, ध्वनि और भौतिक अनुकरण जैसे तत्व (एलिमेंट) शामिल होते हैं।

डेटाबेस विकास — डेटाबेस डेवलपर्स डेटाबेस का डिज़ाइन, कार्यान्वयन और अनुरक्षण करते हैं। वे MySQL, PostgreSQL, Oracle या Microsoft SQL Server जैसे DBMS के साथ कार्य करते हैं ताकि डेटा भंडारण और पुनर्प्राप्ति दक्षता सुनिश्चित हो सके।

एंटरप्राइज़ अनुप्रयोग विकास — एंटरप्राइज़ डेवलपर्स व्यावसायों के लिए विशेष सॉफ्टवेयर समाधान तैयार करते हैं, जैसे ग्राहक संबंध प्रबंधन (CRM), एंटरप्राइज़ संसाधन नियोजन (ERP) सॉफ़्टवेयर और संचालन प्रबंधन के अन्य उपकरण।

क्लाउड अनुप्रयोग विकास — इस उपक्षेत्र में डेवलपर्स क्लाउड कंप्यूटिंग संसाधनों और सेवाओं का उपयोग कर अनुप्रयोग तैयार करते हैं। वे Amazon Web Services (AWS), Microsoft Azure या Google Cloud Platform (GCP) जैसे प्लेटफ़ॉर्म्स का उपयोग करते हैं।

एंबेडेड सिस्टम विकास — एंबेडेड सिस्टम डेवलपर्स ऐसे अनुप्रयोगों पर कार्य करते हैं जो माइक्रोकंट्रोलर या IoT उपकरणों जैसे एंबेडेड हार्डवेयर पर चलते हैं। ये उपभोक्ता इलेक्ट्रॉनिक्स से लेकर औद्योगिक स्वचालन तक में प्रयुक्त होते हैं।

एआई और मशीन लर्निंग विकास — एआई व मशीन लर्निंग डेवलपर्स ऐसे अनुप्रयोगों (Applications) का निर्माण करते हैं जिनमें कृत्रिम बुद्धिमत्ता और मशीन लर्निंग एल्गोरिद्म का उपयोग किया जाता है। उदाहरणस्वरूप चैटबॉट्स, अनुशंसा प्रणाली, छवि पहचान सॉफ़्टवेयर आदि।

DevOps और निरंतर एकीकरण/निरंतर परिनियोजन (CI/CD) — DevOps इंजीनियर एवं CI/CD विशेषज्ञ विकास, परीक्षण और परिनियोजन की प्रक्रियाओं को स्वचालित और सुगम बनाते हैं। वे Docker, Kubernetes, Jenkins और Git जैसे उपकरणों के साथ कार्य करते हैं।

ब्लॉकचेन विकास — ब्लॉकचेन डेवलपर्स Ethereum, Hyperledger या Binance Smart Chain जैसे प्लेटफॉर्म्स पर विकेंद्रीकृत अनुप्रयोग (DApps) और स्मार्ट कॉन्ट्रैक्ट्स का निर्माण करते हैं।

सुरक्षा अनुप्रयोग विकास — सुरक्षा-केंद्रित डेवलपर्स ऐसे अनुप्रयोग और उपकरण बनाते हैं जो प्रणालियों और डेटा को साइबर खतरों से सुरक्षित रखते हैं, जैसे एंटी वायरस सॉफ़्टवेयर, फ़ायरवॉल्स, घुसपैठ पहचान प्रणाली, एन्क्रिप्शन उपकरण आदि।

इन उपक्षेत्रों में अक्सर एक-दूसरे से अंतःसंबंध होता है और अनेक डेवलपर एक से अधिक क्षेत्रों में कौशल अर्जित करते हैं। उपक्षेत्र का चयन डेवलपर की रुचि, लक्ष्य और परियोजनाओं की मांग पर निर्भर करता है।

1.5 आईटी उद्योग में अनुप्रयोग विकास के विभिन्न प्रकार

आईटी उद्योग में अनुप्रयोग विकास एक विस्तृत क्षेत्र है, जिसमें विभिन्न आवश्यकताओं और चुनौतियों के समाधान हेतु अनेक प्रकार के सॉफ़्टवेयर समाधान विकसित किए जाते हैं। आईटी उद्योग में अनुप्रयोग विकास के कुछ प्रकार निम्नलिखित हैं— वेब अनुप्रयोग विकास, मोबाइल अनुप्रयोग विकास, गेम विकास, क्लाउड अनुप्रयोग विकास, स्वास्थ्य एवं चिकित्सा सॉफ़्टवेयर विकास, वित्तीय सॉफ़्टवेयर विकास, शैक्षणिक सॉफ़्टवेयर विकास, ई-कॉमर्स सॉफ़्टवेयर विकास। यह क्षेत्र उभरती प्रौद्योगिकियों और बदलती व्यावसायिक आवश्यकताओं के साथ निरंतर विकसित हो रहा है।

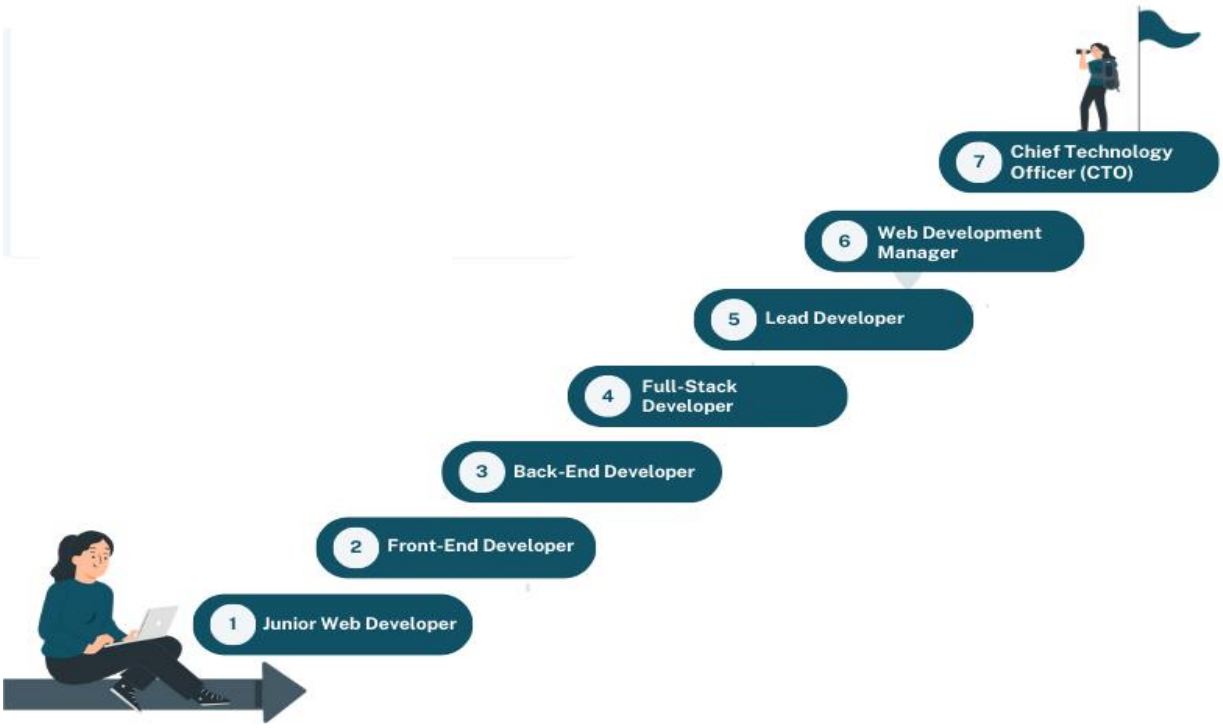
अभ्यास 1.2

1. आईटी अनुप्रयोग विकास के पाँच उपक्षेत्रों की सूची बनाइए।
2. आईटी उद्योग में अनुप्रयोग विकास के विभिन्न प्रकारों की सूची बनाइए।
3. भारत में आईटी अनुप्रयोग विकास उद्योग के किसी भी तीन प्रमुख पहलुओं की सूची बनाइए।

वेब डेवलपर के लिए कैरियर पथ और उन्नति के अवसर

वेब डेवलपर के रूप में कैरियर अनेक प्रकार की उन्नति के अवसर प्रदान करता है और यह आपकी रुचियों, कौशलों तथा कैरियर लक्ष्यों पर निर्भर करते हुए कई दिशाओं में आगे बढ़ सकता है। वेब डेवलपर्स के लिए संभावित कैरियर पथों और विकास के अवसरों का एक संक्षिप्त अवलोकन **चित्र 1.2** में दर्शाया गया है।

वेब डेवलपर का कैरियर एक जूनियर वेब डेवलपर के रूप में आरंभ होता है और धीरे-धीरे मुख्य तकनीकी अधिकारी (Chief Technology Officer) के स्तर तक पहुँच सकता है। इस क्षेत्र की एक विशेषता यह भी है कि आप स्वतंत्र (फ्रीलांस) वेब डेवलपर के रूप में कार्य कर सकते हैं, ग्राहकों के लिए परियोजनाएँ पूर्ण करके अपना पोर्टफोलियो बना सकते हैं। यदि आपके अंदर उद्यमशीलता की भावना है, तो आप अपनी स्वयं की वेब डेवलपमेंट एजेंसी भी आरंभ कर सकते हैं। वेब डेवलपमेंट के क्षेत्र से प्रायः दूरस्थ कार्य (Remote Work) की सुविधा मिलती है, जिससे आप विश्वभर के ग्राहकों और कंपनियों के साथ लचीले तरीके से कार्य कर सकते हैं। ध्यान रखें कि वेब डेवलपमेंट के क्षेत्र में व्यक्तिगत विकास और प्रगति के लिए निरंतर अध्ययन, अनुकूलनशीलता और नई तकनीकों व चुनौतियों को अपनाने की तत्परता आवश्यक होती है। एक सशक्त पोर्टफोलियो का निर्माण, सर्वोत्तम कार्य पद्धतियों में दक्षता तथा उद्योग की नवीनतम प्रवृत्तियों से जुड़े रहना इस क्षेत्र में सफलता प्राप्त करने में सहायक सिद्ध होगा।



चित्र 1.2— वेब डेवलपर के लिए कैरियर पथ और उन्नति के अवसर

1.6 वेब डेवलपर की भूमिकाएँ और उत्तरदायित्व

वेब डेवलपर की भूमिकाएँ और उत्तरदायित्व कार्य की प्रकृति, संस्था और प्रयुक्त तकनीकी स्टैक के आधार पर विभिन्न हो सकते हैं। तथापि, वेब डेवलपर्स से संबंधित कुछ सामान्य भूमिकाएँ और उत्तरदायित्व इस प्रकार हैं—

फ्रंट-एंड डेवलपमेंट

- **प्रयोक्ता इंटरफेस (UI) डिज़ाइन** — आकर्षक और प्रयोक्ता अनुकूल वेब इंटरफेस का निर्माण करना।
- **HTML/CSS** — वेबसाइट की रूपरेखा और साज-सज्जा के लिए HTML और CSS कोड लिखना और उनका अनुरक्षण करना।
- **JavaScript** — इंटरएक्टिविटी और प्रयोक्ता अनुभव बढ़ाने के लिए क्लाइंट-साइड स्क्रिप्टिंग विकसित करना।
- **उत्तरदायी डिज़ाइन (Responsive Design)** — यह सुनिश्चित करना कि वेबसाइट विभिन्न उपकरणों और स्क्रीन आकारों पर सुगमता से कार्य करे।

बैक-एंड डेवलपमेंट

- **सर्वर-साइड स्क्रिप्टिंग** — डेटा प्रोसेसिंग और व्यावसायिक तर्कों (Business Logic) को नियंत्रित करने के लिए सर्वर पर कार्यरत कोड लिखना।
- **डेटाबेस प्रबंधन** — वेब अनुप्रयोगों के लिए डेटाबेस का डिज़ाइन, कार्यान्वयन और अनुरक्षण करना।
- **एपीआई विकास** — फ्रंट-एंड और बैक-एंड सिस्टम के बीच डेटा के आदान-प्रदान के लिए एपीआई का निर्माण और एकीकरण करना।
- **सुरक्षा** — वेब अनुप्रयोगों को हैकिंग और डेटा उल्लंघनों जैसे खतरों से सुरक्षित रखने के लिए सुरक्षा उपाय लागू करना।

फुल-स्टैक डेवलपमेंट

फ्रंट-एंड और बैक-एंड दोनों विकास क्षमताओं को सम्मिलित कर सम्पूर्ण वेब एप्लीकेशन विकास प्रक्रिया पर कार्य करना।

सारांश

- आईटी/आईटीईएस (IT/ITeS) उद्योग में प्रौद्योगिकी, डेटा प्रबंधन और डिजिटल सेवाएँ सम्मिलित हैं।
- आईटी में हार्डवेयर, सॉफ़्टवेयर, नेटवर्किंग और आईटी सेवाएँ शामिल होती हैं।
- आईटीईएस का आशय है — तकनीक के माध्यम से व्यावसायिक प्रक्रियाओं की आउटसोर्सिंग।
- प्रमुख घटकों में कंप्यूटर, सॉफ़्टवेयर और नेटवर्किंग उपकरण आते हैं।
- यह उद्योग आर्थिक विकास, नवाचार और वैश्विक जुड़ाव को बढ़ावा देता है।
- प्रमुख प्रवृत्तियों में क्लाउड कंप्यूटिंग, कृत्रिम बुद्धिमत्ता (AI), साइबर सुरक्षा और दूरस्थ कार्य (Remote work) शामिल हैं।
- आईटी एप्लीकेशन विकास में वेब और मोबाइल विकास जैसे विविध उप-क्षेत्र सम्मिलित हैं।
- भारतीय आईटी उद्योग सॉफ़्टवेयर विकास, आईटी सेवाओं और बीपीओ (BPO) के लिए प्रसिद्ध है।
- वेब डेवलपर्स फ्रंट-एंड, बैक-एंड और फुल-स्टैक विकास में कार्य करते हैं।
- उद्योग की प्रवृत्तियों से अपडेट रहना वेब डेवलपर्स के लिए अत्यावश्यक है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न (MCQ)

1. IT का पूर्ण रूप क्या है?
   (क) इंटरनेट टेक्नोलॉजी
   (ख) सूचना प्रौद्योगिकी
   (ग) इनोवेटिव टेक्नोलॉजी
   (घ) इंटरनेशनल टेक्नोलॉजी

2. निम्नलिखित में से कौन आईटी/आईटीईएस उद्योग का एक प्रमुख घटक है?
   (क) कृषि
   (ख) विनिर्माण
   (ग) हार्डवेयर
   (घ) निर्माण

3. आईटी/आईटीईएस उद्योग में आईटी परामर्श कंपनियों की मुख्य भूमिका क्या है?
   (क) सॉफ़्टवेयर विकास
   (ख) ग्राहक सहायता
   (ग) परामर्श सेवाएँ
   (घ) डेटा एंट्री
4. आईटी उद्योग में दूरस्थ कार्य को अपनाने में किस प्रवृत्ति से तेजी आई है?
   (क) क्लाउड कंप्यूटिंग
   (ख) कृत्रिम बुद्धिमत्ता
   (ग) साइबर सुरक्षा
   (घ) ऑनलाइन कार्य

5. आईटी/आईटीईएस के संदर्भ में BPO का क्या अर्थ है?
   (क) बिजनेस प्रोसेस ऑप्टिमाइजेशन
   (ख) बिजनेस प्रोसेस आउटसोर्सिंग
   (ग) बिजनेस प्रोसेस ऑर्गनाइज़ेशन
   (घ) बिजनेस प्रोसेस ओवरसाइट

6. आईटी/आईटीईएस उद्योग किस क्षेत्र में पर्यावरण की सततता को बढ़ावा देने में महत्वपूर्ण भूमिका निभाता है?
   (क) स्वास्थ्य सेवा
   (ख) दूरसंचार
   (ग) गेमिंग और मनोरंजन
   (घ) फिनटेक

7. आईटी एप्लीकेशन विकास उद्योग में तकनीकी स्टार्टअप के लिए प्रसिद्ध भारतीय शहर कौन-सा है?
   (क) नई दिल्ली
   (ख) मुंबई
   (ग) बेंगलुरु
   (घ) चेन्नई

8. आईटी एप्लीकेशन विकास का कौन-सा उपक्षेत्र ब्लॉकचेन प्लेटफार्मों पर विकेन्द्रीकृत एप्लिकेशन और स्मार्ट अनुबंधों के निर्माण पर केंद्रित होता है?
   (क) गेम डेवलपमेंट
   (ख) डेटाबेस डेवलपमेंट
   (ग) ब्लॉकचेन डेवलपमेंट
   (घ) मोबाइल ऐप डेवलपमेंट

9. आईटी उद्योग में एप्लीकेशन विकास के कुछ सामान्य प्रकार कौन-कौन से हैं?
   (क) गेम डेवलपमेंट, हेल्थकेयर सॉफ़्टवेयर डेवलपमेंट, क्लाउड एप्लीकेशन डेवलपमेंट
   (ख) ई-कॉमर्स, वित्तीय, शैक्षणिक सॉफ़्टवेयर डेवलपमेंट
   (ग) वेब, मोबाइल और डेटाबेस एप्लीकेशन डेवलपमेंट
   (घ) उपरोक्त सभी

10. वेब विकास में किस भूमिका का फोकस वेबसाइट के यूजर इंटरफेस और अनुभव पर होता है?
    (क) फ्रंट-एंड डेवलपर
    (ख) बैक-एंड डेवलपर
    (ग) फुल-स्टैक डेवलपर
    (घ) डेटाबेस डेवलपर

ख. रिक्त स्थान भरें

1. IT का अर्थ _____ Technology है।
2. ITeS को _____ Outsourcing (BPO) के नाम से भी जाना जाता है।
3. हार्डवेयर में कंप्यूटर, सर्वर, _____, नेटवर्किंग उपकरण और परिधीय डिवाइस शामिल होते हैं।

4. आईटी उद्योग मजबूत नेटवर्क और दूरसंचार अवसंरचना पर निर्भर करता है जो _____ को जोड़ता है और _____ स्थानांतरण को सुगम बनाता है।
5. क्लाउड _____ संगठनों को अपने आईटी अवसंरचना को लचीले एवं किफायती तरीके से बढ़ाने में सक्षम बनाता है।
6. AI और ML डेटा विश्लेषण, _____ और निर्णय-निर्धारण प्रक्रियाओं में क्रांतिकारी बदलाव ला रहे हैं।
7. _____ सेवाओं की ओर झुकाव व्यावसायों को ऑन-प्रिमाइसेस अवसंरचना की आवश्यकता को कम करने में मदद करता है।
8. वेब डेवलपर्स वेबसाइटों और _____ एप्लिकेशनों के निर्माण में विशेषज्ञता रखते हैं।
9. गेम डेवलपर्स विभिन्न प्लेटफार्मों जैसे कि कंसोल, पीसी और मोबाइल उपकरणों के लिए _____ तैयार करते हैं।
10. ब्लॉकचेन डेवलपर्स _____ जैसे प्लेटफार्मों पर विकेन्द्रीकृत एप्लिकेशन (DApps) और स्मार्ट अनुबंध विकसित करते हैं।

ग. सही या गलत बताइए (True/False)

1. आईटी/आईटीईएस उद्योग प्रौद्योगिकी, डेटा प्रबंधन और डिजिटल सेवाओं से संबंधित गतिविधियों की एक विस्तृत श्रृंखला को सम्मिलित करता है।
2. हार्डवेयर घटकों में कंप्यूटर, सर्वर, संग्रहण उपकरण, नेटवर्किंग उपकरण और आईटी संचालन में प्रयुक्त परिधीय (पेरिफेरल) शामिल हैं।
3. क्लाउड कंप्यूटिंग से संगठनों को अपने आईटी अवसंरचना को अव्यवस्थित और महंगे तरीके से बढ़ाने की सुविधा मिलती है।
4. एआई और एमएल का डेटा विश्लेषण और स्वचालन प्रक्रियाओं में कोई उपयोग नहीं है।
5. भारतीय आईटी उद्योग सॉफ़्टवेयर विकास, आईटी सेवाओं और बीपीओ के लिए जाना जाता है।
6. गेम डेवलपर्स केवल पीसी और कंसोल गेम्स पर ध्यान देते हैं, मोबाइल गेम्स पर नहीं।
7. भारतीय आईटी कंपनियाँ प्रायः ISO और CMMI जैसी अंतर्राष्ट्रीय गुणवत्ता मानकों और प्रमाणनों का पालन करती हैं।
8. वेब डेवलपर्स केवल वेबसाइट डिज़ाइन करते हैं, उन्हें कोडिंग करने की आवश्यकता नहीं होती।
9. वेब डेवलपर्स को नवीनतम उद्योग प्रवृत्तियों और तकनीकों से अपडेट रहने की आवश्यकता नहीं होती।
10. पर्यावरण की सततता को बढ़ावा देने में आईटी/आईटीईएस उद्योग का कोई महत्व नहीं है।

घ. लघु उत्तरात्मक प्रश्न

1. सूचना प्रौद्योगिकी (IT) उद्योग का मुख्य फोकस क्या है?
2. सूचना प्रौद्योगिकी सक्षम सेवाएँ (ITeS) को परिभाषित कीजिए एवं एक उदाहरण दीजिए।
3. आईटी उद्योग में हार्डवेयर, सॉफ़्टवेयर और नेटवर्किंग से संबंधित एक-एक प्रमुख घटक का नाम बताइए।
4. आईटी/आईटीईएस उद्योग वैश्विक कनेक्टिविटी और सहयोग में किस प्रकार योगदान करता है?
5. आईटी उद्योग में किस प्रमुख प्रवृत्ति ने दूरस्थ कार्य को अपनाने में तेजी लाई है?
6. आईटी उद्योग में वेब डेवलपर्स की प्राथमिक जिम्मेदारियाँ क्या हैं?
7. आईटी/आईटीईएस उद्योग आर्थिक वृद्धि में किस प्रकार योगदान देता है?
8. इस अध्याय में वर्णित आईटी एप्लीकेशन विकास के दो उपक्षेत्रों के नाम बताइए।
9. वेब डेवलपर्स के लिए उद्योग प्रवृत्तियों से अपडेट रहना क्यों आवश्यक है?
10. आईटी/आईटीईएस उद्योग पर्यावरण की सततता में किस प्रकार योगदान करता है?

सत्र 2

वेब डिज़ाइन और विकास

एक बार एक लड़का था जिसका नाम अनिल था। वह इंटरनेट और वेबसाइटों के काम करने के तरीके को लेकर बहुत उत्सुक था। इसलिए उसने वेबसाइट बनाना सीखने के लिए अपने शिक्षक श्री विजय की कक्षा ली। अनिल ने "The Web Wiz" नामक एक सरल वेबसाइट बनाई, जो लोगों को वेबसाइट बनाना सिखाने में मदद करती थी। यह वेबसाइट स्कूल में लोकप्रिय हो गई क्योंकि इसमें आसान पाठ, रोचक प्रश्नोत्तरी और प्रश्न पूछने का स्थान था। उनकी वेबसाइट से कई विद्यार्थियों को मदद मिली, और यहाँ तक कि एक सामुदायिक केंद्र ने भी अनिल को वेब डिज़ाइन पर बोलने के लिए बुलाया। अंततः अनिल ने यह जाना कि जिज्ञासु रहना और नई चीज़ें आज़माने से बड़ी उपलब्धियों को अर्जित किया जा सकता है। वह इस बात से खुश था कि वह दूसरों की मदद कर पाया। जैसा कि चित्र 2.1 में दिखाया गया है।

चित्र 2.1— अनिल वेबसाइट बनाते हुए

इस अध्याय में, आप वेब डिज़ाइन और विकास, वेब डिज़ाइन की सीमाएँ, वेबसाइट निर्माण के लिए मुख्य पूर्वधारणाएँ तथा वेब डिज़ाइनिंग प्रक्रिया के इंटरफ़ेस के बारे में सीखेंगे।

2.1 वेब डिज़ाइन और विकास

वेब डिज़ाइन और विकास एक बहुस्तरीय प्रक्रिया है जिसमें वेबसाइटों को बनाना और उनका रखरखाव करना शामिल है।

क्या आप जानते हैं?
वेब डिज़ाइन और विकास एक व्यापक शब्द है जो वेबसाइट बनाने की पूरी प्रक्रिया को दर्शाता है।

इसमें कई चरणों को समाहित किया गया है, जैसे योजना बनाना, डिज़ाइन करना, कोडिंग करना और परीक्षण करना।

1. योजना और विश्लेषण

- सर्वप्रथम, वेबसाइट का उद्देश्य और लक्ष्य, इसकी मुख्य कार्यप्रणाली तथा लक्षित प्रयोक्ता समूह की पहचान करें।
- जानकारी एकत्र करने हेतु बाज़ार अनुसंधान और प्रतियोगी विश्लेषण करें।
- सामग्री की संरचना की योजना बनाएं और यह तय करें कि वेबसाइट पर कौन-सी जानकारी प्रस्तुत की जाएगी।

2. सूचना संरचना (Information Architecture)

- वेबसाइट की संरचना का एक दृश्य प्रतिनिधित्व बनाएं, जिसमें सभी पृष्ठों और उनके आपसी संबंधों की रूपरेखा हो।
- मुख्य पृष्ठों के प्रारूप या वायरफ़्रेम (Wireframe) बनाएं ताकि रूपरेखा और कार्यप्रणाली निर्धारित की जा सके।

3. डिज़ाइन

- वेबसाइट की दृश्य शैली विकसित करें, जिसमें रंग योजना, टाइपोग्राफ़ी (लेखनशैली), चित्र और ग्राफ़िक्स शामिल हों।
- मुख्य पृष्ठों के विस्तृत नमूने या प्रोटोटाइप बनाएं ताकि डिज़ाइन को दृष्टिगत रूप से देखा जा सके।

4. विकास (Development)

- **फ्रंट एंड विकास**—HTML (हाइपरटेक्स्ट मार्कअप भाषा), CSS (कैस्केडिंग स्टाइल शीट्स) और JavaScript का उपयोग करके प्रयोक्ता इंटरफ़ेस तैयार करें। यही वह भाग है जिससे प्रयोक्ता सीधे संवाद करता है।
- **बैक एंड विकास**—सर्वर साइड लॉजिक, डेटाबेस और एप्लीकेशन की कार्यप्रणाली बनाएं। सामान्य तकनीकों में PHP, Python, Ruby on Rails, Node.js आदि शामिल हैं।
- **सामग्री प्रबंधन प्रणाली (CMS)**—यदि आवश्यक हो, तो WordPress या Drupal जैसे CMS को एकीकृत करें ताकि गैर-तकनीकी प्रयोक्ता भी सामग्री का प्रबंधन कर सकें।
- **उत्तरदायी डिज़ाइन (Responsive Design)**—सुनिश्चित करें कि वेबसाइट विभिन्न उपकरणों और स्क्रीन आकारों पर सहजता से कार्य करे। इसके लिए प्रायः CSS फ्रेमवर्क का उपयोग किया जाता है।

5. सामग्री निर्माण (Content Creation)

- वेबसाइट में टेक्स्ट, चित्र, वीडियो और अन्य मीडिया सामग्री जोड़ें।
- खोज (सर्च) इंजन के लिए सामग्री को उपयुक्त कीवर्ड और मेटाडेटा का प्रयोग करके अनुकूलित करें।

6. परीक्षण (Testing)

- सभी कार्यप्रणालियों और विशेषताओं का परीक्षण करें ताकि वे इच्छित रूप से कार्य करें।
- यह सुनिश्चित करें कि वेबसाइट विभिन्न ब्राउज़र और उपकरणों पर सही कार्य करे।
- लोडिंग समय की जाँच करें और प्रदर्शन को अनुकूलित करें।
- वेबसाइट को सामान्य सुरक्षा खतरों से सुरक्षित बनाना सुनिश्चित करें।

7. परिनियोजन (Deployment)

- एक डोमेन नाम चुनें और वेब होस्टिंग की व्यवस्था करें।
- वेबसाइट की सभी फ़ाइलें होस्टिंग सर्वर पर अपलोड करें।
- डोमेन को होस्टिंग सर्वर से जोड़ने हेतु DNS सेटिंग्स को कॉन्फ़िगर करें।

8. लॉन्च

- एक बार फिर से लाइव सर्वर पर सब कुछ परीक्षण करें।
- वेबसाइट और सर्वर कॉन्फ़िगरेशन का बैकअप बनाएं।
- वेबसाइट लॉन्च के बारे में हितधारकों और प्रयोक्ताओं को सूचित करें।

9. अनुरक्षण और अद्यतन (Maintenance and Updates)

- नियमित रूप से सामग्री, सुरक्षा पैच और सॉफ़्टवेयर को अद्यतन करें ताकि वेबसाइट अद्यतित और सुरक्षित बनी रहे।
- वेबसाइट के प्रदर्शन और प्रयोक्ता प्रतिक्रिया की निगरानी करें ताकि निरंतर सुधार किया जा सके।
- भविष्य में आवश्यकतानुसार संवर्धन और विस्तार की योजना बनाएं।

वेब डिज़ाइन और विकास एक निरंतर प्रक्रिया है जिसमें बारीकियों पर ध्यान देना और प्रयोक्ता आवश्यकताओं तथा तकनीकी विकास के अनुसार सतत् सुधार आवश्यक है। डिज़ाइनर, डेवलपर और सामग्री निर्माताओं के बीच सहयोग एक सफल वेबसाइट निर्माण के लिए अत्यंत आवश्यक है।

2.2 वेब डिज़ाइन और विकास की अवधारणा

वेब डिज़ाइन और वेब विकास दो परस्पर संबंधित किन्तु भिन्न विषय हैं, जो वेबसाइटों और वेब एप्लीकेशनों को बनाने और उनका रखरखाव करने के लिए आवश्यक हैं।

2.2.1 वेब डिज़ाइन

वेब डिज़ाइन का आशय किसी वेबसाइट या वेब एप्लीकेशन के दृश्य और प्रयोक्ता इंटरफ़ेस (User Interface — UI) तत्वों को बनाने की प्रक्रिया से है। यह वेबसाइट के रूप, विन्यास और समग्र प्रस्तुति पर केंद्रित होता है।

क. **दृश्य तत्व (Visual Elements)**—इसमें रंग योजना, टाइपोग्राफ़ी, चित्र, ग्राफ़िक्स और अन्य दृश्य तत्वों का चयन शामिल है ताकि आकर्षक और सुसंगत डिज़ाइन बन सके।

ख. **प्रयोक्ता अनुभव (User Experience — UX)**—वेब डिज़ाइनर प्रयोक्ता के लिए सकारात्मक अनुभव निर्मित करने का प्रयास करते हैं, जैसे वेबसाइट को नेविगेट करना आसान हो, इंटरफ़ेस सहज हो, और मोबाइल व अन्य उपकरणों के लिए अनुकूलन हो।

ग. **लेआउट और संरचना**—डिज़ाइनर यह निर्धारित करते हैं कि सामग्री, बटन, मेनू और अन्य तत्वों को किस प्रकार पृष्ठ पर व्यवस्थित किया जाएगा ताकि प्रयोक्ता की सहभागिता और पहुँच सुगम हो।

घ. **उत्तरदायी डिज़ाइन (Responsive Design)**—यह सुनिश्चित करना कि डिज़ाइन विभिन्न स्क्रीन आकारों और उपकरणों पर अनुकूलित हो — इस अवधारणा को उत्तरदायी डिज़ाइन कहा जाता है।

ङ. **वायरफ़्रेम और प्रोटोटाइप**—डिज़ाइनर सामान्यतः वेबसाइट के लेआउट और कार्यक्षमता की योजना और परीक्षण के लिए वायरफ़्रेम और प्रोटोटाइप बनाते हैं।

च. **ग्राफ़िक डिज़ाइन**—कस्टम आइकन, लोगो और अन्य दृश्य तत्व (एलिमेंट) बनाने हेतु ग्राफ़िक डिज़ाइन कौशल की आवश्यकता होती है।

अभ्यास 2.1

- वेब डिज़ाइन और विकास की विभिन्न चरणों की सूची बनाएँ।
- वेब डिज़ाइन में प्रयुक्त विभिन्न तत्वों के नाम लिखें।

वेब विकास (Web Development)

दूसरी ओर, वेब विकास एक वेबसाइट या वेब एप्लीकेशन की कार्यप्रणाली और अंतःक्रियात्मकता (interactivity) को निर्मित करने की प्रक्रिया है। वेब डेवलपर डिज़ाइन को कार्यात्मक वेबसाइट में परिवर्तित करने हेतु प्रोग्रामिंग भाषाओं और फ्रेमवर्क्स का उपयोग करते हैं। वेब विकास के मुख्य आयाम इस प्रकार हैं —

क. **फ्रंट-एंड विकास**—फ्रंट-एंड डेवलपर क्लाइंट पक्ष (Client Side) पर कार्य करते हैं, और HTML, CSS, JavaScript कोड लिखते हैं जिससे यह तय किया जाता है कि वेबसाइट ब्राउज़र में कैसी दिखाई देगी और प्रयोक्ता किस प्रकार संवाद करेगा।

ख. **बैक-एंड विकास**—बैक-एंड डेवलपर वेबसाइट के सर्वर पक्ष पर कार्य करते हैं। वे डेटाबेस, सर्वर स्क्रिप्टिंग और सर्वर इन्फ्रास्ट्रक्चर का प्रबंधन करते हैं ताकि वेबसाइट डेटा संसाधित कर सके, प्रयोक्ता एकाउंट्स को संभाल सके और अन्य कार्य कर सके।

ग. **फुल स्टैक विकास**—कुछ डेवलपर फ्रंट-एंड और बैक-एंड दोनों में दक्ष होते हैं। इन्हें फुल स्टैक डेवलपर कहा जाता है, और वे एक वेब प्रोजेक्ट के सभी पहलुओं पर कार्य कर सकते हैं।

घ. **डेटाबेस प्रबंधन**—डेटा को प्रभावी रूप से स्टोर करना और पुनः प्राप्त करना अत्यंत महत्वपूर्ण है। वेब डेवलपर सामान्यतः MySQL, MongoDB जैसे डेटाबेस के साथ कार्य करते हैं।

ङ. **सुरक्षा (Security)**—डेवलपर्स को वेबसाइट और प्रयोक्ता डेटा को विभिन्न खतरों जैसे हैकिंग और डेटा चोरी से सुरक्षित रखने हेतु सावधानीपूर्वक सुरक्षा उपाय लागू करने होते हैं।

च. **परीक्षण और डिबगिंग**—डेवलपर कोड का परीक्षण और त्रुटि निवारण (debugging) करते हैं ताकि यह सुनिश्चित किया जा सके कि वेबसाइट सभी ब्राउज़र और उपकरणों पर सही कार्य करे।

छ. **परिनियोजन और अनुरक्षण**—विकास के उपरांत, वेबसाइट को वेब सर्वर पर परिनियोजित करना होता है और उसे अद्यतित और सुरक्षित बनाए रखने के लिए सतत अनुरक्षण आवश्यक होता है।

क्या आप जानते हैं?
वेब डिज़ाइन और विकास एक सहयोगात्मक प्रक्रिया है जिसमें डिज़ाइनर और डेवलपर मिलकर कार्य करते हैं ताकि वेबसाइटें आकर्षक, प्रयोक्ता अनुकूल और कार्यात्मक बन सकें।

वेब डिज़ाइन के मूल तत्व

वेब डिज़ाइन में कई मूलभूत तत्व (एलिमेंट) शामिल होते हैं जो मिलकर एक दृष्टिगत रूप से आकर्षक और प्रयोक्ता अनुकूल वेबसाइट के निर्माण में सहायक होते हैं। वेब डिज़ाइन के कुछ प्रमुख मूल तत्व निम्नलिखित हैं—

कुल विन्यास (Overall Layout) — वेबसाइट का समग्र विन्यास इस प्रकार हो सकता है—

- **ग्रिड संरचना** — सामग्री को व्यवस्थित करने और संपूर्ण वेबसाइट में एकरूपता बनाए रखने के लिए एक ग्रिड प्रणाली स्थापित करें।
- **उत्तरदायी डिज़ाइन (Responsive Design)** — यह सुनिश्चित करें कि इसका लेआउट विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार अनुकूल हो, जिससे प्रयोक्ता को डेस्कटॉप, टैबलेट और स्मार्टफ़ोन पर निर्बाध अनुभव प्राप्त हो।
- **श्वेत क्षेत्र (White Space)** — पठनीयता बढ़ाने और दृश्य अव्यवस्था को कम करने के लिए श्वेत स्थान (नेगेटिव स्थान) का प्रभावी उपयोग करें।

रंग योजना (Colour Scheme) —

- एक ऐसा रंग-पैलेट चुनें जो आपके ब्रांड के अनुरूप हो और इच्छित भाव या वातावरण को व्यक्त करे।
- शीर्षक, टेक्स्ट, लिंक, बटन और पृष्ठभूमि जैसे तत्वों में रंगों का सुसंगत रूप से प्रयोग करें।
- दृश्य बाधाओं वाले प्रयोक्ताओं सहित सभी के लिए पठनीयता सुनिश्चित करने हेतु पहुँच मानकों (Accessibility Standards) पर विचार करें।

अक्षर विन्यास (Typography) —

- शीर्षकों और मुख्य टेक्स्ट के लिए उपयुक्त फ़ॉन्ट का चयन करें, जिसमें पठनीयता और सौंदर्य का संतुलन हो।
- वेबसाइट में फ़ॉन्ट की एकरूपता बनाए रखें ताकि संपूर्ण डिज़ाइन में सामंजस्य बना रहे।

- बेहतर पठनीयता के लिए फ़ॉन्ट आकार और रिक्ति पर ध्यान दें, और विभिन्न स्क्रीन आकारों के लिए उत्तरदायी टाइपोग्राफ़ी (Responsive Typography) पर भी विचार करें।

नेविगेशन सामग्री (Navigation Content) — स्पष्ट और सहज नेविगेशन मेनू डिज़ाइन करें, जैसे शीर्ष नेविगेशन पट्टी (Top Menu Bar) या मोबाइल उपकरणों के लिए हैमबर्गर मेनू। यह सुनिश्चित करें कि नेविगेशन लेबल वर्णनात्मक और प्रयोक्ता-अनुकूल हों। वेबसाइट में प्रयोक्ताओं को जानकारी खोजने में सुविधा प्रदान करने हेतु ब्रेडक्रंब ट्रेल्स, साइटमैप्स और सर्च बार लागू करें।

सामग्री (Content) — उच्च गुणवत्ता वाली, आकर्षक और प्रासंगिक सामग्री तैयार करें जो लक्षित प्रयोक्ता समूह की आवश्यकताओं और रुचियों को संबोधित करती हो। जानकारी को प्रभावी और सरलता से प्रस्तुत करने के लिए छवियाँ, वीडियो और इन्फोग्राफ़िक्स जैसे मल्टीमीडिया तत्वों का प्रयोग करें। सामग्री को तार्किक रूप से अनुभागों और पृष्ठों में विभाजित करें ताकि प्रयोक्ता उसे आसानी से समझ और नेविगेट कर सकें।

चित्र और ग्राफ़िक्स (Imagery and Graphics) —

- ऐसे चित्र और ग्राफ़िक्स सम्मिलित करें जो आपके ब्रांड के अनुरूप हों और वेबसाइट की दृश्य अपील को बढ़ाएँ।
- वेबसाइट प्रदर्शन के लिए चित्रों को गुणवत्ता खोए बिना संपीड़ित (Compress) करें।
- सुनिश्चित करें कि चित्र विभिन्न उपकरणों और स्क्रीन आकारों पर उपयुक्त रूप से लोड हों — इसके लिए उत्तरदायी चित्र (Responsive Images) का उपयोग करें।

प्रयोक्ता संपर्क और प्रतिपुष्टि (User Interaction and Feedback) —

- बटन, फ़ॉर्म और सोशल मीडिया एकीकरण जैसे इंटरैक्टिव तत्व (एलिमेंट) सम्मिलित करें।
- प्रयोक्ता अनुभव बढ़ाने के लिए होवर प्रभाव, एनिमेशन और सूचनाओं के माध्यम से उन्हें प्रतिपुष्टि प्रदान करें।
- यह सुनिश्चित करें कि सभी फ़ॉर्म और इंटरैक्टिव तत्व (एलिमेंट) प्रयोक्ता अनुकूल और सुलभ (Accessible) हों।

संगति (Consistency) — वेबसाइट के सभी पृष्ठों में दृश्यात्मक और क्रियात्मक संगति बनाए रखें। डिज़ाइन मानकों की स्थापना एवं पालन के लिए स्टाइल गाइड और डिज़ाइन पैटर्न का उपयोग करें।

सुलभता (Accessibility) — यह सुनिश्चित करने हेतु पहुँच निर्देशों (जैसे WCAG) का पालन करें कि आपकी वेबसाइट विकलांग व्यक्तियों द्वारा भी प्रयोज्य हो। चित्रों के लिए वैकल्पिक टेक्स्ट (Alt Text) प्रदान करें, उचित कंट्रास्ट सुनिश्चित करें, और HTML के अर्थपूर्ण (Semantic) तत्वों का प्रयोग करें।

प्रदर्शन (Performance) — छवि अनुकूलन, लेज़ी लोडिंग (Lazy Loading), कैशिंग आदि तकनीकों द्वारा पृष्ठ लोड होने के समय को न्यूनतम करके वेबसाइट का प्रदर्शन अनुकूलित करें। सर्वोत्तम प्रदर्शन सुनिश्चित करने हेतु वेबसाइट को विभिन्न ब्राउज़रों और उपकरणों पर परीक्षण करें ताकि क्रॉस-कम्पैटिबिलिटी बनी रहे।

प्रभावी वेब डिज़ाइन स्थिर नहीं होता, बल्कि उसे लगातार मूल्यांकित और अद्यतन किया जाना चाहिए ताकि वह प्रयोक्ताओं की बदलती अपेक्षाओं और डिज़ाइन प्रवृत्तियों के अनुकूल बना रहे। नियमित रूप से प्रयोक्ता प्रतिपुष्टि एकत्र करना और प्रयोज्यता परीक्षण (Usability Testing) करना वेबसाइट के डिज़ाइन को परिष्कृत एवं बेहतर बनाने में सहायक हो सकता है।

2.3. वेब डिज़ाइन में सीमाएँ

वेब डिज़ाइन में बॉर्डर्स (Constraints) से आशय उन प्रतिबंधों या नियमों से है जिनके अंतर्गत डिज़ाइनरों को कार्य करना होता है ताकि वे प्रभावी और कार्यशील वेबसाइटें तैयार कर सकें। ये सीमाएँ विभिन्न स्रोतों से उत्पन्न हो सकती हैं और वेब डिज़ाइन के कई पक्षों को प्रभावित कर सकती हैं। वेब डिज़ाइन की कुछ सामान्य सीमाएँ निम्नलिखित हैं—

तकनीकी सीमाएँ (Technical Constraints) — डिज़ाइनरों को यह सुनिश्चित करना होता है कि वेबसाइट विभिन्न वेब ब्राउज़रों (जैसे Chrome, Firefox, Safari, Internet Explorer) पर सही तरीके से कार्य करे और समान रूप से प्रदर्शित हो। वेबसाइटें उत्तरदायी (Responsive) हों और डेस्कटॉप, लैपटॉप, टैबलेट तथा स्मार्टफोन जैसे विभिन्न उपकरणों के अनुरूप ढल सकें।

सुलभता की सीमाएँ (Accessibility Constraints) — डिज़ाइनरों को वेबसाइट को नि—शक्त व्यक्तियों के लिए भी प्रयोज्य बनाने हेतु पहुँच मानकों का पालन करना होता है। इसमें स्क्रीन रीडर, कीबोर्ड नेविगेशन और अन्य सहायक प्रौद्योगिकियों के लिए विचार करना शामिल है। टेक्स्ट और पृष्ठभूमि रंगों के बीच पर्याप्त कंट्रास्ट सुनिश्चित करना आवश्यक है ताकि दृश्य बाधाओं वाले प्रयोक्ता भी सामग्री को समझ सकें।

सामग्री से संबंधित सीमाएँ (Content Constraints) — वेबसाइटों में अक्सर विभिन्न प्रकार की सामग्री, जैसे टेक्स्ट, चित्र, वीडियो और इंटरैक्टिव तत्वों को समाहित करना होता है। डिज़ाइनरों को सामग्री को प्रभावी तरीके से व्यवस्थित और प्रस्तुत करना होता है। प्रयोक्ताओं को संतुलित मात्रा में जानकारी देना आवश्यक होता है ताकि वे अभिभूत हुए बिना आवश्यक जानकारी प्राप्त कर सकें।

ब्रांड और शैली की सीमाएँ (Brand and Style Constraints) — डिज़ाइनरों को ब्रांड दिशानिर्देशों का पालन करना होता है, जिनमें विशेष रंगों, फ़ॉन्ट्स, प्रतीकों (Logos) और अन्य ब्रांडिंग तत्वों का उपयोग शामिल होता है। संपूर्ण वेबसाइट में डिज़ाइन और प्रयोक्ता अनुभव में एकरूपता बनाए रखना आवश्यक होता है ताकि ब्रांड पहचान और प्रयोक्ता इससे परिचित बना रहे।

बजट सीमाएँ (Budget Constraints) — डिज़ाइनरों को प्रायः सीमित बजट में कार्य करना होता है, जो डिज़ाइन तत्वों के चयन, तकनीकी साधनों (Technology Stack) और परियोजना के विस्तार पर प्रभाव डाल सकता है।

उपयोगिता की सीमाएँ (Usability Constraints) — डिज़ाइनरों को ऐसी इंटरफ़ेस तैयार करनी होती है जो सहज, स्पष्ट और प्रयोक्ता अनुकूल हो। सकारात्मक प्रयोक्ता अनुभव के लिए सामग्री का तार्किक संगठन और प्रभावी नेविगेशन मेनू की रचना अनिवार्य होती है।

समय सीमाएँ (Time Constraints) — डिज़ाइनरों को अक्सर सीमित समय सीमा में कार्य करना होता है, जो डिज़ाइन की गहराई में जाकर अन्वेषण और परीक्षण करने की क्षमता को प्रभावित कर सकता है।

इन बॉर्डर्स को समझना और प्रभावी रूप से प्रबंधित करना वेब डिज़ाइनरों के लिए अत्यंत आवश्यक है ताकि वे ऐसे वेबसाइट विकसित कर सकें जो ग्राहकों और प्रयोक्ताओं दोनों की आवश्यकताओं को पूरा करें और उच्च गुणवत्ता बनाए रखें। इन बॉर्डर्स के बीच संतुलन बनाना रचनात्मकता, समस्या समाधान क्षमता, और वेब डिज़ाइन के सिद्धांतों व श्रेष्ठ प्रक्रियाओं की गहन समझ की अपेक्षा करता है।

2.4 वेबसाइट निर्माण हेतु प्रमुख पूर्वधारणाएँ

वेबसाइट का निर्माण करते समय यह आवश्यक है कि उसकी उद्दिष्ट उपलब्धियों को सुनिश्चित करने तथा लक्षित प्रयोक्ताओं की आवश्यकताओं की पूर्ति हेतु कुछ महत्वपूर्ण पूर्वधारणाओं और विचारों को ध्यान में रखा जाए। वेबसाइट बनाने से पहले जिन प्रमुख पूर्वधारणाओं पर विचार करना चाहिए, वे निम्नलिखित हैं—

1. **उद्देश्य और लक्ष्य** — वेबसाइट बनाने से पहले उसके उद्देश्य और लक्ष्य को स्पष्ट रूप से समझना आवश्यक है। क्या आप एक ई-कॉमर्स साइट, ब्लॉग, पोर्टफोलियो या कॉर्पोरेट वेबसाइट बना रहे हैं? वेबसाइट के प्राथमिक उद्देश्यों को परिभाषित करें।
2. **लक्षित प्रयोक्ता** — अपने लक्षित प्रयोक्ताओं और उनकी आवश्यकताओं की पहचान करें। उनकी जनसांख्यिकी (demographics), पसंद-नापसंद और व्यवहार को समझते हुए वेबसाइट की सामग्री और डिज़ाइन को उनके अनुकूल बनाएं।
3. **सामग्री कार्यनीति** — यह निर्धारित करें कि वेबसाइट पर किस प्रकार की सामग्री प्रदान की जाएगी। इसमें टेक्स्ट (text), चित्र, वीडियो तथा अन्य मल्टीमीडिया घटक शामिल हैं। यह भी योजना बनाएं कि सामग्री को कितनी बार अद्यतन (update) और संधारित (maintain) किया जाएगा।

4. **प्लेटफ़ॉर्म और प्रौद्योगिकी** — वेबसाइट के लिए उपयुक्त प्लेटफ़ॉर्म और तकनीकी संरचना का चयन करें। इसमें वर्डप्रेस (WordPress) जैसे कंटेंट मैनेजमेंट सिस्टम (CMS), कस्टम डेवलपमेंट या विक्स (Wix) और स्क्वेअरस्पेस (Squarespace) जैसे वेबसाइट बिल्डर में से चयन करना शामिल हो सकता है।
5. **प्रयोक्ता अनुभव (UX) और डिज़ाइन** — यह सुनिश्चित करें कि वेबसाइट प्रयोक्ता के अनुकूल और दृष्टिगत रूप से आकर्षक हो। नेविगेशन, लेआउट, रंग योजना और टाइपोग्राफी जैसे तत्वों पर ध्यान दें ताकि सकारात्मक प्रयोक्ता अनुभव प्राप्त हो।
6. **मोबाइल उत्तरदायित्व (Mobile Responsiveness)** — मोबाइल उपकरणों के बढ़ते उपयोग को ध्यान में रखते हुए यह मानकर चलें कि आपकी वेबसाइट विभिन्न स्क्रीन आकारों पर एक्सेस की जाएगी। सुनिश्चित करें कि साइट मोबाइल फ़ोन, टैबलेट और डेस्कटॉप पर अच्छी तरह कार्य करे।
7. **प्रदर्शन अनुकूलन (Performance Optimisation)** — वेबसाइट के लोड समय को कम करने हेतु प्रदर्शन का अनुकूलन करें। इसके अंतर्गत चित्रों का अनुकूलन, कंटेंट डिलीवरी नेटवर्क (CDN) का उपयोग, तथा कोड का मिनिफिकेशन शामिल है।
8. **होस्टिंग और डोमेन** — एक विश्वसनीय होस्टिंग प्रदाता और उपयुक्त डोमेन नाम का चयन करें। यह सुनिश्चित करें कि होस्टिंग योजना आपकी वेबसाइट की आवश्यकताओं और ट्रैफ़िक को संभाल सके।
9. **विपणन और प्रचार** — वेबसाइट को प्रचारित करने के लिए विपणन कार्यनीति तैयार करें। इसमें सोशल मीडिया, ईमेल मार्केटिंग, कंटेंट मार्केटिंग और अन्य प्रचार गतिविधियाँ शामिल हो सकती हैं।
10. **बजट और संसाधन** — वेबसाइट विकास, होस्टिंग, विपणन और निरंतर रखरखाव के लिए आवश्यक बजट का अनुमान लगाएँ। उसी अनुसार संसाधनों का आबंटन करें।
11. **बैकअप और पुनर्प्राप्ति** — वेबसाइट के डेटा को सुरक्षित रखने हेतु नियमित बैकअप और आपदा पुनर्प्राप्ति योजना लागू करें ताकि किसी भी अप्रत्याशित स्थिति में शीघ्र पुनः कार्यशीलता सुनिश्चित हो सके।

अभ्यास 2.2

- वेब डिज़ाइन में प्रयुक्त तीन सामान्य बॉर्डर्स की सूची बनाएं।
- वेबसाइट निर्माण हेतु तीन प्रमुख पूर्वधारणाओं की सूची बनाएं।

वेब डिज़ाइन प्रक्रिया के इंटरफेसेज़

वेब डिज़ाइन एक बहुआयामी प्रक्रिया है जिसमें डिज़ाइन स्वयं तथा डिज़ाइनरों, डेवलपर्स और क्लाइंट के बीच सहयोग की विभिन्न अवस्थाएँ और इंटरफेसेज़ शामिल होते हैं। वेब डिज़ाइन प्रक्रिया के इंटरफेसेज़ का संक्षिप्त अवलोकन निम्नलिखित है—

1. **क्लाइंट/डिज़ाइनर इंटरफेस** — प्रक्रिया की शुरुआत क्लाइंट और डिज़ाइनर के बीच बैठक से होती है, जिसमें परियोजना के लक्ष्य, आवश्यकताएँ और अपेक्षाएँ चर्चा में आती हैं। डिज़ाइनर परियोजना का प्रस्ताव तैयार करता है, जिसमें कार्यक्षेत्र, समय-सीमा और लागत को क्लाइंट की स्वीकृति हेतु प्रस्तुत किया जाता है।
2. **अनुसंधान और योजना** — क्लाइंट और डिज़ाइनर के बीच सतत संवाद के माध्यम से लक्षित प्रयोक्ता, प्रतिस्पर्धियों और परियोजना के उद्देश्यों के बारे में जानकारी एकत्र की जाती है। डिज़ाइनर अपनी टीम के साथ मिलकर बाज़ार अनुसंधान करता है और आवश्यक जानकारियाँ संकलित करता है।
3. **वायरफ़्रेमिंग और प्रोटोटाइपिंग** — डिज़ाइनर वेबसाइट के लेआउट और कार्यात्मकता के वायरफ़्रेम और प्रोटोटाइप तैयार करते हैं। क्लाइंट इन पर समीक्षा करते हैं और अपनी प्रतिक्रिया प्रदान करते हैं।
4. **डिज़ाइन विकास** — डिज़ाइनर रंग योजनाओं, टाइपोग्राफी, चित्रों और ग्राफ़िक्स सहित दृश्य तत्वों पर कार्य करते हैं। वे सुनिश्चित करते हैं कि डिज़ाइन तकनीकी रूप से व्यावहारिक हो और डेवलपमेंट हेतु अनुकूल हो, जिसके लिए वे डेवलपर्स के साथ सहयोग करते हैं।

5. **सामग्री निर्माण** — क्लाइंट टेक्स्ट, चित्रों और वीडियो जैसी सामग्री प्रदान करते हैं, जिसे डिज़ाइनर अपनी डिज़ाइन में सम्मिलित करते हैं। डिज़ाइनर कंटेंट राइटरों के साथ मिलकर यह सुनिश्चित करते हैं कि सामग्री डिज़ाइन में समाहित हो और वांछित संदेश को प्रभावी रूप से संप्रेषित करे।
6. **डिज़ाइन समीक्षा और संशोधन** — क्लाइंट डिज़ाइन की समीक्षा करते हैं और आवश्यक संशोधन हेतु प्रतिक्रिया प्रदान करते हैं। डिज़ाइनर यह प्रतिक्रिया डिज़ाइन टीम को संप्रेषित करते हैं ताकि आवश्यक परिवर्तन किए जा सकें।
7. **परीक्षण (Testing)** — डेवलपर्स वेबसाइट की कार्यात्मकता की जाँच करते हैं और यह सुनिश्चित करते हैं कि यह विभिन्न ब्राउज़रों और उपकरणों पर सही तरीके से कार्य करे। क्लाइंट भी प्रयोक्ता परीक्षण कर सकते हैं और उपयोगिता पर प्रतिक्रिया दे सकते हैं।
8. **क्लाइंट की स्वीकृति और परिनियोजन (Deployment)** — क्लाइंट अंतिम डिज़ाइन की समीक्षा करते हैं और लॉन्च की अनुमति देते हैं। डेवलपर्स वेबसाइट को वेब सर्वर पर परिनियोजित करते हैं, जिससे यह सार्वजनिक रूप से सुलभ हो जाती है।
9. **रख-रखाव (Maintenance)** — क्लाइंट की माँग के अनुसार डिज़ाइनर निरंतर सहायता और अद्यतन प्रदान कर सकते हैं। प्रयोक्ता वेबसाइट के साथ संवाद करते हैं और आगे सुधार हेतु प्रतिक्रिया देते हैं। डेवलपर्स क्लाइंट की प्रतिक्रिया के अनुसार वेबसाइट का रखरखाव और अद्यतन करते हैं ताकि यह सुरक्षित और अद्यतन बनी रहे।

वेब डिज़ाइन प्रक्रिया के सभी इंटरफेसेज़ के मध्य प्रभावी संवाद आवश्यक है ताकि अंतिम उत्पाद क्लाइंट के लक्ष्यों के अनुरूप हो और लक्षित प्रयोक्ताओं की आवश्यकताओं की पूर्ति कर सके। डिज़ाइनर, डेवलपर्स, सामग्री निर्माताओं और क्लाइंट के बीच सहयोग वेब डिज़ाइन परियोजना की सफलता के लिए अत्यंत आवश्यक है।

सारांश

- वेब डिज़ाइन और विकास एक बहुआयामी प्रक्रिया है, जिसमें योजना बनाना, विश्लेषण करना, डिज़ाइन करना, कोडिंग करना, परीक्षण करना, परिनियोजन (डिप्लॉयमेंट) और अनुरक्षण (मेंटेनेन्स) शामिल हैं।
- यह प्रक्रिया वेबसाइट के उद्देश्य, लक्षित प्रयोक्ता वर्ग और सामग्री संरचना को परिभाषित करने से शुरू होती है।
- सूचना संरचना (Information Architecture) में वेबसाइट की संरचना और प्रमुख पृष्ठों के लिए वायरफ़्रेम का दृश्य प्रतिनिधित्व बनाना शामिल होता है।
- वेब डिज़ाइन सौंदर्यशास्त्र, प्रयोक्ता अनुभव (UX), लेआउट, अनुक्रियाशीलता (responsiveness) और दृश्य तत्वों पर केंद्रित होता है।
- वेब विकास में फ्रंट-एंड और बैक-एंड विकास, डेटाबेस, सुरक्षा, परीक्षण, परिनियोजन और अनुरक्षण सम्मिलित होते हैं। वेब डिज़ाइन में सामग्री निर्माण और खोज इंजन के लिए उसका अनुकूलन अत्यंत महत्वपूर्ण होता है।
- परीक्षण से यह सुनिश्चित किया जाता है कि वेबसाइट सही तरीके से कार्य कर रही है, प्रदर्शन कर रही है और सुरक्षित है।
- परिनियोजन में डोमेन का चयन, होस्टिंग और DNS कॉन्फ़िगरेशन शामिल होता है।
- नियमित अनुरक्षण, अद्यतन और प्रयोक्ता प्रतिक्रियाएँ निरंतर सुधार के लिए आवश्यक होती हैं।
- वेब डिज़ाइन और वेब विकास दो भिन्न किंतु घनिष्ठ रूप से संबंधित विषय हैं, जिनमें सहयोग आवश्यक होता है।
- वेब डिज़ाइन के मूल तत्वों में लेआउट, रंग योजना, टाइपोग्राफ़ी, नेविगेशन, सामग्री, चित्रण, प्रयोक्ता अंतःक्रिया, स्थिरता, सुगम्यता और प्रदर्शन अनुकूलन शामिल हैं।
- वेब डिज़ाइन प्रक्रिया में इंटरफेस में शामिल हैं — ग्राहक/डिज़ाइनर, अनुसंधान/योजना, वायरफ़्रेम/प्रोटोटाइप, डिज़ाइन विकास, सामग्री निर्माण, डिज़ाइन समीक्षा/संशोधन, परीक्षण, ग्राहक अनुमोदन/परिनियोजन और अनुरक्षण।
- इन सभी इंटरफेस के बीच प्रभावी संप्रेषण और सहयोग, वेब डिज़ाइन परियोजनाओं की सफलता के लिए अत्यंत आवश्यक हैं।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. वेब डिज़ाइन और विकास की योजना और विश्लेषण चरण का उद्देश्य क्या है?
   क. दृश्य तत्वों का निर्माण करना
   ख. वेबसाइट की कोडिंग करना
   ग. वेबसाइट के लक्ष्य और लक्षित दर्शकों का निर्धारण करना
   घ. वेबसाइट की सामग्री का प्रबंधन करना

2. वेब विकास में सामान्यतः कौन-सी फ्रंट-एंड तकनीकों का उपयोग किया जाता है?
   क. PHP
   ख. पायथन
   ग. HTML, CSS, JavaScript
   घ. रूबी ऑन रेल्स

3. वेब डिज़ाइन का मुख्य उद्देश्य क्या है?
   क. सर्वर-साइड लॉजिक का निर्माण
   ख. प्रयोक्ता अनुभव को बेहतर बनाना
   ग. डेटाबेस का प्रबंधन करना
   घ. वेब ब्राउज़र के लिए कोड लिखना

4. वेब डिज़ाइन में UX का पूर्ण रूप क्या है?
   क. यूज़र एक्सपीरियंस
   ख. यूज़र एक्सटेंशन
   ग. यूज़र एक्सएमएल
   घ. यूनिवर्सल एक्सपीरियंस

5. अनुक्रियाशील डिज़ाइन (Responsive Design) क्या है?
   क. कई रंगों के साथ डिज़ाइन बनाना
   ख. मोबाइल उपकरणों के लिए डिज़ाइन बनाना
   ग. छवियों का अनुकूलन करना
   घ. डेटाबेस लॉजिक का निर्माण करना

6. कौन-सी बाध्यता विकलांग व्यक्तियों के लिए वेबसाइटों को उपयोगी बनाने से संबंधित है?
   क. तकनीकी बाध्यताएँ
   ख. सामग्री बाध्यताएँ
   ग. सुगम्यता (Accessibility) बाध्यताएँ
   घ. ब्रांड बाध्यताएँ

7. प्रभावशाली वेब डिज़ाइन लेआउट बनाने के लिए कौन-सा तत्व (एलिमेंट) महत्वपूर्ण है?
   क. ग्रिड संरचना

ख. ब्रांड रंग
ग. मोबाइल ऐप्स
घ. विपणन कार्यनीति

8. वेबसाइट बनाते समय निम्न में से कौन-सी प्रमुख पूर्वधारणा है?
   क. लक्षित दर्शकों को समझने की आवश्यकता नहीं है
   ख. बजट महत्वपूर्ण नहीं है
   ग. मोबाइल अनुक्रियाशीलता आवश्यक नहीं है
   घ. स्पष्ट उद्देश्य और लक्ष्य परिभाषित होने चाहिए

9. ग्राहक और डिज़ाइनर के बीच किस इंटरफेस में परियोजना के लक्ष्यों पर चर्चा होती है?
   क. ग्राहक/डिज़ाइनर इंटरफेस
   ख. अनुसंधान और योजना इंटरफेस
   ग. परीक्षण इंटरफेस
   घ. अनुरक्षण इंटरफेस

10. वेब डिज़ाइन में वायरफ़्रेम और प्रोटोटाइप का मुख्य उद्देश्य क्या है?
   क. अंतिम रूप से तैयार वेबसाइट का शुभारंभ
   ख. ग्राहक बैठकें
   ग. लेआउट और कार्यप्रणाली की योजना बनाना
   घ. विपणन कार्यनीति

ख. रिक्त स्थान भरें

1. वेब विकास में, _____ डेवलपर HTML, CSS और JavaScript का उपयोग करके प्रयोक्ता इंटरफ़ेस पर कार्य करते हैं।
2. सामग्री निर्माण में टेक्स्ट, चित्र, वीडियो और अन्य मीडिया को _____ में जोड़ना शामिल है।
3. प्रभावशाली वेब डिज़ाइन को प्रयोक्ता की बढ़ती अपेक्षाओं और _____ प्रवृत्तियों के अनुसार निरंतर आँका और अद्यतन किया जाना चाहिए।
4. वेब डिज़ाइन एक बहुआयामी प्रक्रिया है जिसमें विभिन्न _____ और इंटरफेस शामिल होते हैं।
5. वेब डिज़ाइन वेबसाइट या वेब एप्लिकेशन के दृश्य और _____ बनाने की प्रक्रिया को संदर्भित करता है।
6. वेब डेवलपर सूचनाओं को प्रबंधित करने और _____ के लिए डेटाबेस के साथ कार्य करते हैं।
7. कस्टम आइकन, लोगो और अन्य दृश्य संसाधनों के लिए अक्सर _____ के लिए ग्राफ़िक डिज़ाइन कौशल की आवश्यकता होती है।
8. प्रभावशाली वेब डिज़ाइन _____ नहीं होता।
9. डिज़ाइनरों को यह सुनिश्चित करना चाहिए कि वेबसाइटें सही तरीके से कार्य करें और विभिन्न वेब_____ पर समान रूप से दिखाई दें।
10. _____ में समग्र लेआउट में ग्रिड संरचना, अनुक्रियाशील डिज़ाइन और सफेद स्थान का प्रभावी उपयोग शामिल होता है।

ग. सत्य या असत्य

1. वेब डिज़ाइन का मुख्य उद्देश्य सर्वर-साइड लॉजिक बनाना होता है।
2. वेब डेवलपर वेबसाइट की कार्यप्रणाली विकसित करने के लिए प्रोग्रामिंग भाषाओं का उपयोग करते हैं।

3. वेब डिज़ाइनर प्रयोक्ता अनुभव (UX) को विचार में नहीं लिया जाता है।
4. अनुक्रियाशील डिज़ाइन में सुनिश्चित किया जाता है कि वेबसाइटें सभी स्क्रीन आकारों पर समान दिखें।
5. डिज़ाइनरों को बजट प्रतिबंधों के अंतर्गत कार्य करना होता है, जो डिज़ाइन तत्वों को प्रभावित कर सकता है।
6. वेब डिज़ाइन में प्रयोज्यता बाध्यताएँ प्रयोक्ता अनुकूल इंटरफेस बनाने से संबंधित होती हैं।
7. डिज़ाइन की गहराई और परीक्षण पर समय की बाध्यताओं का कोई प्रभाव नहीं पड़ता।
8. वायरफ्रेम और प्रोटोटाइप का उपयोग विपणन उद्देश्यों के लिए किया जाता है।
9. सामग्री निर्माण में केवल वेबसाइट पर टेक्स्ट जोड़ना शामिल होता है।
10. ग्राहक और डिज़ाइनर इंटरफेस वह स्थान होता है जहाँ परियोजना की शुरुआत में लक्ष्यों की चर्चा होती है।

घ. लघु उत्तर प्रश्न

1. वेब डिज़ाइन और विकास प्रक्रिया में कौन-कौन से मुख्य चरण शामिल होते हैं?
2. वेब डिज़ाइन में योजना और विश्लेषण चरण का उद्देश्य स्पष्ट कीजिए।
3. किसी वेबसाइट के लिए लक्षित दर्शक निर्धारण का क्या महत्व है?
4. वेब विकास में फ्रंट-एंड और बैक-एंड विकास के बीच क्या अंतर है?
5. अनुक्रियाशील डिज़ाइन किसी वेबसाइट की सफलता में कैसे योगदान देता है?
6. वेब डिज़ाइन में सामग्री निर्माण की भूमिका क्या है और यह क्यों आवश्यक है?
7. वेब डिज़ाइन में सामान्य बाध्यताएँ कौन-सी होती हैं और वे डिज़ाइन प्रक्रिया को कैसे प्रभावित करती हैं?
8. वेब डिज़ाइन में जैसे रंग योजना और टाइपोग्राफ़ी की स्थिरता बनाए रखना क्यों आवश्यक है?
9. वेब डिज़ाइन में सुगम्यता मानकों का पालन करना क्यों महत्वपूर्ण है?
10. वेबसाइट बनाते समय किन प्रमुख पूर्वधारणाओं को ध्यान में रखा जाना चाहिए?

सत्र 3
वेब डिज़ाइन और विकास उपकरण (टूल्स)

आइए मिलते हैं आरती से, जो एक जिज्ञासु बच्ची है और इंटरनेट का उपयोग बहुत पसंद करती है। वह जानना चाहती थी कि वेबसाइटें कैसे बनाई जाती हैं, इसलिए उसने श्री अजय की कक्षा में भाग लिया। आरती ने कुछ विशेष उपकरणों के बारे में जाना—

टेक्स्ट एडिटर — यह एक विशेष नोटपैड की तरह होता है, जिसमें वेबसाइट से संबंधित सामग्री लिखी जाती है।
HTML — यह किसी वेब पेज की अवसंरचना (skeleton) होता है, जो यह बताती है कि पेज कैसा दिखाई देगा।

CSS— यह एक पेंट ब्रश की तरह होता है, जो वेब पेज को आकर्षक बनाता है।

छवियाँ और ग्राफ़िक्स— ये वेब पेज को सुंदर बनाते हैं, जैसे चित्रों में रंग भरना।

वेब ब्राउज़र— यह एक जादुई खिड़की की तरह होता है, जिससे हम वेब पेज देख सकते हैं।

इन उपकरणों की मदद से आरती ने "द म्यूज़िक" नामक एक उपयोगी वेबसाइट बनाई, जिससे बच्चे सीख सकें। यह वेबसाइट सभी बच्चों को बहुत पसंद आई। आरती ने यह सीखा कि ये उपकरण उसके लिए किसी सुपरपावर जैसे हैं, जो उसे इंटरनेट पर शानदार चीज़ें बनाने और दूसरों को सिखाने में मदद करते हैं, जैसा कि **चित्र 3.1** में दर्शाया गया है।

चित्र 3.1— आरती वेबसाइट बनाते हुए

इस अध्याय में, आप वेब विकास मानकों, वेब विकास उपकरणों, वेबसाइटों के प्रकारों और वेबसाइटों के लिए कोडिंग एवं प्रोग्रामिंग के बारे में जानेंगे।

3.1 वेब विकास मानक

वेब विकास मानक और उपकरण उच्च गुणवत्ता वाली, विश्वसनीय और सभी के लिए सुलभ वेबसाइटें और वेब अनुप्रयोग (web applications) बनाने के लिए अत्यंत आवश्यक घटक हैं। ये मानक विभिन्न ब्राउज़रों और उपकरणों पर संगतता सुनिश्चित करते हैं और साथ ही प्रयोक्ता अनुभव तथा आपके प्रोजेक्ट्स की अनुरक्षणीयता (maintainability) में सुधार लाते हैं।

> **क्या आप जानते हैं?**
> वेब विकास मानक यह सुनिश्चित करते हैं कि वेब पेज विभिन्न उपकरणों, ब्राउज़र और प्लेटफ़ॉर्म पर सुसंगत (consistent), संगत (compatible) और रख-रखाव में आसान हों।

नीचे कुछ प्रमुख वेब विकास मानक और उपकरण दिए गए हैं—

1. **HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज)**— HTML वेब विकास की आधारशिला है। यह वेब पृष्ठों की संरचना और सामग्री को परिभाषित करता है। आधुनिक वेब विकास के लिए HTML5 मानकों से अद्यतन रहना अत्यंत आवश्यक है।
2. **CSS (कैस्केडिंग स्टाइल शीट्स)**— CSS का उपयोग वेब पृष्ठों को सजाने के लिए किया जाता है। CSS3 में उन्नत स्टाइलिंग विशेषताएँ हैं और यह प्रायः उत्तरदायी डिज़ाइन (responsive design) के लिए प्रयुक्त होता है।
3. **JavaScript**— JavaScript एक प्रोग्रामिंग भाषा है, जिसका उपयोग वेब पृष्ठों में इंटरैक्टिविटी और कार्यात्मकता जोड़ने के लिए किया जाता है। ECMAScript 6 (ES6) JavaScript का नवीनतम मानक है।
4. **सुगम्यता (Accessibility)**— वेब कंटेंट एक्सेसिबिलिटी गाइडलाइंस (WCAG) ऐसे मानक प्रदान करते हैं, जो दिव्यांग व्यक्तियों के लिए वेब सामग्री को सुलभ बनाते हैं। अपने वेब प्रोजेक्ट को सुलभ बनाना एक श्रेष्ठ अभ्यास ही नहीं, बल्कि कई मामलों में यह कानूनी आवश्यकता भी है।
5. **HTTP/HTTPS**— हाइपरटेक्स्ट ट्रांसफर प्रोटोकॉल (HTTP) और इसका सुरक्षित संस्करण HTTPS वेब संचार के मूल प्रोटोकॉल हैं। डेटा सुरक्षा और सर्च इंजन ऑप्टिमाइज़ेशन (SEO) के लिए HTTPS का उपयोग अत्यंत आवश्यक है।
6. **उत्तरदायी डिज़ाइन (Responsive Design)**— विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार अनुकूल वेबसाइट बनाना अत्यंत आवश्यक है। मीडिया क्वेरी और उत्तरदायी डिज़ाइन सिद्धांत अब मानक प्रथा बन गए हैं।
7. **सार्थक मार्कअप (Semantic Markup)**— <header>, <nav>, <section>, <article> जैसे अर्थपूर्ण HTML तत्वों का उपयोग आपकी सामग्री के अर्थ और संरचना को ब्राउज़र और डेवलपर दोनों के लिए स्पष्ट बनाता है।
8. **प्रगतिशील संवर्द्धन (Progressive Enhancement)**— ऐसा वेब अनुभव बनाना जो सभी प्रयोक्ताओं के लिए कार्य करे, चाहे उनका डिवाइस या ब्राउज़र कैसा भी हो। एक मूलभूत, कार्यात्मक संस्करण से शुरुआत करें और आधुनिक विशेषताओं के लिए उसे संवर्धित करें।

3.2 वेब विकास उपकरण

1. **टेक्स्ट एडिटर और आईडीई (IDE)** — वेब विकास के लिए लोकप्रिय टेक्स्ट एडिटर में Visual Studio Code, Sublime Text और Atom शामिल हैं। WebStorm जैसे एकीकृत विकास परिवेश (IDE) का उपयोग भी कई डेवलपर्स द्वारा किया जाता है।
2. **संस्करण नियंत्रण (Version Control)** — Git जैसे टूल और GitHub या GitLab जैसे प्लेटफॉर्म डेवलपर्स को कोड परिवर्तनों को ट्रैक करने, सहयोग करने और कोड संस्करणों को प्रभावी रूप से प्रबंधित करने में सहायता करते हैं।
3. **पैकेज प्रबंधक (Package Managers)**— npm (JavaScript के लिए) और Composer (PHP के लिए) जैसे पैकेज प्रबंधक वेब परियोजनाओं में निर्भरता (dependencies) को स्थापित करने और प्रबंधित करने में उपयोग होते हैं।
4. **टास्क रनर और बिल्ड टूल्स (Task Runner and Build Tools)** — Gulp, Grunt और Webpack जैसे उपकरण संपीड़न (minification), संकलन (compilation) और अनुकूलन (optimization) जैसे कार्यों को स्वचालित करके विकास प्रक्रिया को सुचारू बनाते हैं।
5. **ब्राउज़र और डेवलपर टूल्स** — आधुनिक ब्राउज़र जैसे Chrome, Firefox और Edge डेवलपर टूल्स के साथ आते हैं, जो डिबगिंग, प्रोफाइलिंग और अनुकूलन में सहायता करते हैं।
6. **फ्रंट एंड फ्रेमवर्क और लाइब्रेरी** — React, Angular और Vue.js जैसे लोकप्रिय फ्रेमवर्क और लाइब्रेरी जटिल वेब अनुप्रयोगों के विकास को सरल बनाते हैं।
7. **बैकएंड फ्रेमवर्क** — Express (Node.js), Django (Python), और Ruby on Rails (Ruby) जैसे फ्रेमवर्क सर्वर-साइड विकास के लिए संरचना और सहायक सुविधाएँ प्रदान करते हैं।
8. **सामग्री प्रबंधन प्रणाली (CMS)**— WordPress, Drupal और Joomla जैसे CMS प्लेटफ़ॉर्म वेब सामग्री को प्रबंधित और प्रकाशित करने के लिए पूर्वनिर्मित समाधान प्रदान करते हैं।

9. **परीक्षण और डिबगिंग उपकरण** — Jest (JavaScript परीक्षण के लिए), Selenium (ब्राउज़र स्वचालन के लिए), और Postman (API परीक्षण के लिए) जैसे टूल्स आपके वेब अनुप्रयोगों की विश्वसनीयता सुनिश्चित करते हैं।
10. **प्रदर्शन अनुकूलन उपकरण (Performance Optimisation Tools)**— Google PageSpeed Insights और Lighthouse जैसे टूल्स आपकी वेबसाइटों के प्रदर्शन का विश्लेषण और अनुकूलन करने में मदद करते हैं।
11. **सुरक्षा उपकरण (Security Tools)**— OWASP ZAP और अन्य सुरक्षा स्कैनर वेब अनुप्रयोगों में सुरक्षा कमजोरियों की पहचान और समाधान में सहायता करते हैं।
12. **होस्टिंग और परिनियोजन सेवाएँ (Hosting and Deployment Services)**— AWS, Heroku और Netlify जैसी सेवाएँ होस्टिंग और परिनियोजन के समाधान प्रदान करती हैं, जिससे वेब अनुप्रयोगों को प्रकाशित करना आसान हो जाता है।

3.3 वेबसाइट के प्रकार

वेबसाइटों को उनकी कार्यप्रणाली और प्रयोक्ताओं को सामग्री प्रदान करने के तरीके के आधार पर मुख्य रूप से दो श्रेणियों में विभाजित किया जा सकता है— **स्थैतिक वेबसाइट (Static Websites)** और **गतिशील वेबसाइट (Dynamic Websites)**।

3.3.1 स्थैतिक वेबसाइट (Static Websites)

सामग्री — स्थैतिक वेबसाइटों में ऐसे वेब पृष्ठ होते हैं जिनकी सामग्री स्थिर होती है और जब तक किसी वेब डेवलपर या प्रशासक (Administrator) द्वारा मैन्युअल रूप से संपादित न की जाए, तब तक उसमें कोई परिवर्तन नहीं होता।

प्रौद्योगिकियाँ — ये प्रायः HTML, CSS और कुछ हद तक बुनियादी इंटरएक्टिविटी (अंतःक्रियाशीलता) के लिए JavaScript का उपयोग करके बनाई जाती हैं।

विशेषताएँ —

- **स्थिर सामग्री** — स्थैतिक वेबसाइटों की सामग्री तब तक अपरिवर्तित रहती है जब तक कोई वेब विकास का जानकार व्यक्ति उसे अद्यतन नहीं करता।
- **तेज़ लोडिंग** — ये वेबसाइटें सामान्यतः तेज़ी से लोड होती हैं क्योंकि सर्वर किसी प्रोसेसिंग के बिना प्रयोक्ताओं को पूर्व-निर्मित HTML पृष्ठ सीधे प्रदान करता है।
- **सरल होस्टिंग** — इन्हें साधारण वेब सर्वरों पर होस्ट किया जा सकता है; इसके लिए सर्वर-साइड स्क्रिप्टिंग या डेटाबेस की आवश्यकता नहीं होती।
- **सीमित अंतःक्रियाशीलता** — इन वेबसाइटों को प्रयोक्ता के साथ व्यापक अंतःक्रिया या वैयक्तिकृत सामग्री प्रदाय के लिए नहीं बनाया गया होता।
- **उदाहरण उपयोग** — व्यक्तिगत ब्लॉग, लघु व्यावसाय वेबसाइटें, पोर्टफोलियो और ऐसी जानकारी संबंधी वेबसाइटें जिनकी सामग्री बार-बार नहीं बदलती।

क्या आप जानते हैं?

स्थैतिक वेबसाइट वह साइट होती है जिसमें प्रयोक्ता की भागीदारी न्यूनतम या नहीं के बराबर होती है, और इसका डिज़ाइन सामान्यतः सभी प्लेटफार्मों पर समान रहता है।

गतिशील वेबसाइट (Dynamic Websites)

सामग्री — गतिशील वेबसाइटें तात्कालिक रूप से सामग्री उत्पन्न करती हैं, अक्सर डेटाबेस या अन्य स्रोतों से डेटा लेकर। यह प्रणाली वास्तविक समय में अद्यतन और प्रयोक्ता के साथ अंतःक्रिया की अनुमति देती है।

प्रौद्योगिकियाँ — गतिशील वेबसाइटें सर्वर-साइड स्क्रिप्टिंग भाषाओं (जैसे PHP, Python, Ruby) तथा क्लाइंट-साइड तकनीकों (HTML, CSS, JavaScript) के संयोजन का उपयोग करती हैं।

विशेषताएँ —

- **डेटा-आधारित सामग्री** — सामग्री प्रयोक्ता इनपुट, डेटाबेस क्वेरी या बाह्य API के आधार पर गतिशील रूप से उत्पन्न होती है।
- **प्रयोक्ता अंतःक्रिया** — प्रयोक्ता फॉर्म, लॉगिन, टिप्पणियाँ और अन्य सुविधाओं के माध्यम से वेबसाइट से अंतःक्रिया कर सकते हैं।
- **वैयक्तिकरण (Personalization)** — प्रयोक्ता की प्राथमिकताओं या व्यवहार के अनुसार सामग्री को अनुकूलित किया जा सकता है।
- **सामग्री प्रबंधन प्रणाली (CMS)** — कई गतिशील वेबसाइटें वर्डप्रेस (WordPress), जूमला (Joomla), या ड्रुपल (Drupal) जैसी CMS का उपयोग करती हैं जिससे सामग्री का निर्माण और प्रबंधन सुगम हो जाता है।
- **उदाहरण उपयोग के मामले** — ई–कॉमर्स वेबसाइटें, सोशल मीडिया प्लेटफ़ॉर्म, ऑनलाइन मंच (forums), समाचार वेबसाइटें, तथा वे सभी वेबसाइटें जहाँ सामग्री बार-बार बदलती है या प्रयोक्ता की पसंद के अनुसार अनुकूलित की जाती है।

क्या आप जानते हैं?

डायनामिक (गतिशील) वेबसाइट वह वेबसाइट होती है जो प्रयोक्ताओं के साथ उनकी क्रिया के अनुसार बदलती रहती है।

ध्यान दें कि कई आधुनिक वेबसाइटों में स्थैतिक (Static) और डायनामिक (Dynamic) सामग्री दोनों के तत्व (एलिमेंट) सम्मिलित होते हैं। उदाहरण के लिए, किसी वेबसाइट में उसके मूल पृष्ठों (जैसे होमपेज और अबाउट पेज) के लिए स्थैतिक पृष्ठ हो सकते हैं, जबकि प्रयोक्ता एकाउंट्स, उत्पाद सूचीकरण या वास्तविक समय अधिसूचनाओं के लिए डायनामिक तत्व (एलिमेंट) शामिल हो सकते हैं।

अंततः, किसी परियोजना के विशेष उद्देश्यों और आवश्यकताओं के अनुसार स्थैतिक और डायनामिक वेबसाइट के बीच चयन किया जाता है। स्थैतिक वेबसाइटें सरल और रख-रखाव में आसान होती हैं, लेकिन इनमें इंटरैक्टिविटी की कमी होती है, जबकि डायनामिक वेबसाइटें अधिक कार्यात्मकता और लचीलापन प्रदान करती हैं, परंतु इनके लिए अधिक जटिल विकास और होस्टिंग अवसंरचना की आवश्यकता होती है।

3.3.3 स्थैतिक और डायनामिक वेबसाइटों के बीच अंतर

स्थैतिक और डायनामिक वेबसाइटें दो भिन्न प्रकार की वेबसाइटें होती हैं जो अलग-अलग उद्देश्यों की पूर्ति करती हैं और जिनकी विशेषताएँ भी अलग होती हैं।

पहलू	स्थैतिक वेबसाइटें	डायनामिक वेबसाइटें
सामग्री	निश्चित सामग्री / शायद ही कभी बदलती है।	सामग्री गतिशील रूप से बदल सकती है।
डिज़ाइन	सभी पृष्ठों में एकसमान डिज़ाइन और विन्यास।	डिज़ाइन और विन्यास पृष्ठ या प्रयोक्ता के अनुसार बदल सकते हैं।

पहलू	स्थैतिक वेबसाइटें	डायनामिक वेबसाइटें
इंटरैक्टिविटी	सीमित इंटरैक्टिविटी और प्रयोक्ता सहभागिता।	अत्यधिक इंटरैक्टिव और प्रयोक्ता क्रियाओं के प्रति प्रतिक्रियाशील।
डेटा	डेटा तब तक स्थिर रहता है जब तक मैन्युअल रूप से अद्यतन न किया जाए।	डेटा को डेटाबेस या बाह्य स्रोतों से प्राप्त किया जा सकता है।
रख-रखाव	कम अद्यतन की आवश्यकता के साथ रख-रखाव आसान।	नियमित रख-रखाव और अद्यतनों की आवश्यकता होती है।
विकास	स्थैतिक फ़ाइलों के आधार पर विकसित करना सरल।	सर्वर-साइड स्क्रिप्टिंग के साथ विकास अधिक जटिल।

तालिका 3.1 स्थैतिक और डायनामिक वेबसाइटों के बीच अंतर

स्थैतिक और डायनामिक वेबसाइट के बीच चयन आपकी विशेष आवश्यकताओं पर निर्भर करता है। स्थैतिक वेबसाइटें सरल और कम रख-रखाव वाली साइटों के लिए उपयुक्त होती हैं, जबकि डायनामिक वेबसाइटें जटिल, इंटरैक्टिव और बार-बार अद्यतन होने वाले वेब अनुप्रयोगों के लिए बेहतर होती हैं।

3.4 वेबसाइट के लिए कोडिंग और प्रोग्रामिंग

वेबसाइट के लिए कोडिंग और प्रोग्रामिंग में उस अंतर्निहित कोड का निर्माण करना शामिल होता है जो वेबसाइट को कार्यशील बनाता है और सामग्री प्रदर्शित करता है। वेबसाइटें आमतौर पर HTML, CSS और JavaScript के संयोजन से बनाई जाती हैं, और डायनामिक कार्यात्मकता हेतु सर्वर-साइड प्रोग्रामिंग भाषाएँ तथा डेटाबेस भी शामिल किए जा सकते हैं।

3.4.1 फ्रंट-एंड विकास उपकरण

फ्रंट-एंड विकास वेब विकास का एक महत्वपूर्ण भाग है जो वेबसाइट या वेब एप्लिकेशन के प्रयोक्ता इंटरफ़ेस और प्रयोक्ता अनुभव के निर्माण पर केंद्रित होता है। HTML, CSS और JavaScript फ्रंट-एंड विकास की मूल तकनीकें हैं।

1. **HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज)** — HTML वेब विकास की नींव है। यह वेब पृष्ठों की संरचना और सामग्री को परिभाषित करता है। HTML5, इसका नवीनतम संस्करण, कई नए तत्वों और गुणों को प्रस्तुत करता है जो ऑडियो और वीडियो एम्बेडिंग, सेमेंटिक टैग्स और बेहतर फ़ॉर्म हैंडलिंग जैसी क्षमताओं को बढ़ाते हैं।
2. **CSS (कैस्केडिंग स्टाइल शीट्स)** — CSS का उपयोग वेब पृष्ठों के शैली निर्धारण के लिए किया जाता है। CSS वेब पृष्ठों की प्रस्तुति और विन्यास को नियंत्रित करता है। यह HTML तत्वों के लिए फ़ॉन्ट्स, रंग, मार्जिन, पेडिंग और पोजीशनिंग जैसी शैलियाँ परिभाषित करने की अनुमति देता है। CSS3 ने एनिमेशन, ट्रांजिशन और मीडिया क्वेरी जैसी उन्नत विशेषताएँ प्रदान कीं जो विभिन्न स्क्रीन आकारों और उपकरणों के लिए उत्तरदायी डिज़ाइन को संभव बनाती हैं।
3. **JavaScript** — JavaScript एक प्रोग्रामिंग भाषा है जिसका उपयोग वेब पृष्ठों में इंटरैक्टिविटी और कार्यात्मकता जोड़ने के लिए किया जाता है। यह HTML और CSS में परिवर्तन करने, प्रयोक्ता इनपुट को हैंडल करने और सर्वर से संवाद (Ajax के माध्यम से) करने में सक्षम है। आधुनिक JavaScript, React, Angular और Vue.js जैसे लाइब्रेरी और फ्रेमवर्क्स की सहायता से जटिल कार्यों को सरल बनाती है और सिंगल-पेज एप्लिकेशन (SPA) के निर्माण को सक्षम करती है।
4. **टेक्स्ट एडिटर्स और आईडीई (IDE)** — डेवलपर्स Visual Studio Code, Sublime Text जैसे टेक्स्ट एडिटर्स या WebStorm जैसे आईडीई का उपयोग HTML, CSS और JavaScript कोड लिखने और प्रबंधित करने के लिए करते हैं। इन उपकरणों में अक्सर सिंटैक्स हाइलाइटिंग, कोड कंप्लीशन और डिबगिंग जैसी विशेषताएँ होती हैं।

5. **वर्शन कंट्रोल सिस्टम (जैसे Git)** — Git जैसे वर्शन कंट्रोल सिस्टम डेवलपर्स को प्रोजेक्ट्स पर सहयोग करने, परिवर्तनों का ट्रैक रखने और कोड संस्करणों को प्रभावी तरीके से प्रबंधित करने में सहायता करते हैं। GitHub और GitLab जैसे लोकप्रिय प्लेटफ़ॉर्म रिपॉजिटरी को होस्ट करने के लिए प्रयुक्त होते हैं।
6. **पैकेज मैनेजर (जैसे npm और Yarn)** — पैकेज मैनेजर तृतीय-पक्ष लाइब्रेरी और फ्रेमवर्क्स को इंस्टॉल, अपडेट और प्रबंधित करने की प्रक्रिया को सरल बनाते हैं। JavaScript परियोजनाओं में सामान्यतः npm और Yarn प्रयुक्त होते हैं।
7. **बिल्ड टूल्स (जैसे Webpack और Gulp)** — बिल्ड टूल्स JavaScript और CSS फ़ाइलों के बंडलिंग और मिनिफिकेशन, छवियों के अनुकूलन और निर्भरताओं के प्रबंधन जैसे कार्यों को स्वचालित करते हैं। ये प्रदर्शन और अनुरक्षण को बेहतर बनाते हैं।
8. **ब्राउज़र डेवलपर टूल्स** — ब्राउज़र में अंतर्निर्मित डेवलपर टूल्स होते हैं जो HTML, CSS और JavaScript का निरीक्षण और रीयल टाइम में डिबगिंग की सुविधा देते हैं, जिससे समस्याओं को हल करना आसान होता है।
9. **उत्तरदायी डिज़ाइन और परीक्षण उपकरण** — उत्तरदायी डिज़ाइन सिमुलेटर और ब्राउज़र एक्सटेंशन जैसे उपकरण यह सुनिश्चित करने में सहायक होते हैं कि वेब पृष्ठ विभिन्न स्क्रीन आकारों और उपकरणों पर ठीक से कार्य करें और आकर्षक दिखें।
10. **प्रदर्शन अनुकूलन उपकरण** — Lighthouse और PageSpeed Insights जैसे उपकरण वेब पृष्ठ के प्रदर्शन का विश्लेषण करते हैं और तेज़ लोडिंग समय के लिए अनुकूलन सुझाव प्रदान करते हैं।

फ्रंट-एंड विकास एक गतिशील क्षेत्र है, और आधुनिक तथा प्रयोक्ता केंद्रित वेब अनुभवों के निर्माण के लिए नवीनतम तकनीकों, उपकरणों और सर्वोत्तम प्रथाओं से अद्यतन रहना आवश्यक है।

3.4.2 बैक-एंड विकास उपकरण (Back-end development tools)—

Python, PHP और SQL वेब विकास के बैक-एंड क्षेत्र में सामान्यतः उपयोग किए जाने वाले उपकरण और भाषाएँ हैं।

1. **Python**— Python एक बहुपरकीय (versatile), उच्च स्तरीय प्रोग्रामिंग भाषा है, जो अपनी सरलता और पठनीयता के लिए जानी जाती है। यह प्रायः वेब विकास में विभिन्न फ्रेमवर्क और लाइब्रेरी (libraries) जैसे Django, Flask और FastAPI के माध्यम से बैक-एंड अनुप्रयोगों (applications) को विकसित करने हेतु प्रयुक्त होती है। Python वेब अनुरोधों (web requests) को संभालने, डेटाबेस का प्रबंधन करने तथा व्यावसायिक तर्क (business logic) को लागू करने में उत्कृष्ट समर्थन प्रदान करती है। यह अपने मजबूत सामुदायिक सहयोग और व्यापक पैकेज संग्रह के कारण बैक-एंड विकास के लिए एक लोकप्रिय विकल्प मानी जाती है।
2. **PHP**— PHP (हाइपरटेक्स्ट प्री-प्रोसेसर) एक सर्वर-साइड स्क्रिप्टिंग भाषा है, जिसे विशेष रूप से वेब विकास के लिए डिज़ाइन किया गया है। यह गतिशील वेब पृष्ठों और वेब अनुप्रयोगों को बनाने के लिए सबसे अधिक प्रयुक्त भाषाओं में से एक है। PHP प्रायः Laravel, Symfony और CodeIgniter जैसे लोकप्रिय वेब विकास फ्रेमवर्क के साथ प्रयोग की जाती है। इसमें डेटाबेस से कनेक्ट करने के लिए अंतर्निहित समर्थन (built-in support) उपलब्ध है, जिससे यह Apache या Nginx जैसे वेब सर्वर के साथ प्रयोग करने पर बैक-एंड विकास के लिए उपयुक्त विकल्प बनती है।
3. **SQL (स्ट्रक्चर्ड क्वेरी लैंग्वेज)**— SQL एक प्रोग्रामिंग भाषा नहीं, बल्कि एक डोमेन-विशिष्ट भाषा (domain-specific language) है, जिसका उपयोग रिलेशनल डेटाबेस का प्रबंधन और क्वेरी (query) करने हेतु किया जाता है। बैक-एंड विकास में SQL का उपयोग डेटाबेस बनाने, अपडेट करने और क्वेरी करने के लिए किया जाता है, जिसमें वेब अनुप्रयोगों के लिए डेटा स्टोर होता है। SQL का उपयोग करने वाले सामान्य रिलेशनल डेटाबेस प्रबंधन प्रणाली (RDBMS) हैं—MySQL, PostgreSQL, Oracle और Microsoft SQL Server। डेवलपर्स SQL क्वेरी का उपयोग करके डेटाबेस से संवाद करते हैं, डेटा प्राप्त करते हैं, रिकॉर्ड अपडेट करते हैं और विभिन्न डेटाबेस संबंधी क्रियाएँ करते हैं।

जब किसी वेब अनुप्रयोग के बैक-एंड का विकास किया जाता है, तो इन उपकरणों और भाषाओं के समन्वय से एक मजबूत और कार्यात्मक प्रणाली तैयार की जा सकती है। एप्लिकेशन के लॉजिक को संभालने के लिए Python या PHP का उपयोग किया जा सकता

है, जबकि डेटा को स्टोर और पुनः प्राप्त करने के लिए डेटाबेस प्रबंधन हेतु SQL का प्रयोग होता है। Python और PHP में चयन प्रायः परियोजना की आवश्यकताओं, टीम की विशेषज्ञता और डेवलपर की व्यक्तिगत प्राथमिकताओं पर निर्भर करता है।

अभ्यास 3.2

- बैक-एंड वेब विकास के क्षेत्र में सामान्यतः उपयोग किए जाने वाले उपकरणों और भाषाओं के नाम सूचीबद्ध करें।
- फ्रंट-एंड विकास के लिए सामान्यतः उपयोग की जाने वाली तकनीकों के नाम सूचीबद्ध करें।

सारांश (SUMMARY)

- वेब विकास मानक और उपकरण उच्च गुणवत्ता वाली, विश्वसनीय और सुलभ वेबसाइटों और वेब अनुप्रयोगों के निर्माण में अत्यंत महत्वपूर्ण भूमिका निभाते हैं।
- HTML, CSS, JavaScript और WCAG जैसे सुलभता मानक आधुनिक वेब विकास के मूल कॉलम हैं।
- फ्रंट-एंड उपकरणों में टेक्स्ट एडिटर (text editors), संस्करण नियंत्रण प्रणाली (version control systems) और प्रतिक्रियाशील डिज़ाइन (responsive design) शामिल हैं, जिनका उपयोग प्रयोक्ता इंटरफ़ेस बनाने में किया जाता है।
- बैक-एंड विकास के लिए Python, PHP और SQL जैसे उपकरणों का उपयोग सर्वर-साइड तर्क और डेटाबेस के प्रबंधन हेतु किया जाता है।
- स्थैतिक और गतिशील वेबसाइटों के उद्देश्य भिन्न होते हैं, और इनका चयन परियोजना के लक्ष्यों पर निर्भर करता है।
- वेबसाइट कोडिंग में HTML, CSS, JavaScript और बैक-एंड भाषाओं का उपयोग करके कार्यात्मक वेब अनुप्रयोगों का निर्माण किया जाता है।

अपनी प्रगति जाँचें (Check Your Progress)

क. बहुविकल्पीय प्रश्न (Multiple Choice Questions)

1. वेब विकास में HTML का मुख्य कार्य क्या है?
   (क) वेब पृष्ठों की शैली निर्धारण करना
   (ख) सर्वर-साइड तर्क बनाना
   (ग) वेब पृष्ठों की संरचना और सामग्री को परिभाषित करना
   (घ) वेब पृष्ठों में अन्तरक्रियाशीलता जोड़ना

2. निम्नलिखित में से कौन-सा CSS संस्करण प्रतिक्रियाशील डिज़ाइन के लिए सामान्यतः प्रयुक्त होता है?
   (क) CSS1
   (ख) CSS2
   (ग) CSS3
   (घ) CSS4

3. ECMAScript 6 (ES6) किस वेब विकास तकनीक से संबंधित है?
   (क) HTML
   (ख) CSS
   (ग) JavaScript
   (घ) HTTP

4. निम्न में से कौन-से दिशा-निर्देश विकलांग व्यक्तियों के लिए सुलभ वेब सामग्री बनाने के मानक प्रदान करते हैं?
   (क) HTTP/HTTPS
   (ख) WCAG
   (ग) HTML5
   (घ) CSS3

5. वेब विकास में HTTPS का उपयोग क्यों आवश्यक है?
   (क) यह वेब पृष्ठों की शैली में सुधार करता है।
   (ख) यह वेबसाइट की अन्तरक्रियाशीलता को बढ़ाता है।
   (ग) यह डेटा सुरक्षा और SEO के लिए अत्यंत महत्त्वपूर्ण है।
   (घ) यह सामग्री प्रबंधन को सरल बनाता है।
6. विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार अनुकूलित वेबसाइटें बनाने के लिए कौन-सी मानक पद्धति अपनाई जाती है?
   (क) सेमेंटिक मार्कअप
   (ख) प्रोग्रेसिव एन्हांसमेंट
   (ग) सुलभ डिज़ाइन
   (घ) गतिशील सामग्री निर्माण

7. वेब विकास में परिवर्तनों को ट्रैक करने, सहयोग करने और कोड संस्करण प्रबंधन के लिए कौन-सा उपकरण सामान्यतः उपयोग किया जाता है?
   (क) npm
   (ख) HTTP
   (ग) Git
   (घ) CSS

8. निम्न में से कौन-सा फ्रंट-एंड JavaScript लाइब्रेरी या फ्रेमवर्क है?
   (क) Git
   (ख) MySQL
   (ग) React
   (घ) PHP

9. वेब विकास में कंटेंट मैनेजमेंट सिस्टम (CMS) का मुख्य उद्देश्य क्या है?
   (क) डेटाबेस प्रबंधन
   (ख) प्रतिक्रियाशील डिज़ाइन बनाना
   (ग) सर्वर-साइड तर्क को संभालना
   (घ) सामग्री निर्माण और प्रबंधन को सरल बनाना

10. वेब एप्लीकेशन परीक्षण और स्वचालन (automation) के लिए सामान्यतः कौन-सा उपकरण उपयोग किया जाता है?
    (क) HTML
    (ख) CSS
    (ग) Selenium

(घ) HTTP

ख. रिक्त स्थान भरें (Fill in the Blanks)

1. HTML वेब विकास की नींव है, जो वेब पृष्ठों की _____ और सामग्री को परिभाषित करता है।
2. CSS3 में उन्नत शैली निर्धारण की विशेषताएँ होती हैं और यह सामान्यतः _____ डिज़ाइन के लिए प्रयुक्त होता है।
3. JavaScript एक प्रोग्रामिंग भाषा है, जिसका उपयोग वेब पृष्ठों में _____ और कार्यक्षमता जोड़ने हेतु किया जाता है।
4. वेब विकास में _____ और SEO के लिए HTTPS का उपयोग आवश्यक है।
5. प्रतिक्रियाशील डिज़ाइन में ऐसे _____ बनाए जाते हैं, जो मीडिया क्वेरीज़ और डिज़ाइन सिद्धांतों के माध्यम से विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार अनुकूलित होते हैं।
6. <header>, <nav>, <section>, <article> जैसे सेमेंटिक _____ तत्वों का उपयोग ब्राउज़र और डेवलपर्स दोनों के लिए सामग्री की संरचना को स्पष्ट करने में सहायक होता है।
7. HTML _____ विकास की नींव है।
8. ECMAScript 6 (ES6) _____ के लिए नवीनतम मानक है।
9. Python एक बहुपरकीय, _____ प्रोग्रामिंग भाषा है, जो अपनी सरलता और पठनीयता के लिए जानी जाती है।
10. _____ डेटाबेस बनाने, अद्यतन करने और क्वेरी करने के लिए आवश्यक है, जो वेब अनुप्रयोगों का डेटा स्टोर करता है।

ग. सत्य/असत्य (True/False)

1. CSS3 का उपयोग मुख्यतः स्थैतिक वेब पृष्ठों के निर्माण के लिए किया जाता है।
2. JavaScript एक प्रोग्रामिंग भाषा है, जो वेब पृष्ठों में अन्तरक्रियाशीलता और कार्यक्षमता जोड़ती है।
3. आधुनिक वेब विकास में प्रतिक्रियाशील डिज़ाइन सिद्धांतों को मानक अभ्यास नहीं माना जाता।
4. सेमेंटिक HTML तत्वों का उपयोग वेबसाइट की सुलभता पर कोई प्रभाव नहीं डालता।
5. प्रोग्रेसिव एन्हांसमेंट में ऐसे वेब अनुभव तैयार किए जाते हैं, जो सभी प्रयोक्ताओं के लिए उनके डिवाइस या ब्राउज़र की क्षमताओं की परवाह किए बिना सुलभ हों।
6. वेब विकास मानक और उपकरण वेबसाइटों और वेब अनुप्रयोगों की गुणवत्ता सुनिश्चित करने में न्यूनतम भूमिका निभाते हैं।
7. टेक्स्ट एडिटर और IDE वेब विकास में सामान्यतः उपयोग नहीं किए जाते।
8. Git जैसे संस्करण नियंत्रण प्रणाली का उपयोग मुख्यतः परियोजना प्रबंधन के लिए किया जाता है, न कि कोड परिवर्तनों को ट्रैक करने के लिए।
9. JavaScript एक बहुपरकीय प्रोग्रामिंग भाषा है, जो वेब पृष्ठों में अन्तरक्रियाशीलता और गतिशील व्यवहार जोड़ती है।
10. npm जैसे पैकेज प्रबंधक वेब परियोजनाओं में निर्भरताओं को स्थापित और प्रबंधित करने के लिए उपयोग किए जाते हैं।

घ. लघु प्रश्नोत्तर (Short Answer Questions)

1. वेब विकास में HTML का मुख्य उद्देश्य क्या है?
2. वेब डिज़ाइन और विकास में CSS की भूमिका स्पष्ट करें।
3. प्रतिक्रियाशील डिज़ाइन के कुछ प्रमुख सिद्धांत क्या हैं?
4. सेमेंटिक HTML तत्व (एलिमेंट) वेब सुलभता में कैसे योगदान करते हैं?
5. वेब सुरक्षा और SEO के लिए HTTPS का उपयोग क्यों आवश्यक है?
6. वेब परियोजनाओं के लिए कंटेंट मैनेजमेंट सिस्टम (CMS) का उपयोग करने के क्या लाभ हैं?
7. वेब विकास में प्रोग्रेसिव एन्हांसमेंट का क्या महत्व है?
8. वेब डेवलपर्स द्वारा उपयोग किए जाने वाले दो लोकप्रिय टेक्स्ट संपादकों के नाम बताइए।
9. Git जैसे संस्करण नियंत्रण प्रणाली वेब विकास परियोजनाओं को कैसे लाभ पहुँचाती हैं?
10. वेब विकास में npm जैसे पैकेज प्रबंधक का उद्देश्य क्या होता है?

सत्र 4

आईटी के एप्लीकेशन विकास मॉडल

आशुतोष से मिलिए, वह एक जिज्ञासु बालक है जिसे टैबलेट और स्मार्टफोन का उपयोग करना बहुत पसंद है। वह जानना चाहता था कि उसके उपकरणों में जो रोचक ऐप्स होते हैं, वे कैसे बनाए जाते हैं। इसी कारण उसने "आईटी के एप्लीकेशन विकास मॉडल" के बारे में सीखना शुरू किया। उसने तीन मॉडल के बारे में जाना —

वॉटरफॉल मॉडल — यह किसी नुस्खे को एक-एक चरण में पीछे लौटे बिना पालन करने जैसा है। यह सरल है लेकिन इसमें अधिक परिवर्तन की अनुमति नहीं होती।

एजाइल मॉडल — यह इमारत की तरह ईंटों से बनाना जैसा है, जिसमें आप चलते-चलते बदलाव कर सकते हैं। यह लचीला और मजेदार होता है।

इटरेटिव मॉडल — यह किसी चित्र को बनाते हुए धीरे-धीरे उसमें विवरण जोड़ने जैसा है। यह रचनात्मक होता है।

इन मॉडलों की सहायता से आशुतोष ने अपने मित्रों के लिए एक मनोरंजक ऐप बनाया और उन्हें यह बहुत पसंद आया। आशुतोष ने सीखा कि ऐप बनाना एक रोचक रोमांच की तरह होता है। वह विभिन्न मॉडलों का उपयोग करके अपने उपकरणों में मजेदार चीजें बना सकता है — जैसे खिलौनों से खेलते हुए और उन्हें अपने मित्रों के साथ साझा कर सकता है ताकि सभी मिलकर आनंद उठा सकें। जैसा कि चित्र 4.1 में दर्शाया गया है।

चित्र 4.1 — आशुतोष द्वारा ऐप बनाना

इस अध्याय में, आप आईटी एप्लीकेशन विकास, एप्लीकेशन विकास में एसडीएलसी (SDLC) मॉडलों के महत्व, सॉफ़्टवेयर विकास जीवन चक्र (Software Development Life Cycle — SDLC) मॉडलों तथा वॉटरफॉल, इटरेटिव और एजाइल मॉडल के बारे में समझेंगे।

4.1 आईटी एप्लीकेशन विकास

आईटी एप्लीकेशन विकास मॉडल, जिन्हें सॉफ़्टवेयर विकास कार्यप्रणाली या अवसंरचना (Framework) भी कहा जाता है, ऐसे संरचित दृष्टिकोण होते हैं जो सॉफ़्टवेयर एप्लीकेशनों को बनाने और प्रबंधित करने की प्रक्रिया का मार्गदर्शन करते हैं। ये मॉडल विकास टीमों को उनके कार्य को सुव्यवस्थित करने, सहयोग को बेहतर बनाने और उच्च गुणवत्ता वाले सॉफ़्टवेयर उत्पादों को प्रदान करने में सहायता करते हैं। कई स्थापित एप्लीकेशन विकास मॉडल उपलब्ध हैं, जिनमें से प्रत्येक के अपनी विशेष सिद्धांत, प्रक्रिया और कार्यप्रणालियाँ होती हैं।

4.1.1 एप्लीकेशन विकास में एसडीएलसी मॉडलों का महत्व

सॉफ़्टवेयर विकास जीवन चक्र (SDLC) मॉडल, एप्लीकेशन विकास में निम्नलिखित प्रमुख कारणों से आवश्यक होते हैं—

1. **संरचित दृष्टिकोण** — एसडीएलसी मॉडल सॉफ़्टवेयर विकास के लिए एक संरचित और प्रणालीगत दृष्टिकोण प्रदान करते हैं। ये संपूर्ण विकास प्रक्रिया को चरणों और गतिविधियों में विभाजित करते हैं, जिससे प्रत्येक चरण स्पष्ट रूप से परिभाषित और प्रलेखित होता है। यह संरचना परियोजना प्रबंधन और नियंत्रण को बेहतर बनाती है।
2. **जोखिम प्रबंधन** — एसडीएलसी मॉडल विकास प्रक्रिया के दौरान जोखिमों की पहचान और प्रबंधन में सहायक होते हैं। पूर्वनिर्धारित चरणों का पालन कर और प्रत्येक चरण में जोखिम आकलन को सम्मिलित कर, टीमें समस्याओं का पूर्वानुमान कर सकती हैं और उन्हें समय रहते हल कर सकती हैं।
3. **गुणवत्ता आश्वासन** — एसडीएलसी मॉडल संपूर्ण विकास जीवन चक्र में गुणवत्ता के महत्व पर बल देते हैं। प्रत्येक चरण में गुणवत्ता जांच और परीक्षण सम्मिलित होते हैं, जिससे यह सुनिश्चित किया जा सके कि अंतिम उत्पाद मानकों पर खरा उतरे और उसमें कोई गंभीर त्रुटि न हो।
4. **स्पष्ट संवाद** — एसडीएलसी मॉडल टीम के सदस्यों, हितधारकों और ग्राहकों के बीच स्पष्ट संवाद को संभव बनाते हैं। स्पष्ट चरणों और उपलब्धियों के साथ, परियोजना में संलग्न सभी व्यक्तियों को प्रगति की सामान्य समझ होती है और वे अपेक्षाएँ तथा आवश्यकताएँ प्रभावी तरीके से साझा कर सकते हैं।
5. **पूर्वानुमानित समय-सीमा और बजट** — एसडीएलसी मॉडल परियोजना की समय-सीमा और बजट के बेहतर अनुमान में सहायक होते हैं। परियोजना को छोटे-छोटे चरणों और कार्यों में बाँटने से आवश्यक संसाधनों और समय का सटीक अनुमान लगाया जा सकता है, जिससे यथार्थपरक योजना बनाना संभव होता है।
6. **अनुकूलनशीलता (Scalability)** — एसडीएलसी मॉडल विभिन्न परियोजनाओं की विशिष्ट आवश्यकताओं के अनुसार अनुकूलित किए जा सकते हैं। चाहे आप एक छोटे एप्लीकेशन पर कार्य कर रहे हों या एक बड़े एंटरप्राइज़ स्तर के सिस्टम पर, आप उपयुक्त मॉडल चुन सकते हैं या परियोजना के आकार और जटिलता के अनुसार उसे अनुकूलित कर सकते हैं।
7. **प्रलेखन और अनुरेखणीयता (Traceability)** — एसडीएलसी मॉडल प्रत्येक चरण में विस्तृत प्रलेखन को प्रोत्साहित करते हैं। यह दस्तावेज़ीकरण भविष्य में संदर्भ, रखरखाव और समस्या समाधान के लिए अमूल्य होता है। यह अनुपालन और ऑडिट प्रयोजनों के लिए विकास प्रक्रिया को अच्छी तरह से दर्ज करने में भी सहायता करता है।
8. **ग्राहक संतुष्टि** — एसडीएलसी मॉडल संपूर्ण विकास प्रक्रिया में ग्राहक की भागीदारी और प्रतिक्रिया को प्राथमिकता देते हैं। यह ग्राहक-केंद्रित दृष्टिकोण सुनिश्चित करता है कि अंतिम उत्पाद ग्राहक की दृष्टि और आवश्यकताओं के अनुरूप हो, जिससे अधिक संतुष्टि प्राप्त होती है।
9. **परिवर्तन प्रबंधन** — एसडीएलसी मॉडल परिवर्तन और कार्य-क्षेत्र वृद्धि (scope creep) को प्रभावी रूप से सँभालने के लिए तंत्र प्रदान करते हैं। सभी परिवर्तन मूल्यांकन, प्रलेखित और नियंत्रित तरीके से लागू किए जाते हैं ताकि कार्य में विघ्न कम हो और परियोजना का ध्यान केंद्रित रहे।
10. **निरंतर सुधार** — एसडीएलसी मॉडल निरंतर सुधार की संस्कृति को प्रोत्साहित करते हैं। प्रत्येक परियोजना के बाद, टीमें यह मूल्यांकन कर सकती हैं कि क्या अच्छा रहा और क्या नहीं और इन जानकारियों का उपयोग भविष्य की परियोजनाओं की प्रक्रिया को परिष्कृत करने के लिए कर सकती हैं।

सारांशत:, एसडीएलसी मॉडल एप्लीकेशन विकास में एक संरचित अवसंरचना प्रदान कर योजना, क्रियान्वयन और प्रबंधन को सक्षम बनाते हैं। ये यह सुनिश्चित करते हैं कि विकास प्रयास दक्षतापूर्ण, नियंत्रित और प्रयोक्ता की अपेक्षाओं के अनुरूप उच्च गुणवत्ता युक्त सॉफ़्टवेयर उत्पादों में परिणत हों। किसी विशिष्ट परियोजना के लिए उपयुक्त एसडीएलसी मॉडल का चयन उसकी सफलता की कुंजी है।

4.2 सॉफ़्टवेयर विकास जीवन चक्र (SDLC) मॉडल

सॉफ़्टवेयर विकास जीवन चक्र (Software Development Life Cycle – SDLC) मॉडल ऐसे रूपरेखा अवसंरचना (frameworks) होते हैं जो सॉफ़्टवेयर अनुप्रयोगों या प्रणालियों के विकास की प्रक्रिया का मार्गदर्शन करते हैं। ये मॉडल विकास प्रक्रिया में सम्मिलित चरणों, कार्यों और गतिविधियों को परिभाषित करते हैं तथा सॉफ़्टवेयर परियोजनाओं के प्रबंधन और निष्पादन हेतु एक संरचित दृष्टिकोण प्रदान करते हैं। अनेक प्रकार के SDLC मॉडल होते हैं, जिनमें प्रत्येक के अपने सिद्धांत, लाभ और सीमाएँ होती हैं। नीचे कुछ सर्वाधिक प्रचलित SDLC मॉडल दिए गए हैं—

1. **वॉटरफॉल मॉडल (Waterfall Model)** — यह सॉफ़्टवेयर विकास का एक रैखिक (linear) एवं अनुक्रमिक (sequential) दृष्टिकोण है। यह परियोजना को आवश्यकताओं का संकलन, डिज़ाइन, कार्यान्वयन, परीक्षण, परिनियोजन (deployment) और अनुरक्षण जैसे पृथक् चरणों में विभाजित करता है। प्रत्येक चरण के पूर्ण होने के बाद ही अगला चरण आरंभ किया जा सकता है तथा एक बार पूरा हो जाने के बाद उसमें परिवर्तन करना कठिन होता है। यह मॉडल उन परियोजनाओं के लिए उपयुक्त होता है जिनमें आवश्यकताएँ स्पष्ट और स्थिर होती हैं।
2. **प्रोटोटाइप मॉडल (Prototype Model)** — यह मॉडल ग्राहक की आवश्यकताओं को बेहतर तरीके से समझने, विचारों का परीक्षण करने और अंतिम उत्पाद के विकास से पूर्व प्रणाली को परिष्कृत करने के लिए प्रोटोटाइप तैयार करने पर बल देता है। पारंपरिक SDLC मॉडल की रैखिक प्रकृति के विपरीत, प्रोटोटाइप मॉडल पुनरावृत्तिमूलक (iterative) और लचीला होता है, जो प्रतिक्रिया के अनुसार बदलाव करने की अनुमति देता है।
3. **एजाइल मॉडल (Agile Model)** — यह पुनरावृत्तिमूलक और लचीला दृष्टिकोण है, जो सहयोग, ग्राहक की प्रतिक्रिया और कार्यशील सॉफ़्टवेयर की क्रमिक आपूर्ति पर बल देता है। प्रमुख एजाइल कार्यप्रणालियाँ हैं—स्क्रम (Scrum), कानबन (Kanban), और एक्सट्रीम प्रोग्रामिंग (XP)। एजाइल टीमें "स्प्रिंट" या "इटरेशन" नामक लघु विकास चक्रों में कार्य करती हैं, जिससे परिवर्तनों के अनुसार अनुकूलन करना सरल हो जाता है।
4. **स्क्रम (Scrum)** — यह एक विशिष्ट एजाइल रूपरेखा है जो कार्य को सीमित समयावधि वाले चक्रों (sprints) में विभाजित करती है। इसमें टीमवर्क, नियमित निरीक्षण और अनुकूलन को महत्व दिया जाता है। स्क्रम में उत्पाद स्वामी (Product Owner), स्क्रम मास्टर और विकास टीम जैसे विशेष भूमिकाएँ होती हैं। यह उत्पाद बैकलॉग और बर्नडाउन चार्ट जैसे कलाकृतियों (artifacts) पर आधारित होता है।
5. **कानबन (Kanban)** — यह एक एजाइल कार्यप्रणाली है जो कार्य को दृश्य रूप में प्रस्तुत करने और प्रवाह प्रबंधन (flow management) पर केंद्रित होती है। कार्य वस्तुओं को कानबन बोर्ड पर दर्शाया जाता है और "कार्य प्रगति में" (Work-in-Progress – WIP) की सीमा तय करके आउटपुट को बेहतर बनाया जाता है। यह सतत सुधार और अनुरक्षण संबंधी कार्यों के लिए विशेष रूप से उपयुक्त है।
6. **डेवऑप्स (DevOps)** — यह एक सांस्कृतिक एवं तकनीकी दृष्टिकोण है जो विकास (development) और संचालन (operations) टीमों के बीच सहयोग पर बल देता है। इसका उद्देश्य सॉफ़्टवेयर डिलीवरी पाइपलाइन को स्वचालित करना और उसे अधिक तेज़ व विश्वसनीय बनाना होता है। सतत एकीकरण (CI) और सतत परिनियोजन (CD) डेवऑप्स की सामान्य प्रक्रियाएँ हैं।
7. **लीन विकास (Lean Development)** — यह उत्पादन प्रणाली से प्रेरित एक कार्यप्रणाली है जो सॉफ़्टवेयर विकास प्रक्रिया में अपव्यय को समाप्त करने पर बल देती है। इसका लक्ष्य ग्राहक को अधिकतम मूल्य प्रदान करना, दोषों को न्यूनतम करना और कार्य प्रवाह को अनुकूलित करना होता है। लीन पद्धतियाँ एजाइल और कानबन जैसे अन्य मॉडल्स में सम्मिलित की जा सकती हैं।
8. **स्पाइरल मॉडल (Spiral Model)** — यह मॉडल पुनरावृत्तिमूलक विकास को जोखिम प्रबंधन के तत्वों के साथ संयोजित करता है। इसमें योजना, डिज़ाइन, निर्माण और परीक्षण की कई पुनरावृत्तियाँ होती हैं, जिनमें प्रत्येक चक्र एक "स्पाइरल" का प्रतिनिधित्व करता है। यह जटिल परियोजनाओं के लिए विशेष रूप से उपयुक्त है जहाँ जोखिमों को क्रमिक रूप से निपटाया जाना आवश्यक होता है।

9. **रैपिड एप्लीकेशन डेवलपमेंट (RAD)** — यह मॉडल शीघ्रता से प्रोटोटाइप तैयार करने और प्रयोक्ता की प्रतिक्रिया के आधार पर उन्हें परिष्कृत करने पर केंद्रित होता है। यह उन परियोजनाओं के लिए उपयुक्त होता है जहाँ विकास की गति अत्यंत महत्वपूर्ण होती है तथा आवश्यकताओं में परिवर्तन की संभावना बनी रहती है।
10. **वी-मॉडल (V-Model / Verification and Validation Model)** — यह वॉटरफॉल मॉडल का एक विस्तार है जो विकास के प्रत्येक चरण और उससे संबंधित परीक्षण चरण के बीच संबंध को दर्शाता है। यह प्रत्येक चरण में सत्यापन और प्रमाणीकरण (Verification and Validation) के महत्व को रेखांकित करता है जिससे सॉफ़्टवेयर निर्धारित आवश्यकताओं के अनुरूप तैयार किया जा सके।
11. **इंक्रिमेंटल और इटरेटिव डेवलपमेंट** — इस दृष्टिकोण में परियोजना को छोटे-छोटे भागों में विभाजित करके क्रमशः विकसित किया जाता है। इसमें नियमित परीक्षण और नई सुविधाओं का एकीकरण संभव होता है, जिससे बदलती आवश्यकताओं के अनुसार आसानी से अनुकूलन किया जा सकता है।

सॉफ़्टवेयर टीमें सामान्यतः परियोजना की प्रकृति, संस्थागत बॉर्डर्स और ग्राहक की आवश्यकताओं के आधार पर इन विकास मॉडल्स का चयन करती हैं। इसके अतिरिक्त, आजकल विभिन्न मॉडल्स के तत्वों को सम्मिलित करने वाली **हाइब्रिड कार्यप्रणालियाँ** भी प्रचलन में हैं, ताकि परियोजना की विशिष्ट आवश्यकताओं को प्रभावी तरीके से पूरा किया जा सके।

अभ्यास 4.1

- किसी भी पाँच SDLC मॉडल के नाम लिखिए।
- सॉफ़्टवेयर विकास जीवन चक्र (SDLC) मॉडल अपनाने के तीन मुख्य कारण लिखिए।

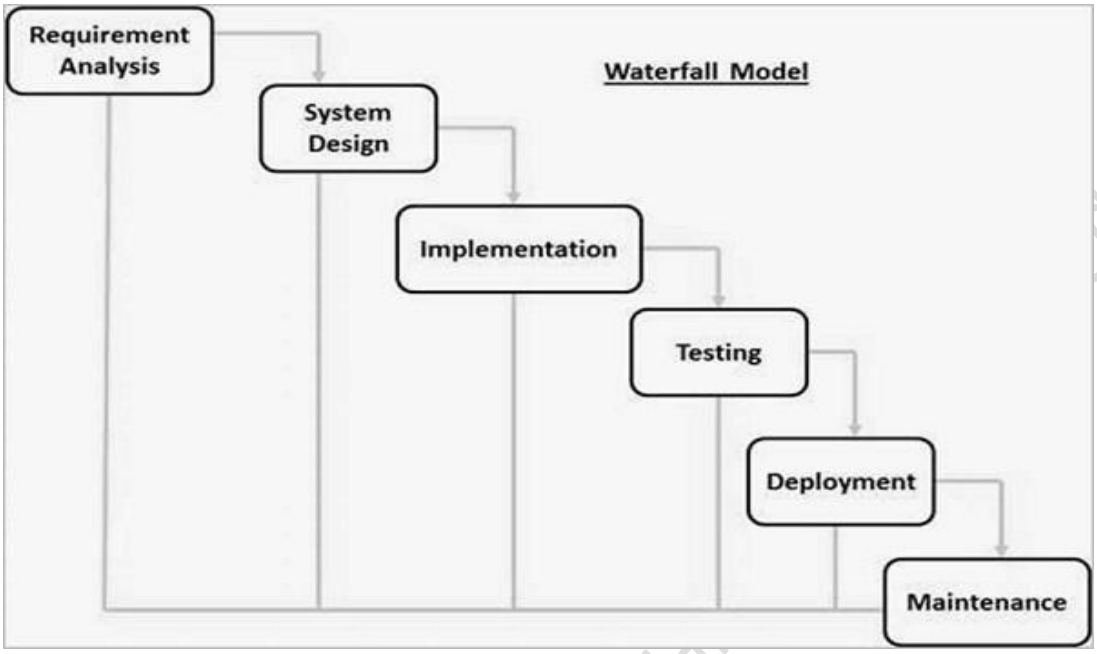
4.3 वॉटरफॉल मॉडल

वॉटरफॉल मॉडल एक पारंपरिक सॉफ़्टवेयर विकास कार्यप्रणाली है जो रैखिक और अनुक्रमिक प्रक्रिया का अनुसरण करती है। इसे सर्वप्रथम डॉ. विंस्टन डब्ल्यू. रॉयस द्वारा 1970 में प्रकाशित एक शोधपत्र में प्रस्तुत किया गया था। इस मॉडल को "वॉटरफॉल" नाम इसलिए दिया गया क्योंकि इसमें सॉफ़्टवेयर विकास की प्रक्रिया को एक झरने की भाँति चरण-दर-चरण नीचे की ओर प्रवाहित होते हुए देखा जाता है, जैसा कि चित्र 4.2 में दर्शाया गया है।

वॉटरफॉल मॉडल के सामान्य चरण निम्नलिखित हैं—

1. **आवश्यकताओं का संकलन एवं विश्लेषण** — इस प्रारंभिक चरण में परियोजना टीम हितधारकों के साथ मिलकर सभी आवश्यकताओं को एकत्र करती है और उनका दस्तावेज़ीकरण करती है। इस चरण का उद्देश्य यह स्पष्ट करना होता है कि सॉफ़्टवेयर को क्या करना है और वह किस प्रकार व्यवहार करेगा।
2. **सिस्टम डिज़ाइन** — एक बार आवश्यकताएँ निश्चित हो जाती हैं, डिज़ाइन चरण आरंभ होता है। इसमें सॉफ़्टवेयर आर्किटेक्ट और डिज़ाइनर विस्तृत प्रणाली संरचना और डिज़ाइन विनिर्देश तैयार करते हैं। इस चरण में प्रणाली की रूपरेखा और घटकों को परिभाषित किया जाता है।
3. **कार्यान्वयन (Implementation)** — इस चरण में सॉफ़्टवेयर का कोडिंग कार्य डिज़ाइन विनिर्देशों के आधार पर शुरू होता है। सॉफ़्टवेयर के प्रत्येक घटक या मॉड्यूल को स्वतंत्र रूप से विकसित किया जाता है।
4. **परीक्षण (Testing)** — कार्यान्वयन के बाद सॉफ़्टवेयर का पूर्ण परीक्षण किया जाता है ताकि यह सुनिश्चित किया जा सके कि वह निर्दिष्ट आवश्यकताओं को पूरा कर रहा है या नहीं। परीक्षण में यूनिट परीक्षण, एकीकरण परीक्षण, सिस्टम परीक्षण और स्वीकृति परीक्षण सम्मिलित होते हैं।
5. **परिनियोजन (Deployment)** — जब परीक्षण पूर्ण हो जाता है और सॉफ़्टवेयर को तैयार घोषित किया जाता है, तब उसे प्रोडक्शन वातावरण में परिनियोजित किया जाता है या प्रयोक्ताओं के लिए उपलब्ध कराया जाता है।

6. **अनुरक्षण एवं समर्थन** — इस चरण में सॉफ़्टवेयर के जीवनचक्र के दौरान अनुरक्षण, बग सुधार और तकनीकी समर्थन दिया जाता है।



चित्र 4.2— वॉटरफॉल मॉडल

वॉटरफॉल मॉडल के लाभ—

1. यह एक संरचित और प्रलेखित (documented) दृष्टिकोण प्रदान करता है।
2. इसे प्रबंधित करना और समझना आसान होता है।
3. यह स्पष्ट और स्थिर आवश्यकताओं वाली परियोजनाओं के लिए उपयुक्त है।

वॉटरफॉल मॉडल की सीमाएँ—

1. यह मानता है कि सभी आवश्यकताएँ प्रारंभ में ही ज्ञात हैं, जो अधिकांशतः संभव नहीं होता।
2. परियोजना के बाद के चरणों में आवश्यकताओं में बदलाव करना कठिन और महँगा होता है।
3. यह लचीलापन प्रदान नहीं करता, इसलिए यह बदलती आवश्यकताओं वाली परियोजनाओं के लिए कम उपयुक्त है।

व्यवहार में देखा गया है कि वॉटरफॉल मॉडल को अब अधिकतर **पुनरावृत्तिमूलक और लचीली कार्यप्रणालियों**, जैसे एजाइल, ने प्रतिस्थापित कर दिया है, जो बदलती आवश्यकताओं के अनुसार अधिक उपयुक्त हैं तथा विकास प्रक्रिया में टीम के सदस्यों और हितधारकों के बीच बेहतर सहयोग को बढ़ावा देती हैं।

4.4 पुनरावृत्त (Iterative) मॉडल

"पुनरावृत्त मॉडल" शब्द का उपयोग विभिन्न क्षेत्रों और संदर्भों में अपनाए गए अनेक पुनरावृत्त दृष्टिकोणों के लिए किया जाता है, जिनमें सॉफ़्टवेयर विकास, परियोजना प्रबंधन और समस्या समाधान शामिल हैं। सामान्यतः, एक इटरेटिव या पुनरावृत्त मॉडल ऐसी प्रक्रिया या कार्यप्रणाली है जिसमें समय के साथ उत्पाद, परियोजना या समाधान को परिष्कृत और सुधारने के उद्देश्य से चरणों या गतिविधियों की एक शृंखला को बार-बार दोहराया जाता है। प्रत्येक पुनरावृत्ति पूर्ववर्ती पुनरावृत्ति पर आधारित होती है, जिसमें फीडबैक और सीखी गई बातों को सम्मिलित कर क्रमिक प्रगति की जाती है।

पुनरावृत्त मॉडल का एक सामान्य उदाहरण सॉफ़्टवेयर विकास में प्रयुक्त "पुनरावृत्त और क्रमिक विकास (Iterative and Incremental Development)" पद्धति है, जिसमें **एजाइल (Agile)** और **स्क्रम (Scrum)** जैसी कार्यप्रणालियाँ शामिल हैं। इस संदर्भ में, पुनरावृत्त मॉडल में किसी परियोजना को छोटे-छोटे हिस्सों या पुनरावृत्तियों में विभाजित किया जाता है, जिनमें से प्रत्येक पुनरावृत्ति एक संभावित

रूप से वितरित करने योग्य उत्पाद भाग (product increment) तैयार करती है। टीमें प्रत्येक पुनरावृत्ति के दौरान कुछ विशेषताओं या आवश्यकताओं पर कार्य करती हैं, हितधारकों से फीडबैक प्राप्त करती हैं, और फिर अगले पुनरावृत्ति में उस फीडबैक के अनुसार सुधार करती हैं। यह प्रक्रिया परियोजना के पूर्ण होने तक चलती रहती है।

पुनरावृत्त मॉडल की प्रमुख विशेषताएँ निम्नलिखित हैं—

1. **पुनरावृत्ति** — इस प्रक्रिया में एक चरण या गतिविधियों के समूह को चक्रों (iterations) के रूप में दोहराया जाता है।
2. **प्रतिक्रिया (Feedback)** — प्रत्येक पुनरावृत्ति में प्रयोक्ताओं, ग्राहकों या हितधारकों से प्राप्त प्रतिक्रिया को सम्मिलित किया जाता है ताकि सुधार किया जा सके।
3. **क्रमिक प्रगति** — प्रत्येक पुनरावृत्ति में नया फ़ीचर जोड़कर या मौजूदा फ़ीचरों को परिष्कृत कर उत्पाद या परियोजना क्रमिक रूप से विकसित होती है।
4. **लचीलापन** — पारंपरिक रैखिक मॉडल की तुलना में पुनरावृत्त मॉडल अधिक लचीले और परिवर्तनीय आवश्यकताओं के अनुरूप होते हैं।
5. **निरंतर सुधार** — प्रत्येक पुनरावृत्ति में पिछले की तुलना में बेहतर परिणाम देने के लक्ष्य से सुधार पर ध्यान दिया जाता है।
6. **जोखिम में कमी** — संभावित समस्याओं और जोखिमों को प्रक्रिया के आरंभिक चरणों में पहचानकर, पुनरावृत्त मॉडल परियोजना जोखिम को कम करने में सहायक होता है।

यह ध्यान देना आवश्यक है कि पुनरावृत्त मॉडलों के विशेष विवरण उस क्षेत्र और संदर्भ पर निर्भर करते हैं जिसमें उनका उपयोग किया जाता है। विभिन्न उद्योगों और क्षेत्रों ने अपनी आवश्यकताओं और चुनौतियों के अनुरूप इन मॉडलों को अपनाया है।

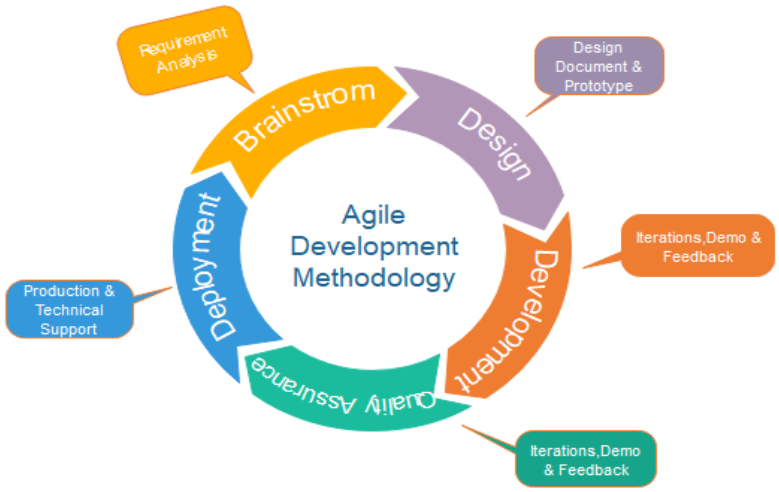
कुल मिलाकर, **पुनरावृत्त मॉडल** जटिल परियोजनाओं, अनिश्चित आवश्यकताओं, अथवा ऐसे परिदृश्यों के लिए एक प्रभावी दृष्टिकोण है जहाँ आरंभ में अंतिम लक्ष्य पूरी तरह स्पष्ट नहीं होता। यह लचीलापन, अनुकूलनशीलता और फीडबैक को सम्मिलित करने की क्षमता प्रदान करता है, जिससे बेहतर परिणाम प्राप्त किए जा सकते हैं।

अभ्यास 4.2

- वॉटरफॉल मॉडल के चरणों के नाम सूचीबद्ध करें।
- वॉटरफॉल मॉडल का चित्र बनाएँ।
- पुनरावृत्त मॉडलों की विभिन्न विशेषताएँ सूचीबद्ध करें।

4.5 एजाइल मॉडल

एजाइल मॉडल, जिसे प्रायः **एजाइल कार्यप्रणाली (Agile Methodology)** या **एजाइल रूपरेखा (Agile Framework)** के रूप में जाना जाता है, सॉफ़्टवेयर विकास और परियोजना प्रबंधन में उपयोग की जाने वाली सिद्धांतों और प्रथाओं का एक समूह है। यह लचीलापन, सहयोग, ग्राहक प्रतिक्रिया और पुनरावृत्त प्रगति पर बल देता है। एजाइल कार्यप्रणालियाँ बदलती आवश्यकताओं के प्रति शीघ्र प्रतिक्रिया देने और शीघ्र मूल्यवर्धक सॉफ़्टवेयर प्रदान करने के उद्देश्य से विकसित की गई हैं। एजाइल की कई विधियाँ हैं, जिनमें **स्क्रम (Scrum)** और **कानबन (Kanban)** दो सर्वाधिक प्रचलित विधियाँ हैं। **एजाइल मॉडल को चित्र 4.3 में दर्शाया गया है।**



चित्र 4.3— एजाइल मॉडल

एजाइल से संबद्ध कुछ प्रमुख अवधारणाएँ और सिद्धांत इस प्रकार हैं—

1. **पुनरावृत्त और क्रमिक विकास** — एजाइल परियोजनाओं को छोटे-छोटे खंडों या पुनरावृत्तियों (iterations) में विभाजित किया जाता है, जो प्रायः दो से चार सप्ताह की होती हैं। प्रत्येक पुनरावृत्ति एक संभावित रूप से वितरित करने योग्य उत्पाद भाग (product increment) उत्पन्न करती है, जिससे निरंतर सुधार की प्रक्रिया बनी रहती है।
2. **ग्राहक सहयोग** — एजाइल टीमें परियोजना के दौरान ग्राहकों, हितधारकों और वास्तविक प्रयोक्ताओं के साथ घनिष्ठ रूप से कार्य करती हैं ताकि यह सुनिश्चित किया जा सके कि उत्पाद उनकी आवश्यकताओं और अपेक्षाओं के अनुरूप हो।
3. **लचीलापन** — एजाइल एक कठोर योजना का अनुसरण करने की तुलना में परिवर्तनों के प्रति उत्तरदायित्व को प्राथमिकता देता है। यह मानता है कि आवश्यकताएँ बदल सकती हैं और उनके अनुसार स्वयं को अनुकूलित करता है।
4. **क्रॉस-फ़ंक्शनल टीमें** — एजाइल टीमें प्रायः बहु-कार्यात्मक (cross-functional) होती हैं, जिनमें डेवलपर, परीक्षक, डिज़ाइनर आदि विभिन्न कौशलों वाले सदस्य होते हैं। ये सहयोग और परियोजना की सफलता के लिए साझा उत्तरदायित्व को प्रोत्साहित करते हैं।
5. **प्रोडक्ट बैकलॉग** — स्क्रम में, जो कि एक लोकप्रिय एजाइल विधि है, एक **प्रोडक्ट बैकलॉग** होता है—यह विशेषताओं और प्रयोक्ता कहानियों (user stories) की एक प्राथमिकता-आधारित सूची होती है जो विकास टीम के कार्य को मार्गदर्शित करती है।
6. **स्प्रिंट्स** — स्क्रम में कार्य सीमित समयावधि वाली पुनरावृत्तियों को **स्प्रिंट** कहा जाता है। प्रत्येक स्प्रिंट के दौरान, टीम बैकलॉग से कुछ चयनित वस्तुओं को पूरा करने की प्रतिबद्धता लेती है।
7. **डेली स्टैंडअप (Scrum)** — टीमें प्रतिदिन एक संक्षिप्त बैठक करती हैं जिसमें प्रगति, चुनौतियों और आगामी योजनाओं पर चर्चा की जाती है। ये बैठकें संक्षिप्त होती हैं और टीम को समन्वित बनाए रखने में सहायक होती हैं।
8. **रेट्रोस्पेक्टिव्स** — प्रत्येक पुनरावृत्ति के अंत में, एजाइल टीमें यह समीक्षा करती हैं कि क्या अच्छा हुआ, क्या सुधारा जा सकता है, और अगली पुनरावृत्ति में क्या परिवर्तन किया जाए।
9. **सतत एकीकरण और परीक्षण** — एजाइल अभ्यासों में प्रायः **सतत एकीकरण (Continuous Integration)** शामिल होता है, जिसमें कोड परिवर्तनों को नियमित रूप से एकीकृत और परीक्षण किया जाता है ताकि समस्याओं का शीघ्र पता चल सके।
10. **कार्यशील सॉफ़्टवेयर को प्रगति का मापदंड मानना** — एजाइल कार्यशील सॉफ़्टवेयर की डिलीवरी को प्रगति का मुख्य मापदंड मानता है, जबकि पारंपरिक परियोजना प्रबंधन में प्रायः प्रलेखन या चरणबद्ध उपलब्धियों को प्राथमिकता दी जाती है।

11. **न्यूनतम कार्यशील उत्पाद (Minimal Viable Product – MVP)** — एजाइल सबसे छोटे ऐसे फ़ीचर सेट के विकास को प्रोत्साहित करता है जो प्रयोक्ताओं के लिए मूल्य प्रदान करे। इससे किसी मूल उत्पाद को शीघ्रता से जारी कर, बाद में पुनरावृत्त रूप में सुधार किया जा सकता है।
12. **ग्राहक प्रतिक्रिया** — एजाइल टीमें ग्राहकों और प्रयोक्ताओं से सक्रिय रूप से फीडबैक प्राप्त करती हैं और उसे विकास प्रक्रिया में सम्मिलित करती हैं, ताकि उत्पाद उनकी आवश्यकताओं के अनुकूल हो सके।

लोकप्रिय एजाइल कार्यप्रणालियाँ (Agile Methodologies) में शामिल हैं —

- **स्क्रम (Scrum)** — यह एक रूपरेखा (framework) है जो कार्य को निश्चित अवधि के स्प्रिंट्स (Sprints) में संगठित करती है। इसमें स्क्रम मास्टर (Scrum Master), उत्पाद स्वामी (Product Owner) और विकास दल (Development Team) जैसे भूमिकाएँ होती हैं।
- **कानबन (Kanban)** — यह एक दृश्यात्मक विधि है जिसमें कार्य प्रवाह (workflow) को प्रबंधित करने और अनुकूलित करने के लिए कानबन बोर्ड का उपयोग किया जाता है। इसका मुख्य उद्देश्य प्रक्रिया में चल रहे कार्य की मात्रा को न्यूनतम करना होता है।
- **एक्सट्रीम प्रोग्रामिंग (Extreme Programming — XP)** — यह एक एजाइल पद्धति है जो तकनीकी उत्कृष्टता पर विशेष बल देती है। इसमें पेयर प्रोग्रामिंग (Pair Programming) और टेस्ट-ड्रिवन डेवलपमेंट (Test-Driven Development) जैसी प्रथाएँ सम्मिलित हैं।

एजाइल मॉडल ने सॉफ़्टवेयर विकास में व्यापक लोकप्रियता प्राप्त की है और इसकी अनुकूलन क्षमता तथा ग्राहक-केंद्रित दृष्टिकोण के कारण इसे अन्य उद्योगों में भी अपनाया जा रहा है।

4.6 देवऑप्स (DevOps)

देवऑप्स एक ऐसी कार्यप्रणाली, सिद्धांतों और सांस्कृतिक दर्शन का समुच्चय है जिसका उद्देश्य सॉफ़्टवेयर विकास (Development) और आईटी संचालन (Operations) टीमों के बीच सहयोग और संप्रेषण को बेहतर बनाना है। देवऑप्स का प्रमुख उद्देश्य सॉफ़्टवेयर डिलीवरी प्रक्रिया को सुव्यवस्थित करना, विकास चक्र की अवधि को घटाना और सॉफ़्टवेयर उत्पादों की समग्र गुणवत्ता में सुधार करना है। यह विकास और संचालन के बीच की दूरी को कम करता है और साझा उत्तरदायित्व तथा सतत सुधार की संस्कृति को बढ़ावा देता है।

नीचे देवऑप्स की कुछ प्रमुख विशेषताएँ दी गई हैं —

1. **सहयोग (Collaboration)** — देवऑप्स क्रॉस-फंक्शनल टीमों को सम्पूर्ण सॉफ़्टवेयर विकास जीवन चक्र में निकटता से सहयोग करने के लिए प्रोत्साहित करता है — योजना बनाने, कोडिंग, परीक्षण, परिनियोजन (Deployment) और निगरानी तक। इस सहयोग से समस्याओं की शीघ्र पहचान और समाधान संभव होता है।
2. **स्वचालन (Automation)** — स्वचालन, देवऑप्स का एक मूलभूत सिद्धांत है। निर्माण (build), परीक्षण और परिनियोजन जैसे दोहराव वाले कार्यों का स्वचालन करके टीमें त्रुटियों को कम कर सकती हैं, कार्यदक्षता बढ़ा सकती हैं और प्रक्रियाओं में समानता सुनिश्चित कर सकती हैं।
3. **सतत एकीकरण (Continuous Integration — CI)** — इसमें विभिन्न डेवलपर्स द्वारा किए गए कोड परिवर्तनों को स्वचालित रूप से एक साझा रिपॉजिटरी में जोड़ा जाता है। यह प्रक्रिया सुनिश्चित करती है कि नया कोड नियमित रूप से परीक्षण हो, जिससे एकीकरण से संबंधित समस्याओं की संभावना घटती है और त्रुटियाँ शीघ्र पकड़ी और सुधारी जा सकती हैं।
4. **सतत परिनियोजन (Continuous Delivery — CD)** — यह सतत एकीकरण का विस्तार है, जिसमें कोड परिवर्तनों को स्वचालित परीक्षणों में सफल होने के बाद तुरंत ही प्रोडक्शन या स्टेजिंग वातावरण में परिनियोजित किया जाता है। इससे तेज़ और विश्वसनीय सॉफ़्टवेयर रिलीज़ संभव होती है।

5. **कोड के रूप में अवसंरचना (Infrastructure as Code — IaC)** — इसमें सर्वर, डेटाबेस, नेटवर्क आदि जैसी अवसंरचना संसाधनों को कोड के माध्यम से परिभाषित और उपलब्ध कराया जाता है। यह अवसंरचना के परिनियोजन को सुसंगत, दोहराने योग्य और प्रबंधनीय बनाता है।
6. **निगरानी और प्रतिपुष्टि (Monitoring and Feedback)** — देवऑप्स, प्रोडक्शन में अनुप्रयोगों और अवसंरचना की निगरानी पर विशेष बल देता है। टीमें प्रदर्शन और प्रयोक्ता व्यवहार को समझने के लिए निगरानी उपकरणों का उपयोग करती हैं, जिससे डेटा-आधारित सुधार किए जा सकते हैं।
7. **माइक्रोसर्विसेज और कंटेनर (Microservices and Containers)** — देवऑप्स अक्सर माइक्रोसर्विसेज आर्किटेक्चर और डॉकर (Docker) जैसे कंटेनरीकरण तकनीकों के साथ अपनाया जाता है। ये तकनीकें जटिल और स्केलेबल अनुप्रयोगों के निर्माण, परिनियोजन और प्रबंधन को सरल बनाती हैं।
8. **सुरक्षा (Security)** — देवऑप्स में सुरक्षा प्रथाओं को विकास और परिनियोजन प्रक्रिया में एकीकृत करने (DevSecOps) पर भी बल दिया जाता है। इससे यह सुनिश्चित होता है कि सुरक्षा को अंतिम चरण में नहीं बल्कि आरंभ से ही विचार में लिया जाए।
9. **संस्कृति (Culture)** — देवऑप्स केवल उपकरणों और प्रथाओं तक सीमित नहीं है, बल्कि यह एक सांस्कृतिक परिवर्तन भी है। यह साझा उत्तरदायित्व, निरंतर अधिगम और प्रयोग तथा पुनरावृत्ति की संस्कृति को बढ़ावा देता है।
10. **प्रतिपुष्टि चक्र (Feedback Loops)** — देवऑप्स टीमें विकास जीवन चक्र के प्रत्येक चरण में प्रतिपुष्टि चक्र को प्रोत्साहित करती हैं। यह प्रतिपुष्टि सुधार के क्षेत्रों की पहचान करने में सहायक होती है और प्रक्रियाओं एवं उपकरणों के विकास को प्रेरित करती है।

देवऑप्स सॉफ़्टवेयर उद्योग में दिन-प्रतिदिन अधिक लोकप्रिय होता जा रहा है क्योंकि यह संगठनों को सॉफ़्टवेयर को अधिक तेज़ी से, अधिक विश्वसनीय और अधिक सुरक्षित रूप से डिलीवर करने में सक्षम बनाता है। यह विकास और संचालन टीमों के बीच की बाधाओं को समाप्त करता है, जिससे नवाचार की गति तेज होती है और ग्राहक अनुभव में सुधार होता है। तथापि, देवऑप्स पद्धतियों को प्रभावी तरीके से लागू करने के लिए सांस्कृतिक परिवर्तन, कौशल विकास और स्वचालन व सहयोग का समर्थन करने वाले उपयुक्त उपकरणों की आवश्यकता होती है।

सारांश

- आईटी एप्लीकेशन विकास मॉडल, सॉफ्टवेयर विकास को संरचित दृष्टिकोणों के साथ दिशा प्रदान करते हैं।
- सॉफ्टवेयर विकास जीवन चक्र (SDLC) मॉडल, परियोजना प्रबंधन को एक व्यवस्थित अवसंरचना प्रदान करते हैं।
- वॉटरफॉल मॉडल एक रैखिक मॉडल है जो स्थिर आवश्यकताओं के लिए उपयुक्त होता है।
- एजाइल (Agile) लचीलापन, ग्राहक सहयोग और क्रमिक प्रगति पर बल देता है।
- देवऑप्स (DevOps), विकास और संचालन के बीच की दूरी को कम करके सॉफ्टवेयर डिलीवरी को सुव्यवस्थित करता है।
- कंटीन्युअस इंटीग्रेशन (CI) कोड एकीकरण और परीक्षण को स्वचालित करता है।
- पुनरावृत्त (Iterative) मॉडल क्रमिक प्रगति और परिवर्तित आवश्यकताओं के अनुरूप ढलने की अनुमति देते हैं।
- लीन विकास (Lean Development) ग्राहक को मूल्य प्रदान करने, दोषों को न्यूनतम करने और कार्य प्रवाह को अनुकूलित करने पर केंद्रित है।
- एजाइल विकास प्रक्रिया के दौरान ग्राहक की प्रतिक्रिया और सहयोग को प्रोत्साहित करता है।
- SDLC मॉडल, दक्ष, नियंत्रित और उच्च गुणवत्ता वाले सॉफ्टवेयर विकास को सुनिश्चित करने में महत्वपूर्ण भूमिका निभाते हैं।

अपनी प्रगति जाँचें

क. बहुविकल्पी प्रश्न

1. वॉटरफॉल मॉडल उन परियोजनाओं के लिए उपयुक्त है जिनमें —
   (क) आवश्यकताएँ बदलती और गतिशील होती हैं
   (ख) आवश्यकताएँ स्थिर और स्पष्ट रूप से परिभाषित होती हैं
   (ग) बार-बार ग्राहक की प्रतिक्रिया ली जाती है
   (घ) निरंतर एकीकरण किया जाता है

2. देवऑप्स (DevOps) का मुख्य उद्देश्य क्या है?
   (क) स्वचालन
   (ख) विकास और संचालन के बीच सहयोग
   (ग) प्रलेखन
   (घ) जोखिम प्रबंधन

3. पुनरावृत्त मॉडल किस प्रकार की परियोजनाओं के लिए उपयुक्त है?
   (क) अनिश्चित आवश्यकताओं वाली परियोजनाएँ
   (ख) स्थिर और स्पष्ट आवश्यकताओं वाली परियोजनाएँ
   (ग) जिनमें ग्राहक की कोई भागीदारी न हो
   (घ) जिनमें किसी परिवर्तन की गुंजाइश न हो

4. देवऑप्स का मुख्य लक्ष्य क्या है?
   (क) प्रलेखन प्रक्रिया को बेहतर बनाना
   (ख) विकास चक्र की अवधि को कम करना
   (ग) टीमों के बीच सहयोग को कम करना
   (घ) परियोजना की जटिलता को बढ़ाना

5. एजाइल में प्रोडक्ट बैकलॉग क्या होता है?
   (क) पूर्ण की गई विशेषताओं की सूची
   (ख) प्राथमिकता के अनुसार व्यवस्थित विशेषताओं और प्रयोक्ता कहानियों की सूची
   (ग) विकास टीम के सदस्यों की सूची
   (घ) ग्राहक प्रतिक्रिया की सूची

6. स्पाइरल मॉडल विशेष रूप से उपयोगी है —
   (क) स्पष्ट आवश्यकताओं वाली परियोजनाओं के लिए
   (ख) न्यूनतम जोखिम वाली परियोजनाओं के लिए
   (ग) जटिल परियोजनाओं के लिए, जहाँ जोखिमों को क्रमिक रूप से संबोधित किया जाना आवश्यक हो
   (घ) त्वरित अनुप्रयोग विकास के लिए

7. लीन विकास (Lean Development) का मुख्य केंद्र बिंदु क्या है?
   (क) अधिकतम दोष उत्पन्न करना

(ख) ग्राहकों को मूल्य प्रदान करना
(ग) कार्य प्रवाह को धीमा करना
(घ) आवश्यकताओं में बार-बार परिवर्तन करना

8. एजाइल में "स्प्रिंट" क्या होता है?
(क) लंबी दूरी की दौड़
(ख) एक सीमित समय की विकास पुनरावृत्ति (iteration)
(ग) डिज़ाइन चरण
(घ) ग्राहक बैठक

9. कौन से SDLC मॉडल में विकास और परीक्षण चरणों के बीच संबंध पर विशेष बल दिया जाता है?
(क) V-मॉडल
(ख) स्क्रम
(ग) स्पाइरल मॉडल
(घ) रैड

10. एजाइल में "MVP" का क्या अर्थ है?
(क) मोस्ट वैल्यूएबल प्रोडक्ट
(ख) मैक्सिमम वायएबल प्रोडक्ट
(ग) मिनिमम वायएबल प्रोडक्ट
(घ) मेजर विजुअल प्रोटोटाइप

ख. रिक्त स्थान भरें

1. पुनरावृत्त मॉडल की मुख्य विशेषता यह है कि वे _____ में क्रियाओं या चरणों की पुनरावृत्ति करते हैं।
2. देवऑप्स विकास और संचालन के बीच की दूरी को _____ उत्तरदायित्व और सतत सुधार की संस्कृति को बढ़ावा देकर कम करता है।
3. एजाइल में टीमें प्रतिदिन स्टैंड-अप बैठकें आयोजित करती हैं जिनमें प्रगति, चुनौतियों और _____ पर चर्चा की जाती है।
4. स्पाइरल मॉडल, पुनरावृत्त विकास को _____ प्रबंधन के तत्वों के साथ जोड़ता है और जटिल परियोजनाओं के लिए उपयोगी है।
5. RAD उन परियोजनाओं के लिए उपयुक्त है जिनमें विकास की गति महत्वपूर्ण होती है और आवश्यकताएँ _____ की जा सकती हैं।
6. V-मॉडल में, सॉफ्टवेयर को निर्दिष्ट आवश्यकताओं के अनुरूप सुनिश्चित करने हेतु प्रत्येक चरण में _____ और सत्यापन पर बल दिया जाता है।
7. कंटीन्युअस इंटीग्रेशन (CI) में कई डेवलपर्स द्वारा किए गए कोड परिवर्तनों को एक साझा _____ में स्वचालित रूप से एकीकृत किया जाता है।
8. कानबन एक एजाइल कार्यप्रणाली है जो कार्य के दृश्यमान रूप से प्रबंधन और _____ के प्रबंधन पर केंद्रित होती है ताकि उत्पादनशीलता बढ़ सके।
9. पुनरावृत्त मॉडल लचीलापन, अनुकूलन और _____ के समावेश की अनुमति देते हैं ताकि बेहतर परिणाम प्राप्त किए जा सकें।
10. देवऑप्स, संपूर्ण सॉफ्टवेयर विकास _____ के दौरान क्रॉस-फंक्शनल टीमों को मिलकर कार्य करने के लिए प्रोत्साहित करता है।

ग. सही या गलत बताइए

1. SDLC मॉडल ऐसे अवसंरचना हैं जो हार्डवेयर एप्लीकेशन के विकास की प्रक्रिया को निर्देशित करते हैं।
2. वॉटरफॉल मॉडल बार-बार बदलती आवश्यकताओं वाली परियोजनाओं के लिए उपयुक्त है।
3. एजाइल मॉडल मूल्यवान सॉफ़्टवेयर को छोटे विकास चक्रों के माध्यम से क्रमिक रूप से वितरित करने पर बल देता है।
4. स्क्रम एक विशिष्ट एजाइल अवसंरचना है जिसमें स्क्रम मास्टर और प्रोडक्ट ओनर जैसी भूमिकाएँ शामिल नहीं होतीं।
5. देवऑप्स, विकास और विपणन टीमों के बीच सहयोग पर केंद्रित होता है।
6. स्पाइरल मॉडल सरल और सीधे-साधे परियोजनाओं के लिए सबसे उपयुक्त है।
7. RAD उन परियोजनाओं के लिए उपयुक्त है जिनमें विकास की गति महत्वपूर्ण नहीं होती।
8. V-मॉडल, विकास के प्रत्येक चरण में सत्यापन और मान्यता के महत्व को उजागर करता है।
9. कंटीन्युअस इंटीग्रेशन (CI) में कोड परिवर्तनों का एकीकरण मैन्युअल रूप से किया जाता है।
10. कानबन में बल कार्य प्रगति (WIP) को अधिकतम करने पर होता है।

घ. लघु उत्तर प्रश्न

1. आईटी एप्लीकेशन विकास मॉडल क्या होते हैं, और वे सॉफ्टवेयर विकास में क्यों महत्वपूर्ण होते हैं?
2. सॉफ्टवेयर विकास जीवन चक्र (SDLC) मॉडल का एप्लीकेशन विकास में मुख्य उद्देश्य क्या है?
3. वॉटरफॉल मॉडल का एक लाभ और एक सीमा बताइए।
4. एजाइल में, विकास प्रक्रिया के दौरान ग्राहक सहयोग का क्या महत्व है?
5. देवऑप्स, विकास और संचालन टीमों के बीच सहयोग को कैसे बेहतर बनाता है?
6. देवऑप्स में कंटीन्युअस इंटीग्रेशन (CI) का मुख्य केंद्र बिंदु क्या है?
7. पारंपरिक रैखिक मॉडल की तुलना में पुनरावृत्त मॉडल कैसे भिन्न होते हैं?
8. एजाइल में "कार्यरत सॉफ़्टवेयर प्रगति का मापक है"—इस सिद्धांत की व्याख्या कीजिए।
9. स्क्रम और कानबन के अतिरिक्त किसी एक एजाइल कार्यप्रणाली का नाम बताकर उसकी विशेषताओं का संक्षिप्त वर्णन कीजिए।
10. लीन विकास (Lean Development) का मुख्य लक्ष्य क्या है, और यह सॉफ्टवेयर विकास प्रक्रिया में उसे कैसे प्राप्त करता है?

सत्र 5
वेब डिज़ाइन विनिर्देश

मिलिए अजय से, जो एक ऐसा बच्चा है जिसे इंटरनेट बहुत पसंद है। वह अपनी खुद की वेबसाइट बनाना चाहता था, इसलिए उसने "वेब डिज़ाइन विनिर्देशों" के बारे में सीखा। उसे यह बात समझ में आई कि यह एक पेड़ के घर (ट्रीहाउस) जैसा होता है। उसने कुछ सरल नियम सीखे, जैसे—

रंग—जैसे ट्रीहाउस को रंगने के लिए रंग चुनना।
फॉन्ट्स—जैसे ट्रीहाउस पर लगे बोर्ड पर लिखे शब्दों का रूप चुनना।
चित्र—ट्रीहाउस को आकर्षक पोस्टरों से सजाना।
लेआउट—ट्रीहाउस के अंदर फर्नीचर को व्यवस्थित करना।
लिंक्स—ट्रीहाउस में गुप्त रास्ते बनाना।

इन नियमों की मदद से अजय ने एक शानदार वेबसाइट बनाई, जिसे हर जगह से लोग देखने आते थे। अजय ने यह सीखा कि ये नियम एक खजाने के नक्शे की तरह हैं, जिनकी मदद से वह एक शानदार वेबसाइट बना सकता है। जैसे वह एक ट्रीहाउस को सजाता है, उसी तरह वह सबके आनंद के लिए एक अद्भुत ऑनलाइन स्थान बना सकता है — जैसा कि चित्र 5.1 में दिखाया गया है।

चित्र 5.1— अजय वेबसाइट डिज़ाइन करते हुए

इस अध्याय में, आप वेब डिज़ाइन विनिर्देशों, व्यावसायिक आवश्यकता विनिर्देश (Business Requirement Specifications), व्यावसायिक आवश्यकता के दस्तावेज़ के घटक, इसके लाभ, दस्तावेज़ तैयार करने की प्रक्रिया, और सॉफ़्टवेयर आवश्यकता विनिर्देश (Software Requirement Specification) के बारे में समझेंगे।

5.1 वेब डिज़ाइन विनिर्देश

वेब डिज़ाइन विनिर्देश परियोजना और उसके उद्देश्यों के अनुसार भिन्न हो सकते हैं। हालांकि, वेबसाइट डिज़ाइन करते समय ध्यान में रखने योग्य कुछ सामान्य विनिर्देश और दिशानिर्देश निम्नलिखित हैं—

1. **उद्देश्य और लक्ष्य** — वेबसाइट का उद्देश्य परिभाषित करें (जैसे— सूचनात्मक, ई-कॉमर्स, पोर्टफोलियो) और वेबसाइट के लिए स्पष्ट लक्ष्य निर्धारित करें (जैसे— बिक्री में वृद्धि, जानकारी प्रदान करना, ब्रांड जागरूकता बनाना)।
2. **लक्ष्य दर्शक** — वेबसाइट के लिए प्राथमिक दर्शक की पहचान करें (जैसे— आयु वर्ग, रुचियाँ, स्थान) और प्रयोक्ता व्यक्तित्व (User Personas) पर विचार करें ताकि दर्शकों की आवश्यकताओं को बेहतर समझा जा सके।

3. **डिज़ाइन संकल्पना** — एक डिज़ाइन अवधारणा तैयार करें जो ब्रांड की पहचान और लक्ष्यों के अनुरूप हो। रंग योजना, टाइपोग्राफ़ी और दृश्य तत्वों का चयन करें जो लक्षित दर्शकों को आकर्षित करें।
4. **लेआउट और नेविगेशन** — वेब पृष्ठों के लेआउट का निर्णय लें (जैसे— एक-स्तरीय, बहु-स्तरीय) और एक स्पष्ट एवं प्रयोक्ता अनुकूल नेविगेशन संरचना की योजना बनाएं जिसमें सहज मेनू और लिंक हों।
5. **सामग्री कार्यनीति** — यह निर्धारित करें कि वेबसाइट पर कौन-सी सामग्री होगी (जैसे— टेक्स्ट, चित्र, वीडियो) और एक सामग्री योजना तैयार करें जो समग्र डिज़ाइन के अनुरूप हो।
6. **अनुक्रियाशील डिज़ाइन (Responsive Design)** — यह सुनिश्चित करें कि वेबसाइट विभिन्न स्क्रीन आकारों और उपकरणों (डेस्कटॉप, टैबलेट, मोबाइल) के अनुसार अनुकूल हो।
7. **पृष्ठ लोड गति** — तेज़ लोडिंग के लिए चित्रों और कोड को अनुकूलित करें तथा प्रदर्शन सुधार हेतु सामग्री वितरण नेटवर्क (CDN) का उपयोग करें।
8. **सुलभता (Accessibility)** — डिज़ाइन करते समय यह सुनिश्चित करें कि वेबसाइट दिव्यांग व्यक्तियों के लिए उपयोगी हो और WCAG (Web Content Accessibility Guidelines) मानकों का पालन करें।
9. **ब्राउज़र संगतता** — विभिन्न वेब ब्राउज़रों पर वेबसाइट का परीक्षण करें ताकि यह विभिन्न प्लेटफार्मों पर समान रूप से कार्य करे।
10. **सर्च इंजन अनुकूलन (SEO)** — बेहतर खोज परिणाम के लिए ऑन-पेज SEO की सर्वोत्तम प्रथाओं (जैसे— मेटा टैग्स, कीवर्ड अनुकूलन) को लागू करें।
11. **सुरक्षा** — सामान्य वेब खतरों से सुरक्षा हेतु उपायों को शामिल करें (जैसे— SSL प्रमाणपत्र, डेटा एन्क्रिप्शन)।
12. **कंटेंट मैनेजमेंट सिस्टम (CMS)** — आवश्यक होने पर एक CMS (जैसे— वर्डप्रेस, ड्रुपल) चुनें और परियोजना की आवश्यकताओं के अनुसार अनुकूलित करें।
13. **परीक्षण और गुणवत्ता आश्वासन** — त्रुटियों की पहचान और समाधान के लिए गहन परीक्षण करें और विभिन्न उपकरणों, ब्राउज़रों और स्क्रीन आकारों के साथ संगतता सुनिश्चित करें।
14. **प्रदर्शन निगरानी** — वेबसाइट प्रदर्शन और प्रयोक्ता व्यवहार की निगरानी के लिए उपकरण स्थापित करें (जैसे— Google Analytics)।
15. **बैकअप और पुनर्प्राप्ति** — नियमित बैकअप प्रक्रिया और आपदा पुनर्प्राप्ति योजना सुनिश्चित करें।
16. **कानूनी और अनुपालन** — कानूनी आवश्यकताओं (जैसे— गोपनीयता हेतु GDPR) और बौद्धिक संपदा अधिकारों के साथ अनुपालन सुनिश्चित करें।
17. **रखरखाव और अद्यतन** — नियमित रखरखाव, अद्यतन और सामग्री प्रबंधन की योजना बनाएं।
18. **प्रयोगकर्ता प्रशिक्षण** — आवश्यकता पड़ने पर वेबसाइट व्यवस्थापकों और सामग्री निर्माताओं के लिए प्रशिक्षण प्रदान करें।
19. **लॉन्च योजना** — विकास से उत्पादन तक के संक्रमण को सुगम बनाने के लिए एक लॉन्च योजना तैयार करें।
20. **प्रतिक्रिया और सुधार** — लॉन्च के बाद प्रयोक्ता प्रतिक्रिया एकत्र करें और आवश्यकता के अनुसार सुधारों के लिए तैयार रहें।

वेब डिज़ाइन एक गतिशील क्षेत्र है, और सर्वोत्तम प्रथाएँ समय के साथ बदलती रहती हैं। आधुनिक और प्रभावी वेबसाइट बनाने के लिए नवीनतम प्रवृत्तियों और तकनीकों से अपडेट रहना आवश्यक है। इसके अतिरिक्त, वेब डेवलपर्स, सामग्री निर्माताओं और अन्य हितधारकों के साथ सहयोग भी एक सफल वेब डिज़ाइन परियोजना के लिए अत्यंत महत्वपूर्ण है।

5.2 व्यावसाय आवश्यकता विनिर्देश (Business Requirement Specifications - BRS)

व्यावसाय आवश्यकता विनिर्देश (BRS) एक ऐसा दस्तावेज़ होता है जिसमें किसी परियोजना या प्रणाली की **व्यावसायिक आवश्यकताओं और उद्देश्यों** को रेखांकित किया जाता है। यह किसी नए उत्पाद, सेवा या सॉफ़्टवेयर अनुप्रयोग के विकास के लिए आधारभूमि प्रदान करता है। BRS में स्पष्ट एवं विस्तारपूर्वक वर्णन किया जाता है कि परियोजना के माध्यम से व्यावसाय क्या प्राप्त करना

चाहता है, जिससे विकास एवं कार्यान्वयन प्रक्रिया में सहायता मिलती है। व्यावसाय आवश्यकता विनिर्देश (BRS) में आमतौर पर निम्नलिखित प्रमुख घटक सम्मिलित होते हैं—

1. **उद्देश्य (Objective)** — परियोजना द्वारा प्राप्त किए जाने वाले समग्र लक्ष्य या उद्देश्य का संक्षिप्त विवरण दें। यह इस बात का उच्च स्तरीय दृष्टिकोण प्रदान करता है कि व्यावसाय क्या प्राप्त करना चाहता है।
2. **परिव्याप्ति (Scope)** — यह परिभाषित करें कि परियोजना में क्या शामिल होगा और क्या नहीं। इससे अपेक्षाओं को नियंत्रित करने और कार्य की सीमा को अनावश्यक रूप से बढ़ने से रोकने में मदद मिलती है।
3. **हितधारक (Stakeholders)** — परियोजना से जुड़े सभी हितधारकों को उनकी भूमिकाओं और जिम्मेदारियों सहित पहचानें और सूचीबद्ध करें। इसमें व्यावसाय के स्वामी, परियोजना प्रबंधक, प्रयोक्ता और बाहरी भागीदार शामिल हो सकते हैं।
4. **कार्यक्षमता आवश्यकताएँ (Functional Requirements)** — उन विशिष्ट कार्यों और विशेषताओं का विवरण दें जो व्यावसायिक उद्देश्यों की पूर्ति के लिए परियोजना या प्रणाली में आवश्यक हैं। इस खंड में उपयोग के मामलों, प्रयोक्ता कथाओं और वर्कफ़्लो आरेखों को शामिल किया जाता है।
5. **गैर-कार्यक्षमता आवश्यकताएँ (Non-Functional Requirements)** — वे गुणवत्ता गुणधर्म या सीमाएँ जिनका परियोजना द्वारा पालन किया जाना आवश्यक है, जैसे प्रदर्शन, सुरक्षा, विस्तार क्षमता, तथा नियामक अनुपालन।
6. **बाधाएँ (Constraints)** — वे सभी सीमाएँ जो परियोजना को प्रभावित कर सकती हैं, जैसे बजट सीमाएँ, संसाधन की कमी, या समयबद्ध सीमाएँ।
7. **पूर्वधारणाएँ (Assumptions)** — आवश्यकता-संग्रह प्रक्रिया के दौरान की गई किसी भी पूर्वधारणा का उल्लेख करें। स्पष्टता और पारदर्शिता के लिए यह महत्वपूर्ण होता है।
8. **निर्भरता (Dependencies)** — अन्य परियोजनाओं, प्रणालियों या बाहरी कारकों पर किसी भी प्रकार की निर्भरता को रेखांकित करें जो परियोजना की सफलता को प्रभावित कर सकती है।
9. **जोखिम और शमन (Risk and Mitigations)** — परियोजना को प्रभावित करने वाले संभावित जोखिमों की पहचान करें और उन्हें कम करने के लिए कार्यनीतियाँ प्रस्तुत करें।
10. **स्वीकृति मानदंड (Acceptance Criteria)** — परियोजना को सफलतापूर्वक पूर्ण माना जाएगा या नहीं, इसका निर्धारण करने हेतु जिन मापदंडों का उपयोग किया जाएगा, उनका स्पष्ट वर्णन करें। ये मापदंड विशिष्ट, मापनीय और प्राप्त करने योग्य होने चाहिए।
11. **प्राथमिकता निर्धारण (Prioritisation)** — यदि आवश्यकताएँ परस्पर विरोधी हैं या संसाधन सीमित हैं, तो यह स्पष्ट करें कि कौन-सी आवश्यकताएँ परियोजना की सफलता के लिए सर्वाधिक महत्वपूर्ण हैं।
12. **अनुमोदन (Sign-Off)** — एक ऐसा खंड शामिल करें जिसमें हितधारक दस्तावेज़ की समीक्षा कर सकें और अपनी स्वीकृति दे सकें। अनुमोदन यह दर्शाता है कि वे प्रलेखित आवश्यकताओं से सहमत हैं।
13. **परिवर्तन नियंत्रण (Change Control)** — परियोजना प्रारंभ हो जाने के बाद आवश्यकताओं में परिवर्तन को किस प्रकार संभाला जाएगा, इसका विवरण दें। यह कार्य की सीमा के अनियंत्रित विस्तार को रोकने और परिवर्तनों का उचित प्रलेखन सुनिश्चित करने में सहायक होता है।

अभ्यास 5.1

- पाँच वेब डिज़ाइन विनिर्देशों की सूची बनाइए।
- व्यावसाय आवश्यकता विनिर्देश (BRS) के प्रमुख घटकों की सूची बनाइए।

एक सुव्यवस्थित व्यावसाय आवश्यकता विनिर्देश (BRS) तैयार करना अत्यंत आवश्यक है जिससे यह सुनिश्चित किया जा सके कि परियोजना व्यावसाय की आवश्यकताओं और उद्देश्यों के अनुरूप है। यह एक ऐसा **संचार उपकरण** के रूप में कार्य करता है जो व्यावसायिक हितधारकों और विकास दल के मध्य की दूरी को पाटता है, जिससे परियोजना की सफलता की संभावनाएँ बढ़ जाती हैं।

5.3 व्यावसाय आवश्यकता दस्तावेज़ (Business Requirement Document - BRD) के घटक

व्यावसाय आवश्यकता दस्तावेज़ (BRD) किसी परियोजना या पहल के आरंभिक चरणों में एक महत्वपूर्ण दस्तावेज़ होता है, विशेषतः सॉफ़्टवेयर विकास या प्रणाली कार्यान्वयन के संदर्भ में। यह परियोजना के उद्देश्यों, परिव्याप्ति और विनिर्देशों का **व्यावसायिक परिप्रेक्ष्य** से व्यापक मार्गदर्शन प्रदान करता है। एक BRD में सामान्यतः निम्नलिखित अनिवार्य तत्व (एलिमेंट) शामिल होते हैं—

1. **शीर्षक पृष्ठ (Title Page)**
   - परियोजना का शीर्षक
   - निर्माण की तिथि
   - संस्करण संख्या
   - लेखक (लेखकों) के नाम
   - संपर्क जानकारी
2. **अनुक्रमणिका (Table of Contents)** — अनुभागों और उप-अनुभागों की सूची जिनके साथ पृष्ठ संख्या दी जाती है, जिससे नेविगेशन सरल हो।
3. **कार्यकारी सारांश (Executive Summary)** — परियोजना का उच्च स्तरीय अवलोकन, जिसमें उद्देश्य, लक्ष्य और प्रमुख डिलिवरेबल्स सम्मिलित हों। यह खंड BRD की संक्षिप्त रूपरेखा प्रदान करता है।
4. **परियोजना पृष्ठभूमि (Project Background)** — उस व्यावसायिक समस्या या अवसर का वर्णन जिसे यह परियोजना संबोधित करती है। इसमें कोई भी प्रासंगिक ऐतिहासिक जानकारी या संदर्भ भी शामिल किया जाता है।
5. **उद्देश्य और लक्ष्य (Objectives and Goals)** — यह स्पष्ट और विशिष्ट कथन होते हैं जो बताते हैं कि परियोजना क्या प्राप्त करना चाहती है और उसकी सफलता को किस प्रकार मापा जाएगा।
6. **परिव्याप्ति और सीमाएँ (Scope and Boundaries)** — परियोजना में क्या सम्मिलित है (इन-स्कोप) और क्या नहीं (आउट-ऑफ-स्कोप), इसका स्पष्ट निर्धारण। साथ ही, परियोजना की सीमाएँ या प्रतिबंध भी परिभाषित किए जाते हैं।
7. **हितधारक विश्लेषण (Stakeholder Analysis)** — प्रमुख हितधारकों की पहचान की जाती है, उनकी भूमिकाओं और जिम्मेदारियों सहित, साथ ही उनकी अपेक्षाओं और चिंताओं का अवलोकन भी किया जाता है।
8. **कार्यक्षमता आवश्यकताएँ (Functional Requirements)** — प्रणाली या परियोजना को कौन-सी कार्यक्षमताएँ प्रदान करनी चाहिए, इसका विस्तृत विवरण। इसमें उपयोग मामलों, प्रयोक्ता कथाओं या परिदृश्यों को शामिल किया जाता है जो यह दर्शाते हैं कि प्रणाली का उपयोग कैसे किया जाएगा, साथ ही व्यापार नियम और तर्क भी वर्णित होते हैं।
9. **गैर-कार्यात्मक आवश्यकताएँ (Non-Functional Requirements)**
   - प्रदर्शन अपेक्षाएँ (जैसे प्रतिक्रिया समय, थ्रूपुट)।
   - सुरक्षा आवश्यकताएँ।
   - विस्तारक्षमता और विश्वसनीयता की आवश्यकताएँ।
   - अनुपालन और नियामक आवश्यकताएँ।
10. **डेटा आवश्यकताएँ (Data Requirements)**
   - डेटा स्रोतों का विवरण।
   - डेटा स्वरूप और संरचनाएँ।
   - डेटा संग्रहण और प्रबंधन आवश्यकताएँ।

11. **प्रयोक्ता इंटरफ़ेस आवश्यकताएँ (User Interface Requirements)**
    • डिज़ाइन और लेआउट विनिर्देश।
    • नेविगेशन और प्रयोक्ता संवाद दिशा-निर्देश।
    • यदि उपलब्ध हों तो प्रोटोटाइप या वायरफ्रेम।
12. **सिस्टम इंटरफ़ेस (System Interfaces)**
    • अन्य प्रणालियों या घटकों के साथ एकीकरण बिंदु।
    • संचार प्रोटोकॉल और डेटा स्वरूप।
13. **परीक्षण और गुणवत्ता आश्वासन (Testing and Quality Assurance)**
    • परीक्षण परिदृश्य, परीक्षण मामले और मानदंड।
    • गुणवत्ता मानक और परीक्षण कार्यप्रणालियाँ।
14. **परियोजना समयरेखा (Project Timeline)** — परियोजना की उच्च स्तरीय समय-सारणी या समयरेखा जिसमें उपलब्धियां और निर्भरता सम्मिलित हों।
15. **बजट और संसाधन आवश्यकताएँ (Budget and Resource Requirements)** — परियोजना की अनुमानित लागत, संसाधन आबंटन जिसमें मानव संसाधन और उपकरण शामिल हैं।
16. **जोखिम विश्लेषण (Risk Analysis)** — संभावित जोखिमों की पहचान और उन्हें कम करने हेतु कार्यनीतियाँ, साथ ही वैकल्पिक योजना।
17. **परिवर्तन प्रबंधन और प्रशिक्षण (Change Management and Training)** — परियोजना के परिणामस्वरूप होने वाले परिवर्तनों को लागू और प्रबंधित करने की योजना, साथ ही वास्तविक प्रयोक्ताओं और समर्थन कर्मचारियों हेतु प्रशिक्षण आवश्यकताएँ।

व्यावसाय आवश्यकता दस्तावेज़ एक **लाइव दस्तावेज़** होता है, जो परियोजना की प्रगति के साथ-साथ और नई जानकारी प्राप्त होने पर अद्यतन किया जा सकता है। यह इसे सुनिश्चित करने में महत्वपूर्ण भूमिका निभाता है कि सभी हितधारकों के मध्य परियोजना के उद्देश्यों और आवश्यकताओं को लेकर एक साझा समझ बनी रहे।

5.4 व्यावसायिक आवश्यकता के दस्तावेज़ (Business Requirements Document) के लाभ

व्यावसायिक आवश्यकता के दस्तावेज़ (BRD) परियोजना प्रबंधन और सॉफ्टवेयर विकास के क्षेत्र में एक अत्यंत महत्त्वपूर्ण दस्तावेज़ है, जो किसी परियोजना की आवश्यकताओं और अपेक्षाओं की रूपरेखा प्रस्तुत करता है। व्यावसायिक आवश्यकता के दस्तावेज़ तैयार करने और उपयोग करने के कुछ लाभ निम्नलिखित हैं—

1. यह किसी परियोजना के व्यावसायिक उद्देश्यों और आवश्यकताओं को स्पष्ट रूप से परिभाषित करने में सहायक होता है। यह सुनिश्चित करता है कि परियोजना के उद्देश्य और लक्ष्य के संबंध में सभी हितधारक एकमत हों।
2. यह परियोजना के कार्यक्षेत्र (scope) को परिभाषित करने में सहायता करता है, जो परियोजना प्रबंधकों के लिए संसाधनों, समयसीमा और बजट की प्रभावी योजना हेतु अत्यंत आवश्यक है।
3. आवश्यकताओं को समग्र रूप से दस्तावेज बना कर व्यावसायिक आवश्यकता के दस्तावेज़ों से परियोजना के प्रारंभिक चरण में ही संभावित जोखिमों और चुनौतियों की पहचान करने में सहायता मिलती है। इससे सक्रिय रूप से जोखिम प्रबंधन किया जा सकता है।
4. व्यावसायिक आवश्यकता के दस्तावेज़ व्यापार हितधारकों और परियोजना टीम के बीच एक संप्रेषण उपकरण के रूप में कार्य करता है। यह तकनीकी शब्दावली और व्यावसायिक भाषा के बीच का अंतर मिटा देता है, जिससे दोनों पक्षों के लिए एक-दूसरे की आवश्यकताओं और बॉर्डर्स को समझना आसान हो जाता है।

5. परियोजना जीवनचक्र के दौरान व्यावसायिक आवश्यकता का दस्तावेज़ निर्णय-निर्माण के लिए एक संदर्भ बिंदु के रूप में कार्य करता है। यह दस्तावेजित आवश्यकताओं का हवाला देकर विवादों या मतभेदों को सुलझाने में सहायता कर सकता है।
6. जब परियोजना की आवश्यकताओं में परिवर्तन उत्पन्न होते हैं, तब व्यावसायिक आवश्यकता का दस्तावेज़ एक आधार रेखा (baseline) के रूप में कार्य करता है जिसके विरुद्ध प्रस्तावित परिवर्तनों का मूल्यांकन किया जा सकता है। यह कार्यक्षेत्र में अनियंत्रित विस्तार (scope creep) को नियंत्रित करने में सहायक होता है।
7. एक स्पष्ट रूप से परिभाषित व्यावसायिक आवश्यकता के दस्तावेज़ गुणवत्ता आश्वासन और परीक्षण प्रयासों के लिए मानदंड (benchmark) के रूप में कार्य करते हैं। परीक्षक इसका उपयोग यह सुनिश्चित करने के लिए कर सकते हैं कि अंतिम उत्पाद निर्दिष्ट आवश्यकताओं के अनुरूप हो।
8. आवश्यकताओं की स्पष्ट समझ होने से परियोजना प्रबंधक लागत का अधिक सटीक अनुमान लगा सकते हैं, जिससे लागत अधिकता (cost overruns) का जोखिम घटता है।
9. व्यावसायिक आवश्यकता दस्तावेज़ से कार्यों और गतिविधियों की बेहतर योजना और समय निर्धारण में सहायता मिलती है, जिससे समय प्रबंधन और परियोजना निष्पादन में सुधार होता है।
10. अंततः, जब दस्तावेज के रूप में व्यावसायिक आवश्यकताओं को पूरा किया जाता है, तो परियोजना के ग्राहक की अपेक्षाओं को पूरा करने और संतोष स्तर को बढ़ाने की संभावना अधिक होती है।
11. यह परियोजना की व्यावसायिक आवश्यकताओं का एक संरचित और दस्तावेजीकृत अभिलेख प्रदान करता है। यह दस्तावेज़ लेखा परीक्षण (audit), अनुपालन (compliance) और भविष्य में संदर्भ के लिए उपयोगी हो सकता है।
12. व्यावसायिक आवश्यकता का दस्तावेज़ परियोजना में बाद में जुड़ने वाले नए टीम सदस्य अथवा हितधारकों के लिए एक उपयोगी संसाधन सिद्ध हो सकता है। यह परियोजना के लक्ष्यों और आवश्यकताओं का एक समग्र अवलोकन प्रदान करता है।

सारांश रूप में, व्यावसायिक आवश्यकता का दस्तावेज़ (BRD) परियोजना सफलता सुनिश्चित करने में एक केंद्रीय भूमिका निभाता है, क्योंकि यह प्रभावी संप्रेषण को संभव बनाता है, कार्यक्षेत्र को नियंत्रित करता है, जोखिम को कम करता है और परियोजना के संपूर्ण जीवनचक्र में निर्णय-निर्माण एवं गुणवत्ता आश्वासन हेतु आधार प्रदान करता है।

अभ्यास 5.2

- व्यावसायिक आवश्यकता के दस्तावेज़ (बीआरडी) के **किसी भी पाँच लाभ** सूचीबद्ध कीजिए।
- व्यावसायिक आवश्यकता के दस्तावेज़ में सामान्यतः शामिल किए जाने वाले **किसी भी पाँच आवश्यक तत्वों** की सूची तैयार कीजिए।

5.5 व्यावसायिक आवश्यकता के दस्तावेज़ तैयार करने की चरणबद्ध प्रक्रिया

व्यावसायिक आवश्यकता के दस्तावेज़ (बीआरडी) एक औपचारिक दस्तावेज़ होता है जो किसी विशिष्ट परियोजना या पहल के लिए व्यावसायिक आवश्यकताओं को प्रस्तुत करता है। यह परियोजना नियोजन की नींव के रूप में कार्य करता है और यह सुनिश्चित करता है कि इसमें शामिल सभी पक्षों को यह स्पष्ट रूप से ज्ञात हो कि क्या प्राप्त करना है। व्यावसायिक आवश्यकता के दस्तावेज़ तैयार करने की चरणबद्ध प्रक्रिया निम्नलिखित है—

1. सबसे पहले व्यावसायिक आवश्यकता के दस्तावेज़ का उद्देश्य स्पष्ट रूप से लिखें। यह परियोजना किस समस्या या अवसर को संबोधित करना चाहता है?
2. परियोजना के कार्यक्षेत्र (scope) को परिभाषित करें। परियोजना में क्या सम्मिलित है और क्या नहीं? इसकी सीमाएँ क्या हैं?
3. उन सभी हितधारकों की पहचान करें जो परियोजना में सम्मिलित होंगे या जिन पर इसका प्रभाव पड़ेगा। इसमें आंतरिक एवं बाह्य पक्ष दोनों सम्मिलित होते हैं।

4. हितधारकों के साथ मिलकर उनकी आवश्यकताओं को एकत्रित करें और दस्तावेजित करें। ये आवश्यकताएँ **विशिष्ट, मापनीय, प्राप्त करने योग्य, प्रासंगिक और समयबद्ध (SMART)** होनी चाहिए।
5. सभी आवश्यकताएँ समान महत्त्व की नहीं होतीं। उनकी प्राथमिकता इस आधार पर तय करें कि वे परियोजना की सफलता के लिए कितनी महत्त्वपूर्ण हैं।
6. आवश्यकताओं को एक संगठित अवसंरचना में व्यवस्थित करें। आप एक तालिका या स्प्रेडशीट का उपयोग करके एक मैट्रिक्स तैयार कर सकते हैं जिसमें आवश्यकताओं की आईडी, विवरण, प्राथमिकता, स्रोत (जिसने यह आवश्यकता बताई) और स्थिति सम्मिलित हो।
7. प्रत्येक उच्च-स्तरीय आवश्यकता को विस्तारपूर्वक क्रियात्मक आवश्यकताओं (functional requirements) में विभाजित करें। ये यह दर्शाएँगी कि प्रणाली या समाधान को क्या करना है।
8. गैर-क्रियात्मक आवश्यकताएँ (non-functional requirements) प्रणाली के गुणात्मक पक्षों का विवरण देती हैं, जैसे—प्रदर्शन, सुरक्षा, विस्तारशीलता और प्रयोज्यता। इन्हें स्पष्ट रूप से लिखें।
9. हितधारकों के साथ व्यावसायिक आवश्यकता के दस्तावेज़ की समीक्षा करें ताकि यह सुनिश्चित किया जा सके कि यह उनकी आवश्यकताओं और अपेक्षाओं का सही प्रतिनिधित्व करता है। जाँचें कि आवश्यकताएँ पूर्ण, संगत और व्यवहार्य हैं या नहीं।
10. जहाँ लागू हो, प्रयोक्ता के दृष्टिकोण से आवश्यकताओं की पूर्ति को दर्शाने के लिए प्रयोक्ता परिदृश्य (use cases) या प्रयोक्ता कहानियाँ (user stories) तैयार करें।
11. जटिल आवश्यकताओं या प्रक्रियाओं की समझ को बेहतर बनाने के लिए आरेखों, फ्लोचार्ट्स या अन्य दृश्य चित्रों का उपयोग करें।
12. प्रत्येक आवश्यकता के लिए स्वीकृति मानदंड (acceptance criteria) निर्दिष्ट करें, जिन्हें पूर्ण किए जाने पर ही आवश्यकता को सफलतापूर्वक लागू माना जाएगा।
13. परियोजना के दौरान आवश्यकताओं में परिवर्तन प्रबंधन की प्रक्रिया को स्पष्ट रूप से परिभाषित करें। इससे दस्तावेज़ की अखंडता बनाए रखने में सहायता मिलती है।
14. जब व्यावसायिक आवश्यकता के दस्तावेज़ पूर्ण हो जाएं और सभी हितधारक सहमत हों, तब प्रमुख हितधारकों से औपचारिक अनुमोदन (sign-off) प्राप्त करें।
15. अनुमोदित व्यावसायिक आवश्यकता के दस्तावेज़ को सभी संबंधित परियोजना टीम सदस्यों और हितधारकों को वितरित करें। यह सुनिश्चित करें कि सभी इसकी विषयवस्तु से अवगत हैं।
16. व्यावसायिक आवश्यकता का दस्तावेज़ परियोजना नियोजन की नींव के रूप में कार्य करता है, जिसमें परियोजना योजना, बजट और समयरेखा का निर्माण सम्मिलित है। परियोजना के दौरान जैसे-जैसे आवश्यकताएँ परिवर्तित हों या विकसित हों, व्यावसायिक आवश्यकता के दस्तावेज़ को अद्यतन करते रहें।

व्यावसायिक आवश्यकता के दस्तावेज़ का प्रारूप और संरचना संगठनात्मक मानकों और परियोजना की विशिष्ट आवश्यकताओं के अनुसार भिन्न हो सकता है। मुख्य बात यह है कि यह दस्तावेज़ परियोजना में सम्मिलित सभी पक्षों के लिए व्यावसायिक आवश्यकताओं को प्रभावी तरीके से संप्रेषित कर सके।

5.6 प्रयोक्ता आवश्यकता विनिर्देश (User Requirement Specification – URS) तैयार करने की चरणबद्ध प्रक्रिया

प्रयोक्ता आवश्यकता विनिर्देश (User Requirement Specification – URS) एक ऐसा दस्तावेज़ होता है जिसमें किसी प्रणाली या उत्पाद की क्रियात्मक (functional) और अक्रियात्मक (non-functional) आवश्यकताएँ वास्तविक प्रयोक्ता (end-user) के दृष्टिकोण से प्रस्तुत की जाती हैं। यह डेवलपर्स और संबंधित पक्षों के लिए एक रूपरेखा (blueprint) का कार्य करता है जिससे वे यह समझ सकें कि प्रयोक्ता को उस प्रणाली से क्या अपेक्षाएँ और आवश्यकताएँ हैं। प्रयोक्ता आवश्यकता विनिर्देश तैयार करने के चरण इस प्रकार हैं—

1. प्रयोक्ता आवश्यकता विनिर्देश दस्तावेज़ का उद्देश्य स्पष्ट रूप से लिखें। उस परियोजना या प्रणाली की बॉर्डर्स और कार्यक्षेत्र को परिभाषित करें जिसका प्रलेखन किया जाना है। अपेक्षित परिणाम क्या है और प्रमुख उद्देश्य क्या हैं?
2. उस परियोजना से जुड़े सभी हितधारकों (stakeholders) की पहचान करें। इनमें वास्तविक प्रयोक्ता , ग्राहक, परियोजना प्रबंधक, सॉफ्टवेयर डेवलपर, गुणवत्ता आश्वासन टीम और नियामक संस्थाएँ शामिल हो सकती हैं।
3. विभिन्न विधियों के माध्यम से प्रयोक्ता की आवश्यकताएँ एकत्र करें, जैसे साक्षात्कार, सर्वेक्षण, अवलोकन, तथा वर्तमान प्रणालियों से प्राप्त सुझाव (यदि लागू हों)। यह सुनिश्चित करें कि आप विभिन्न प्रकार के प्रयोक्ताओं को शामिल करें ताकि विविध दृष्टिकोणों को सम्मिलित किया जा सके।
4. एकत्र की गई आवश्यकताओं को स्पष्टता के लिए श्रेणियों में विभाजित करें। सामान्य श्रेणियों में क्रियात्मक आवश्यकताएँ (जैसे— प्रणाली को क्या करना चाहिए) और अक्रियात्मक आवश्यकताएँ (जैसे— कार्य निष्पादन, सुरक्षा, उपयोग में सहजता आदि) होती हैं।
5. प्रत्येक आवश्यकता के लिए निम्नलिखित जानकारी प्रदान करें—
   क) संदर्भ के लिए एक विशिष्ट पहचान संख्या या कोड
   ख) आवश्यकता का स्पष्ट और संक्षिप्त विवरण
   ग) उस आवश्यकता का स्रोत (जैसे— प्रयोक्ता साक्षात्कार, नियामक दिशा-निर्देश)
   घ) आवश्यकता की प्राथमिकता या महत्त्व (जैसे— अत्यंत आवश्यक, उच्च, मध्यम, निम्न)
   ङ) अन्य आवश्यकताओं पर निर्भरता (यदि हो)
6. प्रत्येक आवश्यकता के लिए स्वीकृति मानदंड (acceptance criteria) परिभाषित करें। ये वे शर्तें होती हैं जिन्हें उस आवश्यकता को पूर्ण मानने के लिए पूरा करना अनिवार्य होता है। स्वीकृति मानदंड आवश्यकताओं को परीक्षण योग्य और मापनीय बनाते हैं।
7. प्रारूप प्रयोक्ता आवश्यकता विनिर्देश के दस्तावेज़ को हितधारकों और विषय विशेषज्ञों के साथ साझा करें ताकि उसकी समीक्षा और पुष्टि की जा सके। यह सुनिश्चित करें कि सभी आवश्यकताएँ सटीक, पूर्ण और परियोजना के लक्ष्यों के अनुरूप हों।
8. यदि प्रणाली में डेटा संग्रहण, प्रक्रिया या रिपोर्टिंग सम्मिलित है, तो डेटा आवश्यकताओं और रिपोर्टिंग स्वरूपों को विस्तारपूर्वक उल्लेख करें।
9. प्रत्येक आवश्यकता को उसके स्रोत, स्वीकृति मानदंड और संबंधित परीक्षण मामलों से जोड़ने के लिए एक ट्रेसेबिलिटी मैट्रिक्स (traceability matrix) तैयार करें। यह सुनिश्चित करने में मदद करता है कि सभी आवश्यकताओं का समुचित रूप से ध्यान रखा गया है।
10. सभी संगत हितधारकों के साथ प्रयोक्ता आवश्यकता विनिर्देश की अंतिम समीक्षा करें। जब सभी सहमत हों, तो प्रमुख हितधारकों से औपचारिक स्वीकृति या हस्ताक्षर प्राप्त करें।
11. प्रयोक्ता आवश्यकता बिनिर्देश में समय के साथ होने वाले परिवर्तनों का लेखा-जोखा रखें। संशोधनों और अद्यतनों को दर्ज करने के लिए एक संस्करण नियंत्रण प्रणाली बनाए रखें।
12. परियोजना की जीवन-चक्र अवधि के दौरान, प्रयोक्ता आवश्यकता विनिर्देश को संदर्भ दस्तावेज़ के रूप में उपयोग करें ताकि डिज़ाइन, विकास, परीक्षण और प्रमाणीकरण प्रयासों को दिशा दी जा सके।

जैसे-जैसे परियोजना आगे बढ़ती है और नई जानकारियाँ प्राप्त होती हैं, आवश्यकताओं या प्राथमिकताओं में हुए परिवर्तनों को प्रतिबिंबित करने के लिए प्रयोक्ता आवश्यकता विनिर्देश को अद्यतन करने के लिए तैयार रहें। एक सुव्यवस्थित और दस्तावेजीकृत प्रयोक्ता आवश्यकता विनिर्देश परियोजना की सफलता के लिए अत्यंत महत्वपूर्ण होता है, क्योंकि यह सुनिश्चित करता है कि अंतिम उत्पाद या प्रणाली प्रयोक्ताओं की आवश्यकताओं और अपेक्षाओं के अनुरूप हो।

5.7 सॉफ़्टवेयर आवश्यकता विनिर्देश (Software Requirement Specification – SRS)

सॉफ़्टवेयर आवश्यकता विनिर्देश (Software Requirement Specification – SRS) एक दस्तावेज़ होता है जिसमें सॉफ़्टवेयर प्रणाली की विस्तृत क्रियात्मक और अक्रियात्मक आवश्यकताएँ वर्णित होती हैं। यह सॉफ़्टवेयर विकास टीम और ग्राहक या हितधारकों के बीच एक अनुबंध के रूप में कार्य करता है, जिससे यह सुनिश्चित हो कि दोनों पक्षों को सॉफ्टवेयर से अपेक्षित कार्यों की स्पष्ट समझ हो।

क्या आप जानते हैं?

- सॉफ़्टवेयर आवश्यकता विनिर्देश ग्राहक और ठेकेदारों के बीच हुए समझौते के आधार पर तैयार किया जाता है।

सॉफ़्टवेयर आवश्यकता विनिर्देश में सामान्यतः निम्नलिखित बातें सम्मिलित होती हैं—

1. **परिचय—**
   उद्देश्य— दस्तावेज़ और सॉफ्टवेयर परियोजना का उद्देश्य वर्णित करें।
   परिसीमा (Scope)— सॉफ्टवेयर की सीमाएँ निर्धारित करें कि यह क्या करेगा और क्या नहीं करेगा।
   दस्तावेज़ सम्मेल (Document Conventions)— दस्तावेज़ में प्रयुक्त किसी भी मानक या परंपराओं को स्पष्ट करें।
2. **समग्र विवरण (Overall Description)—**
   उत्पाद परिप्रेक्ष्य (Product Perspective)— सॉफ़्टवेयर बड़े तंत्र या पारिस्थितिकी तंत्र में किस प्रकार अनुकूल बैठता है, यह बताएं।
   उत्पाद कार्य (Product Functions)— सॉफ़्टवेयर के प्रमुख कार्यों और विशेषताओं की सूची बनाकर उनका विवरण दें।
   प्रयोक्ता वर्ग और विशेषताएँ (User Classes and Characteristics)— विभिन्न प्रकार के प्रयोक्ताओं और उनकी विशेषताओं की पहचान करें।
   प्रचालन वातावरण (Operating Environment)— वह हार्डवेयर, सॉफ़्टवेयर और नेटवर्क वातावरण बताएं जिसमें सॉफ़्टवेयर कार्य करेगा।
   डिज़ाइन और कार्यान्वयन प्रतिबंध (Design and Implementation Constraints)— डिज़ाइन या कार्यान्वयन को प्रभावित करने वाले किसी भी प्रतिबंध की व्याख्या करें।
3. **विशिष्ट आवश्यकताएँ (Specific Requirements)—**
   क्रियात्मक आवश्यकताएँ (Functional Requirements)— वे विशेष कार्य और क्षमताएँ जिन्हें सॉफ़्टवेयर को प्रदान करना आवश्यक है।
   अक्रियात्मक आवश्यकताएँ (Non-Functional Requirements)— प्रदर्शन, विश्वसनीयता, सुरक्षा, उपयोगिता, और स्केलेबिलिटी जैसे पहलुओं को परिभाषित करें।
   बाह्य इंटरफ़ेस आवश्यकताएँ (External Interface Requirements)— सॉफ़्टवेयर अन्य प्रणालियों या इंटरफेस के साथ किस प्रकार संपर्क करेगा, यह बताएं।
   प्रयोक्ता इंटरफ़ेस आवश्यकताएँ (User Interface Requirements)— प्रयोक्ता इंटरफ़ेस की डिज़ाइन और व्यवहार को स्पष्ट करें।
   प्रणाली विशेषताएँ (System Features)— सॉफ़्टवेयर की अतिरिक्त विशेषताओं या घटकों का वर्णन करें।
4. **प्रदर्शन आवश्यकताएँ (Performance Requirements)—**
   प्रतिक्रिया समय (Response Time)— विभिन्न कार्यों के लिए स्वीकार्य प्रतिक्रिया समय निर्दिष्ट करें।
   थ्रूपुट (Throughput)— प्रणाली की अपेक्षित प्रसंस्करण क्षमता को परिभाषित करें।
   स्केलेबिलिटी (Scalability)— यह बताएं कि बढ़ते लोड को प्रणाली कैसे संभालेगी।

5. **डिज़ाइन प्रतिबंध (Design Constraints)—**
   मानक अनुरूपता (Standards Compliance)— ऐसे किसी भी औद्योगिक या नियामक मानक की पहचान करें जिसका पालन अनिवार्य है।
   हार्डवेयर सीमाएँ (Hardware Limitations)— किसी भी हार्डवेयर बॉर्डर्स या आवश्यकताओं को निर्दिष्ट करें।
   सॉफ़्टवेयर सीमाएँ (Software Limitations)— किसी भी सॉफ़्टवेयर या प्लेटफ़ॉर्म बॉर्डर्स को स्पष्ट करें।
6. **गुणवत्ता विशेषताएँ (Quality Attributes)—**
   विश्वसनीयता (Reliability)— सामान्य और असामान्य स्थितियों में सॉफ़्टवेयर का व्यवहार कैसा होना चाहिए, यह बताएं।
   सुरक्षा (Security)— सुरक्षा आवश्यकताओं और अभिगम नियंत्रण को परिभाषित करें।
   प्रयोगिता (Usability)— उपयोग में सहजता और प्रयोक्ता अनुभव आवश्यकताओं को स्पष्ट करें।
   अनुरक्षणीयता (Maintainability)— यह स्पष्ट करें कि सॉफ़्टवेयर का अनुरक्षण और अद्यतन किस प्रकार किया जाएगा।
7. **अन्य आवश्यकताएँ (Other Requirements)—**
   कानूनी और नियामक आवश्यकताएँ (Legal and Regulatory Requirements)— किसी भी कानूनी या नियामकीय प्रतिबंधों का विवरण दें।
   प्रलेखन आवश्यकताएँ (Documentation Requirements)— वह प्रलेख बताएं जो सॉफ्टवेयर के साथ संलग्न होना चाहिए।
   प्रशिक्षण आवश्यकताएँ (Training Requirements)— प्रयोक्ताओं या व्यवस्थापकों को आवश्यक प्रशिक्षण का वर्णन करें।
   अतिरिक्त जानकारी जैसे आरेख, मॉक-अप या डेटा मॉडल सम्मिलित करें।
   दस्तावेज़ में प्रयुक्त तकनीकी शब्दों या संक्षिप्ताक्षरों की परिभाषा दें।
   सॉफ़्टवेयर आवश्यकता विनिर्देश में समय-समय पर किए गए परिवर्तनों का अभिलेख रखें।
   सॉफ़्टवेयर आवश्यकता विनिर्देश एक व्यापक और सुव्यवस्थित दस्तावेज़ होना चाहिए जो सॉफ्टवेयर की आवश्यकताओं की स्पष्ट और पूर्ण समझ प्रदान करे। यह विकास प्रक्रिया के दौरान एक संदर्भ बिंदु के रूप में कार्य करता है और यह सुनिश्चित करने में सहायक होता है कि अंतिम उत्पाद ग्राहक की अपेक्षाओं को पूर्ण रूप से पूरा करे।

5.8 एस.आर.एस. (SRS) दस्तावेज़ का महत्त्व

सॉफ़्टवेयर आवश्यकता विनिर्देश (Software Requirements Specification - SRS) दस्तावेज़ सॉफ़्टवेयर विकास प्रक्रिया में एक अत्यंत महत्त्वपूर्ण दस्तावेज़ होता है। यह संपूर्ण परियोजना के लिए एक रूपरेखा (ब्लूप्रिंट) के रूप में कार्य करता है और किसी सॉफ़्टवेयर प्रणाली के सफल विकास को सुनिश्चित करने में महत्वपूर्ण भूमिका निभाता है। निम्नलिखित कुछ प्रमुख कारण हैं जिनकी वजह से SRS दस्तावेज़ आवश्यक होता है—

1. सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ ग्राहक या हितधारकों और विकास दल के बीच एक सेतु के रूप में कार्य करता है। यह सॉफ़्टवेयर से अपेक्षित कार्यों और उसके व्यवहार को स्पष्ट रूप से परिभाषित और संप्रेषित करने में सहायक होता है।
2. यह सॉफ़्टवेयर प्रणाली की विशेषताओं, कार्यात्मकताओं और बाधाओं की रूपरेखा प्रस्तुत कर परियोजना की परिधि को परिभाषित करता है। इससे *स्कोप क्रीप* (scope creep) की समस्या से बचा जा सकता है, जिसमें विकास के दौरान अतिरिक्त आवश्यकताएँ जुड़ जाती हैं।
3. सॉफ़्टवेयर आवश्यकता विनिर्देश परियोजना के लिए आवश्यक समय, प्रयास और संसाधनों का अनुमान लगाने हेतु आधार प्रदान करता है। यह परियोजना नियोजन और बजट निर्धारण के लिए आवश्यक होता है।

4. विस्तृत आवश्यकताओं को निर्दिष्ट करके, सॉफ़्टवेयर आवश्यकता विनिर्देश गुणवत्ता आश्वासन और परीक्षण के लिए एक मानक (बेंचमार्क) के रूप में कार्य करता है। परीक्षक इसका उपयोग परीक्षण मामलों (test cases) के निर्माण और सॉफ़्टवेयर के निर्दिष्ट मानदंडों की पूर्ति की पुष्टि के लिए करते हैं।
5. यह परियोजना के आरंभिक चरण में संभावित जोखिमों और चुनौतियों की पहचान में सहायक होता है। आवश्यकताओं और बाधाओं को प्रलेखित कर सॉफ़्टवेयर आवश्यकता विनिर्देश टीमों को इन समस्याओं का पूर्वानुमान लगाकर समाधान करने में सक्षम बनाता है।
6. विकासकर्ता और डिज़ाइनकर्ता सॉफ़्टवेयर की डिज़ाइन और कार्यान्वयन के लिए सॉफ़्टवेयर आवश्यकता विनिर्देश को संदर्भ दस्तावेज़ के रूप में उपयोग करते हैं। यह उन्हें अपेक्षित कार्यक्षमता और व्यवहार की स्पष्ट समझ प्रदान करता है।
7. यदि विकास प्रक्रिया के दौरान परिवर्तन की माँग की जाती है, तो सॉफ़्टवेयर आवश्यकता विनिर्देश का उपयोग उन परिवर्तनों के प्रभाव का मूल्यांकन करने के लिए एक आधाररेखा (बेसलाइन) के रूप में किया जा सकता है। इससे परिवर्तन अनुरोधों का प्रबंधन प्रभावी रूप से किया जा सकता है।
8. एक सुव्यवस्थित सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ ग्राहक की अपेक्षाओं के प्रबंधन में सहायक होता है। ग्राहक दस्तावेज़ की समीक्षा करके यह सुनिश्चित कर सकते हैं कि उनकी आवश्यकताएँ सही रूप से दर्ज की गई हैं, जिससे गलतफहमी या असंतोष की संभावना कम हो जाती है।
9. कुछ विशिष्ट उद्योगों में सॉफ़्टवेयर को कानूनी और विनियामक आवश्यकताओं का पालन करना होता है। सॉफ़्टवेयर आवश्यकता विनिर्देश का उपयोग यह सुनिश्चित करने के लिए किया जा सकता है कि सॉफ़्टवेयर इन मानकों का अनुपालन करता है।
10. यह सॉफ़्टवेयर विकास जीवनचक्र के दौरान एक महत्वपूर्ण दस्तावेज़ के रूप में कार्य करता है। यह परियोजना की आवश्यकताओं का एक ऐतिहासिक अभिलेख (रिकॉर्ड) प्रदान करता है, जो रखरखाव, अद्यतन और भविष्य के संदर्भ में उपयोगी होता है।
11. सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ व्यापार विश्लेषक, विकासकर्ता, परीक्षक और परियोजना प्रबंधकों सहित विभिन्न हितधारकों के बीच सहयोग को प्रोत्साहित करता है। यह उनके प्रयासों को एक साझा उद्देश्य की ओर संरेखित करने में सहायता करता है।
12. आवश्यकताओं को स्पष्ट रूप से प्रलेखित करके सॉफ़्टवेयर आवश्यकता विनिर्देश अस्पष्टता को कम करता है और गलतफहमियों एवं भ्रम की संभावनाओं को न्यूनतम करता है, जो महंगे त्रुटियों का कारण बन सकते हैं।

सारांश के तौर पर सॉफ़्टवेयर आवश्यकताओं विनिर्देश (SRS) दस्तावेज़ किसी सॉफ़्टवेयर परियोजना की दिशा तय करने, हितधारकों की आवश्यकताओं की पूर्ति सुनिश्चित करने और परियोजना टीमों के बीच प्रभावी संप्रेषण एवं सहयोग को सुगम बनाने के लिए अत्यंत आवश्यक है। यह संपूर्ण सॉफ़्टवेयर विकास प्रक्रिया को आरंभ से अंतिम वितरण तक मार्गदर्शित करने वाला एक मूलभूत दस्तावेज़ होता है।

5.9 सॉफ़्टवेयर आवश्यकता विनिर्देश लेखन हेतु दिशा-निर्देश

सॉफ़्टवेयर आवश्यकताओं विनिर्देश (SRS) का लेखन सॉफ़्टवेयर विकास प्रक्रिया का एक महत्वपूर्ण चरण होता है। सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ परियोजना का एक रूपरेखा होता है, जिसमें यह विस्तृत रूप से बताया जाता है कि सॉफ़्टवेयर को क्या करना चाहिए, वह कैसे व्यवहार करेगा और क्या नहीं करेगा। एक प्रभावी सॉफ़्टवेयर आवश्यकता विनिर्देश तैयार करने के लिए निम्नलिखित दिशा-निर्देशों का पालन किया जाना चाहिए—

1. **परिचय** — दस्तावेज़ का एक संक्षिप्त परिचय दें, जिसमें उसका उद्देश्य, कार्यक्षेत्र और परियोजना से संबंधित कोई भी पृष्ठभूमि जानकारी शामिल हो।
2. **उद्देश्य** — सॉफ़्टवेयर का उद्देश्य स्पष्ट रूप से बताएं और वह किस समस्या का समाधान करेगा, यह निर्दिष्ट करें।

3. **कार्यक्षेत्र** — सॉफ़्टवेयर की बॉर्डर्स को परिभाषित करें। परियोजना में क्या शामिल है और क्या नहीं, इसे स्पष्ट रूप से निर्दिष्ट करें ताकि भविष्य में गलतफहमी न हो।

4. **कार्यात्मक आवश्यकताएँ**

   क) सॉफ़्टवेयर द्वारा प्रदत्त सभी कार्यों और विशेषताओं की सूची दें। भाषा सरल और स्पष्ट होनी चाहिए।
   ख) आवश्यकताओं को श्रेणियों या खंडों में विभाजित करें ताकि पठनीयता बढ़े।
   ग) प्रत्येक कार्य अथवा विशेषता हेतु इनपुट और आउटपुट निर्दिष्ट करें।
   घ) यदि कोई प्रयोक्ता भूमिकाएँ एवं उनके अधिकार हों, तो उनका उल्लेख करें।

5. **गैर-कार्यात्मक आवश्यकताएँ**

   क) प्रदर्शन, सुरक्षा, विस्तारशीलता (scalability), उपयोगिता (usability) जैसे पहलुओं का विवरण दें।
   ख) जहाँ संभव हो, मापने योग्य मानदंडों का उपयोग करें, जैसे प्रतिक्रिया समय, त्रुटि दर या सुरक्षा मानक।

6. **प्रयोक्ता अंतरफलक (User Interfaces)**

   क) प्रयोक्ता इंटरफेस का विवरण दें, जैसे मॉकअप, वायरफ्रेम या डिज़ाइन दिशानिर्देश (यदि उपलब्ध हों)।
   ख) प्रयोक्ता किस प्रकार प्रणाली से संवाद करेगा — नेविगेशन व उपयोग प्रवाह सहित — यह बताएं।

7. **डेटा आवश्यकताएँ**

   क) प्रणाली द्वारा प्रयुक्त डेटा का विवरण दें — डेटा प्रकार, स्रोत और भंडारण आवश्यकताएँ।
   ख) किसी भी डेटा प्रमाणीकरण या प्रसंस्करण नियमों का उल्लेख करें।

8. **मान्यताएँ और आश्रितताएँ (Dependencies)**

   क) आवश्यकताओं के संग्रहण के दौरान की गई किसी भी मान्यता (assumption) को स्पष्ट रूप से लिखें।
   ख) बाहरी प्रणालियाँ, लाइब्रेरी या अन्य घटक जिन पर सॉफ़्टवेयर निर्भर करता है, उनका उल्लेख करें।

9. **बाधाएँ (Constraints)** — विकास अथवा संचालन को प्रभावित करने वाली कोई भी बाधा, जैसे बजट, समय, या तकनीकी सीमाएँ।

10. **गुणवत्ता और परीक्षण आवश्यकताएँ** — गुणवत्ता सुनिश्चित करने हेतु परीक्षण मानदंडों का विवरण दें। परीक्षण विधियाँ, परीक्षण परिदृश्य (test scenarios) और स्वीकृति मानदंड स्पष्ट करें।

11. **दस्तावेज़ीकरण आवश्यकताएँ** — सॉफ़्टवेयर के साथ दिए जाने वाले दस्तावेज़ जैसे प्रयोक्ता पुस्तिका, प्रणाली दस्तावेज़ या एपीआई दस्तावेज़ का उल्लेख करें।

12. **परिवर्तन नियंत्रण (Change Control)** — विकास के दौरान आवश्यकताओं में परिवर्तन को नियंत्रित करने की प्रक्रिया को परिभाषित करें, जिसमें परिवर्तन अनुरोध और अनुमोदन प्रक्रिया शामिल हो।

13. **ट्रेसेबिलिटी (Traceability)** — आवश्यकताओं, डिज़ाइन दस्तावेज़ और परीक्षण मामलों के बीच ट्रेसेबिलिटी स्थापित करें ताकि यह सुनिश्चित हो सके कि सभी आवश्यकताएँ अंतिम उत्पाद में सम्मिलित हुई हैं।

14. **समीक्षा और अनुमोदन** — यह स्पष्ट करें कि सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ की समीक्षा और अनुमोदन कौन करेगा — हितधारक एवं विषय विशेषज्ञों सहित। अंतिम रूप देने से पूर्व हितधारकों के साथ दस्तावेज़ की समीक्षा और पुष्टि करें ताकि यह सुनिश्चित हो सके कि सभी आवश्यकताएँ सटीक, पूर्ण और स्पष्ट रूप से परिभाषित हैं।

सॉफ़्टवेयर आवश्यकता विनिर्देश को स्पष्ट और संक्षिप्त भाषा में लिखा जाना चाहिए, जिसमें ऐसी अस्पष्ट या अत्यधिक तकनीकी भाषा से बचा जाए जो गलतफहमी का कारण बन सकती है। सॉफ़्टवेयर आवश्यकता विनिर्देश एक *जीवंत दस्तावेज़* होता है जो परियोजना के दौरान विकसित होता रहता है, अतः यह आवश्यक है कि विकास प्रक्रिया के साथ-साथ संप्रेषण और प्रलेखन की अच्छी पद्धतियों का पालन किया जाए।

अभ्यास 5.3.

- सॉफ़्टवेयर आवश्यकता विनिर्देश लिखने के लिए कोई पाँच दिशा-निर्देश सूचीबद्ध करें।
- सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ के महत्व को दर्शाने के लिए कोई पाँच प्रमुख कारण सूचीबद्ध करें।

5.10 एस.आर.एस. (SRS) दस्तावेज़ तैयार करना

सॉफ्टवेयर विकास की प्रक्रिया में **सॉफ्टवेयर आवश्यकताएँ विनिर्देशन (Software Requirements Specification – SRS)** दस्तावेज़ तैयार करना एक महत्वपूर्ण चरण होता है। सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ में विकसित किए जाने वाले सॉफ्टवेयर की **कार्यात्मक (Functional)** और **गैर-कार्यात्मक (Non-functional)** आवश्यकताओं का वर्णन किया जाता है। यह दस्तावेज़ विकास दल और हितधारकों दोनों के लिए एक संदर्भ के रूप में कार्य करता है, जिससे यह सुनिश्चित होता है कि सॉफ्टवेयर से संबंधित अपेक्षाएँ स्पष्ट रूप से समझी जा सकें। एक सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ में सामान्यतः निम्नलिखित अनुभाग और जानकारी सम्मिलित होती है—

1. **परिचय —**
   - **उद्देश्य** — स्पष्ट करें कि यह सॉफ्टवेयर क्यों विकसित किया जा रहा है और यह किस समस्या का समाधान करना चाहता है।
   - **परिधि (Scope)** — परियोजना की सीमाएँ परिभाषित करें, जिसमें क्या शामिल है और क्या नहीं, यह स्पष्ट किया जाए।
   - **दस्तावेज़ सम्मेलन** — दस्तावेज़ में प्रयुक्त नामकरण या प्रारूपण (formatting) से संबंधित दिशा-निर्देश बताएं।
2. **समग्र विवरण (Overall Description)—**
   - **उत्पाद परिप्रेक्ष्य** — यदि लागू हो तो यह स्पष्ट करें कि यह सॉफ्टवेयर किसी बड़े सिस्टम का हिस्सा कैसे है।
   - **उत्पाद कार्य** — सॉफ्टवेयर की मुख्य कार्यक्षमताएँ या विशेषताएँ सूचीबद्ध करें।
   - **प्रयोक्ता वर्ग और विशेषताएँ** — प्रयोक्ताओं के प्रकार और उनकी भूमिकाओं की पहचान करें।
   - **संचालन आवश्यकताएँ** — हार्डवेयर, सॉफ्टवेयर और नेटवर्क से संबंधित आवश्यकताओं को स्पष्ट करें।
   - **डिज़ाइन और कार्यान्वयन सीमाएँ** — किसी भी तकनीकी विकल्प या नियामक आवश्यकता जैसी बॉर्डर्स का उल्लेख करें।
   - **मान्यताएँ और निर्भरताएँ** — आवश्यकताओं के एकत्रीकरण के दौरान की गई मान्यताओं और अन्य प्रणालियों/घटकों पर निर्भरता को दस्तावेज़ित करें।
3. **विशिष्ट आवश्यकताएँ (Specific Requirements)—**
   - **कार्यात्मक आवश्यकताएँ** — सॉफ्टवेयर की विशिष्ट कार्यों या सुविधाओं का विस्तृत विवरण दें, आमतौर पर प्रयोक्ता कहानियाँ (user stories), उपयोग प्रकरण (use cases) या आवश्यकताओं की सूची के रूप में।
   - **गैर-कार्यात्मक आवश्यकताएँ** — प्रदर्शन, सुरक्षा, प्रयोज्यता (usability) आदि से संबंधित आवश्यकताओं को शामिल करें।
   - **बाहरी इंटरफ़ेस आवश्यकताएँ** — सॉफ्टवेयर अन्य प्रणालियों के साथ कैसे संपर्क करेगा, जैसे एपीआई (APIs), डेटा प्रारूप, और प्रोटोकॉल।
   - **प्रयोक्ता इंटरफ़ेस आवश्यकताएँ** — प्रयोक्ता इंटरफ़ेस डिज़ाइन की जानकारी दें, जिसमें वॉयरफ्रेम्स या मॉक-अप्स (mock-ups) भी हो सकते हैं।
   - **प्रणाली विशेषताएँ** — त्रुटि प्रबंधन, लॉगिंग और रिपोर्टिंग जैसी अतिरिक्त प्रणाली-स्तरीय विशेषताओं का वर्णन करें।
   - **डेटा आवश्यकताएँ** — डेटा संग्रहण, डेटा प्रसंस्करण और डेटा प्रवाह की आवश्यकताओं का उल्लेख करें।

- **गुणवत्ता विशेषताएँ** — विश्वसनीयता (reliability), मापनीयता (scalability), अनुरक्षणीयता (maintainability) जैसी गुणवत्ता विशेषताओं को परिभाषित करें।
- **सीमाएँ** — विकास को प्रभावित करने वाली कोई भी सीमाएँ, जैसे बजट या समय की सीमाएँ, उल्लेख करें।
- **उपयोग प्रकरण (Use Cases)** — यह दर्शाने के लिए विस्तृत परिदृश्य दें कि प्रणाली का प्रयोग कैसे किया जाएगा।

4. **व्यावसायिक नियम (Business Rules)** — सॉफ्टवेयर को जिन व्यावसायिक नियमों का पालन करना आवश्यक है, उन्हें दस्तावेज़ित करें।
5. **परीक्षण और सत्यापन मानदंड** — यह परिभाषित करें कि सॉफ्टवेयर द्वारा निर्धारित आवश्यकताओं को पूरा करने की जाँच कैसे की जाएगी।
6. **स्वीकृति मानदंड (Acceptance Criteria)** — वे मानदंड सूचीबद्ध करें जिनके पूर्ण होने पर ग्राहक या हितधारक सॉफ्टवेयर को स्वीकार करेंगे।
7. **शब्दावली और संदर्भ** — दस्तावेज़ में प्रयुक्त संक्षिप्ताक्षरों और तकनीकी शब्दों की एक शब्दावली अवश्य दें ताकि साझा समझ बनी रहे। आवश्यक हो तो किसी बाहरी दस्तावेज़ या स्रोत की जानकारी भी दें। किसी भी अतिरिक्त जानकारी जैसे चित्र, आरेख, चार्ट, या अन्य दस्तावेज़ी सामग्री को भी शामिल करें।

सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ बनाते समय हितधारकों — जैसे वास्तविक प्रयोक्ताओं और विकासकर्ताओं — की सहभागिता अत्यंत आवश्यक है ताकि आवश्यकताएँ उनकी अपेक्षाओं का सही प्रतिनिधित्व कर सकें। इसके अतिरिक्त, इस दस्तावेज़ को समय-समय पर परियोजना की प्रगति के अनुसार अद्यतन किया जाना चाहिए ताकि नई जानकारियाँ या परिवर्तन दिखाई दे सकें।

सारांश

- वेब डिज़ाइन विनिर्देशों में बीआरएस, बीआरडी और सॉफ़्टवेयर आवश्यकता विनिर्देश जैसे विभिन्न दस्तावेज़ सम्मिलित होते हैं, जो परियोजना के उद्देश्य, परिधि और सॉफ्टवेयर आवश्यकताओं को परिभाषित करते हैं।
- वेबसाइट डिज़ाइन करते समय लक्षित दर्शक, उत्तरदायी डिज़ाइन, पहुँचयोग्यता और एसईओ जैसे घटकों पर विचार किया जाता है ताकि प्रयोक्ता सहभागिता प्रभावी हो।
- ब्राउज़र अनुकूलता, सुरक्षा और प्रदर्शन की निगरानी, वेब डिज़ाइन के आवश्यक पहलू हैं।
- बीआरएस और बीआरडी हितधारकों के बीच संप्रेषण उपकरण के रूप में कार्य करते हैं, जबकि एसआरएस सॉफ्टवेयर की कार्यात्मकता और गैर-कार्यात्मक आवश्यकताओं को रेखांकित करता है।
- प्रभावी एसआरएस लेखन में परिचय, उत्पाद विवरण, आवश्यकताएँ, सीमाएँ और सत्यापन मानदंड जैसे खंड शामिल होते हैं।
- स्पष्ट दस्तावेज़ीकरण यह सुनिश्चित करता है कि सॉफ्टवेयर परियोजनाएँ नियत पथ पर रहें, प्रयोक्ता आवश्यकताओं की पूर्ति करें और उद्योग मानकों का पालन करें।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. एक एसआरएस (सॉफ़्टवेयर आवश्यकता विनिर्देश) दस्तावेज़ का मुख्य उद्देश्य क्या होता है?
   (क) डिज़ाइन अवधारणा को प्रदर्शित करना
   (ख) व्यावसायिक आवश्यकताओं को रेखांकित करना
   (ग) वेब डिज़ाइन लेआउट को परिभाषित करना
   (घ) कंटेंट कार्यनीति को निर्दिष्ट करना

2. सॉफ़्टवेयर आवश्यकता विनिर्देश दस्तावेज़ का कौन-सा भाग उस हार्डवेयर और सॉफ्टवेयर वातावरण का वर्णन करता है जिसमें सॉफ्टवेयर संचालित होगा?
   (क) कार्यात्मक आवश्यकताएँ
   (ख) गैर-कार्यात्मक आवश्यकताएँ
   (ग) संचालन वातावरण
   (घ) प्रणाली विशेषताएँ

3. सॉफ्टवेयर विकास के संदर्भ में यूआरएस का पूर्ण रूप क्या है?
   (क) प्रयोक्ता आवश्यकता विनिर्देश (User Requirement Specification)
   (ख) यूनिवर्सल रिसोर्स स्कीमा (Universal Resource Schema)
   (ग) प्रयोक्ता पंजीकरण प्रणाली (User Registration System)
   (घ) यूनिफ़ॉर्म रिपोर्टिंग स्टैंडर्ड (Uniform Reporting Standard)

4. व्यावसायिक आवश्यकता विनिर्देश (बीआरएस) दस्तावेज़ का कौन-सा भाग संभावित जोखिमों और उन्हें कम करने की कार्यनीतियों की पहचान करता है?
   (क) कार्यात्मक आवश्यकताएँ
   (ख) जोखिम और निवारण
   (ग) परिधि और सीमाएँ
   (घ) स्वीकृति मानदंड

5. वेब डिज़ाइन प्रक्रिया में वेबसाइट के स्पष्ट उद्देश्य और लक्ष्य स्थापित करने का मुख्य उद्देश्य क्या है?
   (क) वेबसाइट को दृश्यात्मक रूप से आकर्षक बनाना
   (ख) डिज़ाइन और कार्यात्मक निर्णयों का मार्गदर्शन करना
   (ग) वेबसाइट लोडिंग गति बढ़ाना
   (घ) सही रंग योजना चुनना

6. व्यावसायिक आवश्यकता विनिर्देश दस्तावेज़ का कौन-सा भाग उन विशिष्ट कार्यों और विशेषताओं को रेखांकित करता है जो परियोजना या प्रणाली को व्यावसायिक लक्ष्यों को प्राप्त करने हेतु आवश्यक हैं?
   (क) सीमाएँ
   (ख) कार्यात्मक आवश्यकताएँ
   (ग) परिधि
   (घ) मान्यताएँ

7. वेब डिज़ाइन के संदर्भ में "उत्तरदायी डिज़ाइन (Responsive Design)" का क्या तात्पर्य है?
   (क) वेबसाइट को गाढ़े और जीवंत रंगों से डिज़ाइन करना
   (ख) यह सुनिश्चित करना कि वेबसाइट प्रयोक्ता की क्रियाओं पर शीघ्र प्रतिक्रिया दे
   (ग) वेबसाइट को विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार अनुकूल बनाना
   (घ) जटिल एनिमेशन के साथ वेबसाइट बनाना

8. वेब डिज़ाइन में एसईओ का पूर्ण रूप क्या है?
   (क) सर्च इंजन ऑर्डर
   (ख) साइट एन्हांसमेंट ऑप्टिमाइज़ेशन
   (ग) सुरक्षा और एन्क्रिप्शन संचालन
   (घ) सर्च इंजन ऑप्टिमाइज़ेशन

9. बीआरएस दस्तावेज़ का कौन-सा भाग प्रमुख हितधारकों और परियोजना में उनकी भूमिकाओं को रेखांकित करता है?
   (क) परिधि
   (ख) हितधारक
   (ग) सीमाएँ
   (घ) मान्यताएँ

10. वेब डिज़ाइन की पहुँच योग्यता के संदर्भ में डब्ल्यूसीएजी का पूर्ण रूप क्या है?
   (क) वेब कम्पैटिबिलिटी एंड ग्राफिक्स
   (ख) वेब कंटेंट एक्सेसिबिलिटी गाइडलाइंस
   (ग) वेबसाइट एंड कंटेंट एक्सेसिबिलिटी ग्रुप
   (घ) वेब कोडिंग एंड ग्राफिक्स

ख. रिक्त स्थान भरिए

1. एक एसआरएस दस्तावेज़ परियोजना की प्रारंभिक अवस्था में संभावित _____ और चुनौतियों की पहचान में सहायता करता है।
2. यूआरएस (प्रयोक्ता आवश्यकता विनिर्देश) एक ऐसा दस्तावेज़ है जो कार्यात्मक और गैर-कार्यात्मक आवश्यकताओं को _____ के दृष्टिकोण से परिभाषित करता है।
3. एसआरएस दस्तावेज़ का प्रदर्शन आवश्यकताएँ अनुभाग स्वीकृत _____ समय और प्रणाली की प्रसंस्करण क्षमता को परिभाषित करता है।
4. एक सुव्यवस्थित यूआरएस दस्तावेज़ डेवलपर्स और हितधारकों के लिए एक _____ के रूप में कार्य करता है।
5. एसआरएस दस्तावेज़ का _____ वातावरण अनुभाग उस हार्डवेयर, सॉफ्टवेयर और नेटवर्क परिवेश का विवरण प्रदान करता है जिसमें सॉफ्टवेयर कार्य करेगा।
6. _____ दस्तावेज़ वास्तविक प्रयोक्ता के दृष्टिकोण से आवश्यकताओं का उल्लेख करता है।
7. एसआरएस दस्तावेज़ में गैर-कार्यात्मक आवश्यकताएँ प्रदर्शन, _____ और उपयोगिता जैसे आयाम शामिल हैं।
8. बीआरएस दस्तावेज़ का _____ अनुभाग परियोजना के प्रमुख लक्ष्यों और उद्देश्यों को रेखांकित करता है।
9. एसआरएस दस्तावेज़ में _____ आवश्यकताएँ प्रदर्शन, सुरक्षा और मापनीयता जैसे पहलुओं का वर्णन करती हैं।
10. वेब डिज़ाइन में उत्तरदायी डिज़ाइन यह सुनिश्चित करता है कि वेबसाइट विभिन्न _____ और _____ के अनुसार अनुकूलित हो।

ग. सत्य/असत्य लिखिए

1. एसआरएस दस्तावेज़ छोटे स्तर के सॉफ़्टवेयर अनुप्रयोगों के विकास के लिए आवश्यक नहीं होता।
2. यूआरएस दस्तावेज़ मुख्यतः तकनीकी विनिर्देशों और कार्यान्वयन विवरणों पर केंद्रित होता है।
3. एसआरएस दस्तावेज़ में परिवर्तन नियंत्रण प्रक्रियाओं की आवश्यकता नहीं होती क्योंकि परिवर्तन को हतोत्साहित किया जाना चाहिए।

4. बीआरएस (व्यावसायिक आवश्यकता विनिर्देश) दस्तावेज़ एक लाइव दस्तावेज़ होता है जिसे परियोजना की प्रगति के अनुसार अद्यतन किया जाता है।
5. व्यावसायिक आवश्यकताओं को परिभाषित करते समय सीमाएँ, मान्यताएँ और निर्भरताएँ महत्वपूर्ण तत्व (एलिमेंट) होते हैं।
6. बीआरएस दस्तावेज़ स्थिर रहता है और परियोजना के जीवनचक्र में कभी अद्यतन नहीं किया जाता।
7. एक एसआरएस दस्तावेज़ को डेटा आवश्यकताओं को स्पष्ट रूप से निर्दिष्ट करना चाहिए, जिसमें डेटा प्रारूप, स्रोत और संग्रहण आवश्यकताएँ शामिल हों।
8. उत्तरदायी डिज़ाइन का प्राथमिक उद्देश्य वेबसाइट को रंगीन और एनिमेशन-युक्त बनाना होता है।
9. "स्कोप क्रीप (Scope Creep)" उस प्रक्रिया को कहा जाता है जिसमें परियोजना की बॉर्डर्स को सावधानीपूर्वक परिभाषित किया जाता है।
10. प्रयोक्ता प्रतिक्रिया संग्रहण और उस प्रतिक्रिया के आधार पर वेबसाइट सुधार वेब डिज़ाइन के महत्वपूर्ण पहलू नहीं हैं।

घ. लघु उत्तर प्रश्न

1. किसी वेब डिज़ाइन परियोजना में बीआरएस (व्यावसायिक आवश्यकता विनिर्देश)दस्तावेज़ का मुख्य उद्देश्य क्या है?
2. वेबसाइट डिज़ाइन करते समय लक्षित दर्शकों की पहचान क्यों महत्वपूर्ण होती है?
3. उत्तरदायी डिज़ाइन वेबसाइटों पर बेहतर प्रयोक्ता अनुभव में कैसे योगदान देता है?
4. एसआरएस दस्तावेज़ में गैर-कार्यात्मक आवश्यकताओं के कुछ उदाहरण क्या हैं?
5. वेब डिज़ाइन प्रक्रिया में उपयोगिता परीक्षण (usability testing) की क्या भूमिका होती है?
6. सॉफ़्टवेयर आवश्यकताओं के प्रबंधन में ट्रेसएबिलिटी मैट्रिक्स (traceability matrix) कैसे सहायता करता है?
7. एसआरएस दस्तावेज़ में "स्वीकृति मानदंड (Acceptance Criteria)" निर्धारित करने का क्या महत्त्व है?
8. वेब डिज़ाइन में विभिन्न वेब ब्राउज़रों के लिए संगतता परीक्षण करना क्यों आवश्यक है?
9. "स्कोप क्रीप (Scope Creep)" क्या है और सॉफ़्टवेयर विकास परियोजना में इसे रोकना क्यों महत्वपूर्ण है?
10. एसआरएस दस्तावेज़ के "समग्र विवरण (Overall Description)" अनुभाग में कौन-कौन से प्रमुख तत्व (एलिमेंट) शामिल होते हैं?

सत्र 6

प्रोग्रामिंग के लिए निम्न-स्तरीय और उच्च-स्तरीय डिज़ाइन (Low-level Design and High-level Design)

आइए मिलते हैं **अंकित** से, जो कंप्यूटर गेम्स का शौकीन बच्चा था। वह "प्रोग्रामिंग के लिए निम्न-स्तरीय और उच्च-स्तरीय डिज़ाइन" के बारे में जानना चाहता था। उसे यह समझ आया कि यह ठीक वैसे ही है जैसे LEGO ईंटों से कुछ बनाना।

निम्न-स्तरीय डिज़ाइन — यह छोटी LEGO ईंटों को बनाने जैसा है — जैसे कि पात्र कैसे चलेंगे और खेल में अंक कैसे जोड़े जाएँगे।

उच्च-स्तरीय डिज़ाइन — यह पूरे LEGO किले की योजना बनाने जैसा है — कहानी, लक्ष्य, और यह कि सब कुछ एक-दूसरे से कैसे जुड़ा होगा।

अंकित ने इन डिज़ाइनों के साथ एक शानदार खेल तैयार किया। इसमें एक शानदार कहानी थी, और पात्र बिल्कुल सही तरीके से चलते थे। अंकित ने सीखा कि गेम बनाना LEGO मास्टर बनने जैसा है। **निम्न-स्तरीय डिज़ाइन छोटे भागों के लिए होता है**, और **उच्च-स्तरीय डिज़ाइन पूरी रूपरेखा के लिए**। ठीक LEGO किला बनाने की तरह, उसने ऐसे अद्भुत गेम बनाए जिन्हें सबने पसंद किया, जैसा कि **चित्र 6.1** में दर्शाया गया है।

चित्र 6.1— LEGO किला डिज़ाइन करता अंकित

इस अध्याय में, आप निम्न-स्तरीय डिज़ाइन, उच्च-स्तरीय डिज़ाइन, वस्तु-उन्मुख डिज़ाइन (Object-Oriented Design), डेटाबेस डिज़ाइन, एपीआई (API - एप्लीकेशन प्रोग्रामिंग इंटरफ़ेस) डिज़ाइन, तथा उच्च-स्तरीय डिज़ाइन के उद्देश्य के बारे में जानेंगे।

निम्न-स्तरीय और उच्च-स्तरीय डिज़ाइन, सॉफ़्टवेयर विकास प्रक्रिया के दो अत्यंत महत्वपूर्ण चरण होते हैं और इन दोनों की भूमिकाएँ भिन्न होती हैं। आइए हम इन दोनों को विस्तार से समझें—

6.1 निम्न-स्तरीय डिज़ाइन (Low-level Design - LLD)—

निम्न-स्तरीय डिज़ाइन उस प्रक्रिया को कहते हैं जिसमें उच्च-स्तरीय डिज़ाइन को अधिक विशिष्ट और ठोस रूपरेखा में रूपांतरित किया जाता है ताकि सॉफ़्टवेयर का क्रियान्वयन (implementation) संभव हो सके। इसमें यह स्पष्ट किया जाता है कि सॉफ़्टवेयर का प्रत्येक घटक/मॉड्यूल किस प्रकार कार्य करेगा और अन्य घटकों के साथ किस प्रकार अंतःक्रिया करेगा। एलएलडी प्रत्येक मॉड्यूल के आंतरिक विवरणों पर केंद्रित होता है, जिसमें डेटा संरचनाएँ, एल्गोरिदम, क्लास संरचनाएँ और कार्य (functions) शामिल होते हैं। यह सॉफ़्टवेयर के तकनीकी पहलुओं की बारीकियों से संबंधित होता है।

एलएलडी अत्यधिक विस्तृत होता है और कोड के अधिक समीप होता है। इसमें उच्च-स्तरीय घटकों को छोटे, प्रबंधनीय भागों में विभाजित किया जाता है। एलएलडी में प्रायः डेवलपर्स यह निर्णय लेते हैं कि किस प्रकार की डेटा संरचनाओं, डेटाबेस स्कीमा, कोडिंग

मानकों और डिज़ाइन पैटर्न का उपयोग क्रियान्वयन चरण में किया जाएगा। निम्न-स्तरीय डिज़ाइन में यूएमएल डायग्राम जैसे क्लास डायग्राम, अनुक्रम डायग्राम (sequence diagrams), और गतिविधि डायग्राम (activity diagrams) का प्रयोग सामान्यतः किया जाता है। इसके अतिरिक्त, छद्म-कोड (pseudocode) या कोड अंश (code snippets) भी कार्यान्वयन की प्रक्रिया को स्पष्ट करने हेतु प्रयोग किए जाते हैं।

एलएलडी का मुख्य प्रतिफल (output) एक **व्यापक डिज़ाइन दस्तावेज़** होता है जो यह वर्णन करता है कि प्रत्येक घटक को किस प्रकार लागू किया जाएगा। यह दस्तावेज़ कोडिंग चरण के दौरान डेवलपर्स के लिए एक मार्गदर्शिका के रूप में कार्य करता है। एलएलडी यह सुनिश्चित करने में सहायक होता है कि सॉफ़्टवेयर को सही और दक्षतापूर्वक लागू किया जाए। यह अस्पष्टताओं को कम करता है और कोडिंग के दौरान त्रुटियों की संभावना को घटाता है।

क्या आप जानते हैं?
एलएलडी (LLD) एक चरण-दर-चरण परिष्करण प्रक्रिया है जिसका उपयोग डेटा संरचनाओं, आवश्यक सॉफ़्टवेयर संरचना (आर्किटेक्चर), स्रोत कोड और अंततः निष्पादन कलन विधियों (performance algorithms) के डिज़ाइन के लिए किया जा सकता है।

6.2 उच्च-स्तरीय डिज़ाइन (High-Level Design – HLD)

उच्च-स्तरीय डिज़ाइन डिज़ाइन की प्रारंभिक अवस्था होती है, जो समग्र प्रणाली की संरचना और वास्तुकला को परिभाषित करने पर केंद्रित होती है। इसमें प्रमुख घटकों, उनके आपसी अंतःक्रियाओं तथा उनके बीच डेटा और नियंत्रण के प्रवाह को रेखांकित किया जाता है। एलचएलडी पूरे सिस्टम पर केंद्रित रहता है और कार्यान्वयन (implementation) की सूक्ष्म जानकारियों से ऊपर उठकर एक व्यापक दृष्टिकोण प्रस्तुत करता है। यह ऐसे प्रश्नों के उत्तर देता है—"सिस्टम को क्या करना चाहिए?" और "इसे किस प्रकार संगठित किया जाना चाहिए?"

एचएलडी, एलएलडी की तुलना में कम विस्तार वाला होता है। यह समग्र दृष्टिकोण और प्रणाली-स्तरीय अवधारणाओं से संबंधित होता है। एचएलडी के दौरान, वास्तुविद् (architects) और डिज़ाइनर सिस्टम की वास्तुकला, प्रमुख मॉड्यूल, इंटरफेस और उपयोग की जाने वाली प्रौद्योगिकियों का निर्धारण करते हैं। इसमें प्रायः गैर-कारात्मक आवश्यकताओं (non-functional requirements) जैसे—विस्तारशीलता (scalability), प्रदर्शन (performance) और सुरक्षा (security) को भी परिभाषित किया जाता है।

एचएलडी में UML (यूनिफाइड मॉडलिंग लैंग्वेज) आरेख जैसे—यूज़ केस आरेख, सिस्टम आर्किटेक्चर आरेख और फ्लोचार्ट का उपयोग सामान्यतः किया जाता है। इसके साथ-साथ, पाठ्य विवरण और दस्तावेज़ीकरण (documentation) की भी महत्त्वपूर्ण भूमिका होती है। एचएलडी का मुख्य निष्कर्ष एक उच्च-स्तरीय डिज़ाइन दस्तावेज़ (high-level design document) होता है, जो विकास टीम के लिए एक रोडमैप के रूप में कार्य करता है। यह दस्तावेज़ सॉफ़्टवेयर विकास जीवन चक्र (Software Development Life Cycle – SDLC) के अगले चरणों का मार्गदर्शन करता है। एचएलडी यह सुनिश्चित करता है कि प्रणाली की समग्र संरचना परियोजना के उद्देश्यों और आवश्यकताओं के अनुरूप हो। यह परियोजना की आरंभिक अवस्था में ही प्रमुख आर्किटेक्चरल निर्णय लेने में सहायता करता है, जिससे दीर्घकाल में समय और संसाधनों की बचत होती है।

क्या आप जानते हैं?
निम्न-स्तरीय डिज़ाइन (एलएलडी) यह निर्धारित करता है कि व्यक्तिगत घटकों/मॉड्यूलों को तकनीकी रूप से कैसे कार्यान्वित किया जाएगा, जबकि उच्च-स्तरीय डिज़ाइन (एचएलडी) सिस्टम की समग्र संरचना और वास्तुकला को परिभाषित करता है।

6.3 वस्तु-उन्मुख डिज़ाइन (Object-Oriented Design – OOD)

वस्तु-उन्मुख प्रोग्राम (Object-Oriented Programming – OOP) कंप्यूटर विज्ञान और सॉफ़्टवेयर विकास में एक लोकप्रिय प्रतिमान (paradigm) है, जो "ऑब्जेक्ट्स" (वस्तुओं) की अवधारणा पर आधारित है। ऑब्जेक्ट्स कक्षाओं (classes) की प्रतिलिपियाँ होते

हैं, जो प्रयोक्ता द्वारा परिभाषित डेटा प्रकार होते हैं, जिनमें डेटा (गुण) और उस डेटा पर कार्य करने वाले मेथड (क्रियाएँ) दोनों समाहित होते हैं। यह सॉफ़्टवेयर प्रणालियों के डिज़ाइन और संगठन हेतु सबसे लोकप्रिय और व्यापक रूप से उपयोग किया जाने वाला दृष्टिकोण है।

वस्तुनिष्ठ-उन्मुख डिज़ाइन, वस्तुनिष्ठ प्रोग्रामिंग (OOP) का एक प्रमुख पक्ष है, जो कोड को पुनः उपयोगी (reusable) और आत्मनिर्भर वस्तुओं में संगठित करने को प्रोत्साहित करता है। वस्तुनिष्ठ-उन्मुख डिज़ाइन के कुछ मौलिक सिद्धांत और अवधारणाएँ निम्नलिखित हैं—

ऑब्जेक्ट्स (Objects)— ऑब्जेक्ट्स कक्षा (class) की प्रतिलिपियाँ होते हैं। वे सॉफ़्टवेयर प्रणाली के अंदर वास्तविक जीवन की संस्थाओं, अवधारणाओं या वस्तुओं का प्रतिनिधित्व करते हैं। उदाहरणस्वरूप, एक बैंकिंग अनुप्रयोग में "खाता (Account)", "ग्राहक (Customer)" और "लेन-देन (Transaction)" जैसे ऑब्जेक्ट्स हो सकते हैं।

कक्षा (Class)— कक्षा एक रूपरेखा (blueprint) या साँचा होता है, जिससे ऑब्जेक्ट्स बनाए जाते हैं। यह ऑब्जेक्ट्स की संरचना और व्यवहार को परिभाषित करती है। OOD में, कक्षाएँ डेटा (गुण या विशेषताएँ) और व्यवहार (मेथड या क्रियाएँ) दोनों को समाहित करती हैं। उदाहरण के लिए, “Car” नामक कक्षा में “color” और “make” जैसे गुण तथा “start” और “stop” जैसी क्रियाएँ हो सकती हैं।

संकेन्द्रण (Encapsulation)— संकेन्द्रण वह अवधारणा है जिसमें डेटा (गुणों) और उन पर कार्य करने वाली क्रियाओं (मेथड्स) को एकल मॉड्यूल यानी कक्षा में समाहित किया जाता है। यही मॉड्यूल ऑब्जेक्ट कहलाती है। संकेन्द्रण ऑब्जेक्ट की कार्यप्रणाली को बाहरी दृष्टि से छुपाता है और केवल आवश्यक इंटरफेस ही प्रदर्शित करता है, जिससे अन्य ऑब्जेक्ट्स के साथ इसका संप्रेषण नियंत्रित होता है।

उत्तराधिकार (Inheritance)— उत्तराधिकार वह तंत्र है, जिसके द्वारा एक कक्षा दूसरी कक्षा के गुणों और व्यवहारों को प्राप्त कर सकती है। यह कोड की पुनः उपयोगिता को बढ़ावा देता है और वर्गों का पदानुक्रम (hierarchy) बनाता है। उदाहरणस्वरूप, “Vehicle” नामक मूल कक्षा से “Car” और “Motorcycle” जैसी व्युत्पन्न कक्षाएँ गुण ग्रहण कर सकती हैं।

बहुरूपता (Polymorphism)— बहुरूपता से विभिन्न कक्षाओं के ऑब्जेक्ट्स को एक सामान्य मूल कक्षा के ऑब्जेक्ट्स के रूप में व्यवहार किया जा सकता है। यह अवधारणा डाइनैमिक मेथड डिस्पैच (dynamic method dispatch) को सक्षम बनाती है, जिसमें ऑब्जेक्ट के वास्तविक प्रकार के अनुसार उचित मेथड को क्रियान्वित किया जाता है। बहुरूपता सामान्यतः मेथड ओवरराइडिंग और इंटरफेस/सारवर्ती कक्षाओं (abstract classes) के माध्यम से प्राप्त की जाती है।

संक्षेपण (Abstraction)— संक्षेपण जटिल वास्तविकता को सरल रूप में प्रस्तुत करने की प्रक्रिया है, जिसमें कक्षाओं को उनके आवश्यक गुणों और व्यवहारों के आधार पर मॉडल किया जाता है, जबकि अनुपयुक्त विवरणों की उपेक्षा की जाती है। यह बड़े सॉफ़्टवेयर सिस्टम्स की जटिलता को प्रबंधित करने में सहायक होता है।

संबंध (Association)— संबंध दो या अधिक ऑब्जेक्ट्स के बीच का संबंध दर्शाता है। यह एक सामान्य संबंध हो सकता है, या समष्टि (aggregation – "पूर्ण–भाग" संबंध) तथा समवाय (composition – समष्टि का सशक्त रूप) जैसे जटिल संबंधों को भी दर्शा सकता है।

UML (यूनिफाइड मॉडलिंग लैंग्वेज)— यूएमएल एक दृश्यात्मक भाषा है जिसका उपयोग वस्तुनिष्ठ-उन्मुख प्रणालियों को मॉडल करने और प्रदर्शित करने के लिए किया जाता है। इसमें कक्षा आरेख (class diagrams), उपयोग प्रकरण आरेख (use case diagrams), अनुक्रम आरेख (sequence diagrams) जैसे आरेख शामिल होते हैं, जो सॉफ़्टवेयर डिज़ाइन को दर्शाने और दस्तावेज़ीकृत करने में सहायता करते हैं।

क्या आप जानते हैं?

- वस्तुनिष्ठ प्रोग्रामिंग कोड को डिज़ाइन और व्यवस्थित करने की एक संरचित और मॉड्यूलर विधि प्रदान करता है। यह मॉड्युलरिटी, पुनः उपयोगिता (reusability) और अनुरक्षणीयता (maintainability) को बढ़ावा देता है, जिससे यह सॉफ़्टवेयर इंजीनियरिंग के क्षेत्र में अत्यंत उपयोगी सिद्ध होता है।
- डेटाबेस डिज़ाइन ऐसी प्रक्रियाओं का समूह है, जो किसी व्यावसाय की डेटा प्रबंधन प्रणाली को निर्मित, कार्यान्वित और बनाए रखने में सहायता करता है।

6.4 डेटाबेस डिज़ाइन

किसी प्रोग्रामिंग परियोजना के लिए डेटाबेस डिज़ाइन करना एक सुदृढ़ और दक्ष अनुप्रयोग (application) तैयार करने की दृष्टि से एक अत्यंत महत्वपूर्ण चरण होता है। डेटाबेस डिज़ाइन की प्रक्रिया में डेटाबेस की संरचना परिभाषित करना शामिल होता है, जिसमें तालिकाएँ (tables), परस्पर संबंध (relationships) और बाध्यताएँ (constraints) सम्मिलित होती हैं।

प्रोग्रामिंग के लिए डेटाबेस डिज़ाइन करते समय अपनाए जाने वाले प्रमुख चरण और विचार निम्नलिखित हैं—

1. **आवश्यकताओं को परिभाषित करें** — अपनी एप्लिकेशन की आवश्यकताओं को समझने से शुरुआत करें। किस प्रकार का डेटा संग्रहित करना आवश्यक है? विभिन्न डेटा तत्वों के बीच क्या संबंध हैं? अपेक्षित डेटा अभिगमन पैटर्न क्या हैं (जैसे— पढ़ने की आवृत्ति अधिक, लिखने की आवृत्ति अधिक, अथवा संतुलित)?
2. **एंटिटी-रिलेशनशिप आरेख (ER डायग्राम)** — डेटाबेस में संस्थाओं (entities) और उनके संबंधों को दृश्य रूप में प्रदर्शित करने हेतु एक ER आरेख बनाएँ। प्रत्येक तालिका के लिए प्राथमिक कुंजियाँ (primary keys) तथा उनके बीच संबंध स्थापित करने हेतु विदेशी कुंजियाँ (foreign keys) चिन्हित करें।
3. **सामान्यीकरण (Normalization)** — डेटा पुनरावृत्ति को न्यूनतम करने और डेटा की अखंडता (integrity) को बनाए रखने हेतु सामान्यीकरण के सिद्धांतों को लागू करें। पुनरावृत्त जानकारी को हटाने के लिए डेटा को अलग-अलग तालिकाओं में विभाजित करें।
4. **डेटाबेस प्रबंधन प्रणाली (DBMS) का चयन** — अपनी परियोजना की आवश्यकताओं के अनुरूप एक उपयुक्त DBMS का चयन करें। प्रचलित विकल्पों में MySQL, PostgreSQL, SQLite, Microsoft SQL Server, तथा NoSQL डेटाबेस जैसे MongoDB सम्मिलित हैं।
5. **तालिका डिज़ाइन** —

   क) ER डायग्राम में पहचानी गई संस्थाओं के आधार पर तालिकाओं का डिज़ाइन करें।
   ख) प्रत्येक कॉलम के लिए डेटा प्रकार (जैसे — पूर्णांक, वर्णमाला स्ट्रिंग, दिनांक) परिभाषित करें।
   ग) प्रत्येक तालिका के लिए प्राथमिक कुंजी निर्धारित करें — सामान्यतः स्वतः-वृद्धिशील पूर्णांक या विशिष्ट पहचानकर्ता।
   घ) तालिकाओं के बीच संबंध स्थापित करने हेतु विदेशी कुंजियाँ परिभाषित करें।
6. **इंडेक्सिंग (Indexing)** — उन स्तंभों की पहचान करें जिनका उपयोग अक्सर खोज और पुनः प्राप्ति संचालन में होता है तथा उन पर इंडेक्स बनाएँ। इंडेक्स क्वेरी निष्पादन की गति में महत्वपूर्ण सुधार कर सकते हैं।
7. **बाध्यताएँ (Constraints)** — डेटा अखंडता सुनिश्चित करने हेतु उपयुक्त बाध्यताएँ परिभाषित करें, जैसे — अद्वितीय बाध्यता (unique constraints), जांच बाध्यता (check constraints), और विदेशी कुंजी बाध्यता (foreign key constraints)।
8. **डेटा अभिगमन पैटर्न** — यह विचार करें कि आपकी एप्लिकेशन किस प्रकार से डेटा तक पहुँच बनाएगी। सर्वाधिक सामान्य तथा प्रदर्शन-संवेदी क्वेरियों को समर्थन देने हेतु तालिका डिज़ाइन व इंडेक्सिंग को अनुकूलित करें।
9. **सुरक्षा** — डेटाबेस की सुरक्षा सुनिश्चित करने हेतु प्रमाणीकरण (authentication), प्राधिकरण (authorization), और एन्क्रिप्शन (encryption) जैसे उपायों को लागू करें।

10. **डेटा माइग्रेशन और प्रारंभिक डेटा भरना (Seeding)** — यदि मौजूदा डेटा को नई प्रणाली में स्थानांतरित करना है, तो उसके लिए योजना बनाएँ। साथ ही यदि आवश्यक हो, तो आरंभिक डेटा को भरने की प्रक्रिया भी निर्धारित करें।
11. **परीक्षण और अनुकूलन (Testing and Optimization)** — यह सुनिश्चित करने के लिए डेटाबेस डिज़ाइन का समुचित परीक्षण करें कि यह एप्लिकेशन की आवश्यकताओं को पूरा करता है और दक्षतापूर्वक कार्य करता है। जैसे-जैसे एप्लिकेशन का आकार बढ़े, वैसा ही डेटाबेस के प्रदर्शन की निगरानी और अनुकूलन करते रहें।
12. **बैकअप और पुनर्प्राप्ति (Recovery)** — किसी भी विफलता या डेटा क्षति की स्थिति में डेटा की सुरक्षा के लिए नियमित बैकअप और पुनर्प्राप्ति प्रक्रियाएँ लागू करें।
13. **दस्तावेजीकरण** — डेटाबेस योजना का दस्तावेज तैयार करें, जिसमें तालिकाओं की परिभाषाएँ, संबंध और बाध्यताएँ सम्मिलित हों। यह दस्तावेज भविष्य के विकास और रखरखाव के लिए अत्यंत उपयोगी होगा।
14. **विस्तारशीलता (Scalability)** — जैसे-जैसे एप्लिकेशन का उपयोग बढ़े, डेटाबेस डिज़ाइन किस प्रकार से विस्तारित होगा — इस पर विचार करें। उपयुक्त हार्डवेयर और विस्तार कार्यनीतियों का चयन करें।
15. **रखरखाव** — चल रहे रखरखाव कार्यों की योजना बनाएँ, जैसे — स्कीमा अपडेट, प्रदर्शन ट्यूनिंग और डेटा आर्काइविंग।
16. **संस्करण नियंत्रण (Version Control)** — यदि लागू हो, तो समय के साथ डेटाबेस स्कीमा में होने वाले परिवर्तनों को प्रबंधित करने हेतु संस्करण नियंत्रण प्रणाली का उपयोग करें।
17. **अनुपालन और नियमन** — यह सुनिश्चित करें कि आपका डेटाबेस डिज़ाइन किसी भी संगत उद्योग मानकों या डेटा सुरक्षा अधिनियमों का पालन करता है।
18. **निगरानी और लॉगिंग** — डेटाबेस के प्रदर्शन की निगरानी और समस्याओं की पहचान हेतु निगरानी तथा लॉगिंग को लागू करें।

जैसे-जैसे आपकी एप्लिकेशन विकसित होती है और उसकी आवश्यकताओं की समझ गहरी होती है, वैसे-वैसे अपने डिज़ाइन को परिष्कृत और अनुकूलित करना आवश्यक होता है। डेटाबेस प्रशासकों और डेवलपर्स जैसे अन्य टीम सदस्यों के साथ सहयोग भी एक सफल डेटाबेस डिज़ाइन विकसित करने में अत्यंत महत्वपूर्ण है।

क्या आप जानते हैं?
डेटाबेस डिज़ाइन एक पुनरावृत्ति (iterative) की प्रक्रिया है।

6.5 एपीआई (API — एप्लिकेशन प्रोग्रामिंग इंटरफेस) डिज़ाइन

एपीआई (एप्लिकेशन प्रोग्रामिंग इंटरफ़ेस) का डिज़ाइन सॉफ़्टवेयर सिस्टम के निर्माण का एक महत्वपूर्ण पहलू होता है — चाहे वह वेब एप्लिकेशन हो, मोबाइल एप्लिकेशन हो या अन्य कोई सॉफ़्टवेयर घटक। एक सुव्यवस्थित एपीआई डेवलपर्स के लिए आपके सिस्टम से संवाद करना सरल बना सकता है, सुसंगतता को बढ़ावा देता है और भविष्य की विस्तारशीलता (scalability) सुनिश्चित करता है।

क्या आप जानते हैं?
एपीआई ऐसे सॉफ़्टवेयर इंटरफ़ेस होते हैं जो दो अनुप्रयोगों को अनुरोध (request) और प्रत्युत्तर (response) के माध्यम से एक-दूसरे से संवाद करने की अनुमति देते हैं।

एपीआई डिज़ाइन के कुछ प्रमुख सिद्धांत और सर्वोत्तम व्यवहार इस प्रकार हैं—

1. **स्पष्ट उद्देश्य और क्षेत्र (Scope)** — अपने एपीआई का उद्देश्य परिभाषित करें। यह किस समस्या का समाधान करता है या कौन-सी कार्यक्षमता प्रदान करता है? एपीआई का क्षेत्र स्पष्ट रूप से परिभाषित करें ताकि वह अत्यधिक जटिल या बड़ा न हो जाए।

2. **सुसंगतता (Consistency)** — नामकरण संप्रत्ययों (naming conventions), URI संरचनाओं और अनुरोध/प्रत्युत्तर प्रारूपों में सुसंगतता बनाए रखें। एपीआई अनुरोधों के परिणाम को सूचित करने के लिए HTTP स्थिति कोड और त्रुटि संदेशों का सुसंगत उपयोग करें।
3. **RESTful सिद्धांतों का पालन करें** — जहाँ संभव हो, REST (Representational State Transfer) सिद्धांतों का पालन करें। CRUD (Create, Read, Update, Delete) संचालन के लिए HTTP विधियों (GET, POST, PUT, DELETE) का उपयोग करें। संसाधन URIs ऐसे बनाएँ जो संस्थाओं का प्रतिनिधित्व करें और उन पर क्रियाएँ HTTP क्रियाओं से की जाएँ।
4. **संस्करण नियंत्रण (Versioning)** — अपने एपीआई के URI में संस्करण संख्या शामिल करें (जैसे— /v1/resource), ताकि भविष्य में अद्यतन किए जा सकें बिना मौजूदा ग्राहकों को प्रभावित किए। URL संस्करण नियंत्रण या हेडर संस्करण नियंत्रण जैसी उपयुक्त कार्यनीतियाँ अपनाएँ।
5. **प्रमाणीकरण और प्राधिकरण** — OAuth2 या API कुंजियों जैसे सुरक्षित प्रमाणीकरण और प्राधिकरण तंत्र को लागू करें। प्रमाणीकरण और आवश्यक अनुमति प्राप्त करने की प्रक्रिया को स्पष्ट रूप से दस्तावेज में दर्ज करें।
6. **अनुरोध और प्रत्युत्तर प्रारूप** — अनुरोध और प्रत्युत्तर पेलोड्स के लिए JSON या XML जैसे मानक डेटा प्रारूपों का उपयोग करें। प्रत्युत्तर पेलोड्स सुव्यवस्थित, पूर्ण और आवश्यक जानकारी से युक्त होने चाहिए।
7. **त्रुटि प्रबंधन (Error Handling)** — अर्थपूर्ण त्रुटि कोड और संदेशों के साथ स्पष्ट और सुसंगत त्रुटि प्रत्युत्तर परिभाषित करें। त्रुटि में उस फ़ील्ड का उल्लेख करें जिससे समस्या उत्पन्न हुई है, ताकि डेवलपर्स डिबग कर सकें।
8. **पृष्ठांकन और फ़िल्टरिंग** — बड़े डेटा सेट के लिए पृष्ठांकन (pagination) लागू करें ताकि प्रदर्शन सुधरे और सर्वर पर भार कम हो। जहाँ लागू हो, फ़िल्टरिंग और क्रमबद्धता (sorting) की सुविधा भी प्रदान करें।
9. **दर सीमा (Rate Limiting)** — एपीआई के अनुचित उपयोग से सुरक्षा और समान उपयोग सुनिश्चित करने हेतु दर सीमा लागू करें। दर सीमा और सीमा पार करने पर उत्तर की प्रक्रिया का स्पष्ट रूप से दस्तावेज करें।
10. **प्रलेखन (Documentation)** — एपीआई का व्यापक और अद्यतन प्रलेखन प्रदान करें। Swagger या OpenAPI जैसे उपकरण इंटरैक्टिव दस्तावेज तैयार करने में सहायक होते हैं। उपयोग के उदाहरण और कोड नमूने शामिल करें जिससे डेवलपर्स को एपीआई उपयोग में सुविधा हो।
11. **परीक्षण (Testing)** — एपीआई के सभी एंडपॉइंट्स का, कोनों की स्थितियों (edge cases) और त्रुटि परिदृश्यों सहित, परीक्षण करें। सतत परीक्षण के लिए स्वचालित परीक्षण उपकरण और निरंतर एकीकरण (CI) अपनाएँ।
12. **सुरक्षा (Security)** — इनपुट सत्यापन, पैरामीटराइज़्ड क्वेरीज़ और सामान्य सुरक्षा जोखिमों (जैसे— SQL Injection, XSS) से सुरक्षा जैसे सर्वोत्तम सुरक्षा उपाय अपनाएँ।
13. **प्रतिक्रिया और समुदाय सहभागिता** — एपीआई उपयोग करने वाले डेवलपर्स से प्रतिक्रिया को प्रोत्साहित करें और उनकी आवश्यकताओं के प्रति उत्तरदायी रहें। एपीआई के चारों ओर एक समुदाय विकसित करें — जैसे फ़ोरम या सहायता चैनल।

एपीआई डिज़ाइन एक पुनरावृत्त (iterative) प्रक्रिया है, और इसे निरंतर बेहतर बनाने हेतु प्रयोक्ताओं से प्रतिक्रिया प्राप्त करना आवश्यक है। साथ ही, उद्योग की सर्वोत्तम प्रथाओं और नवीनतम मानकों के साथ अपडेट रहना सफल एपीआई बनाए रखने के लिए अनिवार्य है।

6.6 निम्न-स्तरीय डिज़ाइन (Low-Level Design) बनाना

निम्न-स्तरीय डिज़ाइन (Low-Level Design या LLD) बनाना सॉफ्टवेयर विकास प्रक्रिया का एक आवश्यक चरण है। इसमें किसी सॉफ्टवेयर प्रणाली के उच्च-स्तरीय डिज़ाइन या वास्तुकला खाके को लेकर उसे छोटे-छोटे, अधिक विस्तृत घटकों में विभाजित किया जाता है। ये घटक सामान्यतः मॉड्यूल, क्लास, फंक्शन, डेटा संरचना (data structures) और उनके आपसी संबंध होते हैं। LLD का उद्देश्य यह सुनिश्चित करना होता है कि सिस्टम के प्रत्येक घटक को कैसे कार्यान्वित किया जाएगा, इसका एक विस्तृत योजना प्रस्तुत की जाए।

निम्न-स्तरीय डिज़ाइन बनाने की चरणबद्ध प्रक्रिया इस प्रकार है —

1. **आवश्यकताओं को समझें** — LLD बनाने से पहले सिस्टम की आवश्यकताओं की गहन समझ आवश्यक है। इसमें कार्यात्मक (functional) और गैर-कार्यात्मक (non-functional) दोनों प्रकार की आवश्यकताएँ शामिल होती हैं। उच्च-स्तरीय डिज़ाइन और प्रणाली की वास्तुकला को भी स्पष्ट रूप से समझना आवश्यक है।
2. **मॉड्यूल/घटक पहचानें** — सिस्टम को छोटे-छोटे मॉड्यूल या घटकों में विभाजित करें। प्रत्येक मॉड्यूल की एक स्पष्ट जिम्मेदारी होनी चाहिए और उसे किसी विशिष्ट कार्यात्मकता को समाहित करना चाहिए।
3. **इंटरफेस परिभाषित करें** — प्रत्येक मॉड्यूल या घटक के लिए वे इंटरफेस (या एपीआई) परिभाषित करें जिनके माध्यम से वह अन्य मॉड्यूलों से संवाद करेगा। इनमें इनपुट पैरामीटर, आउटपुट तथा अपेक्षित व्यवहार शामिल होने चाहिए।
4. **डेटा संरचना डिज़ाइन करें** — उन डेटा संरचनाओं को परिभाषित करें जिन्हें मॉड्यूल उपयोग करेंगे। इसमें डेटा ऑब्जेक्ट्स, डेटाबेस तथा अन्य डेटा भंडारण प्रणालियों की संरचना शामिल है।
5. **कलन विधियाँ और तर्क (Algorithm and Logic)** — प्रत्येक मॉड्यूल के अंदर प्रयुक्त होने वाली कलन विधियाँ (algorithms) और तर्क की रूपरेखा बनाएं। इसमें फ्लोचार्ट, छद्म कोड (pseudocode) या विस्तृत वर्णन हो सकते हैं कि मॉड्यूल कैसे कार्य करेगा।
6. **त्रुटि प्रबंधन** — यह निर्दिष्ट करें कि प्रत्येक मॉड्यूल के अंदर त्रुटियों और अपवादों को कैसे संभाला जाएगा। त्रुटि कोड, संदेश और आवश्यक क्रियाएँ भी परिभाषित करें।
7. **समवर्तिता (concurrent) और मल्टीथ्रेडिंग** — यदि आपके सिस्टम को समवर्ती (concurrent) या मल्टीथ्रेडेड होने की आवश्यकता है, तो यह निर्दिष्ट करें कि प्रत्येक मॉड्यूल में इसे कैसे संभाला जाएगा। समकालिकता (synchronization) तंत्र और थ्रेड-सुरक्षित डेटा संरचनाओं को परिभाषित करें।
8. **प्रदर्शन विचार** — प्रदर्शन को बेहतर बनाने की कार्यनीतियाँ, जैसे कैशिंग, अनुक्रमण (indexing) या समांतर प्रोसेसिंग (parallel processing) पर विचार करें और जहाँ आवश्यक हो, उन्हें डिज़ाइन में शामिल करें।
9. **सुरक्षा** — डिज़ाइन में संभावित सुरक्षा जोखिमों और कमजोरियों की पहचान करें और उन्हें संबोधित करने की योजना बनाएं। इसमें एन्क्रिप्शन, प्रमाणीकरण (authentication) और पहुँच नियंत्रण (access control) उपाय शामिल हो सकते हैं।
10. **परीक्षण और प्रमाणीकरण** — यह परिभाषित करें कि प्रत्येक मॉड्यूल का परीक्षण और प्रमाणीकरण कैसे किया जाएगा। इसमें यूनिट परीक्षण, एकीकरण परीक्षण तथा अन्य प्रासंगिक परीक्षण विधियाँ शामिल हैं।
11. **प्रलेखन** — LLD का पूर्ण विवरण दस्तावेजीकृत करें। इसमें डिज़ाइन आरेख, कोड टिप्पणियाँ (code comments) और अन्य ऐसी जानकारी शामिल होनी चाहिए जो डेवलपर्स को डिज़ाइन समझने और लागू करने में सहायक हो।
12. **समीक्षा और प्रतिक्रिया** — अपनी विकास टीम के साथ डिज़ाइन की समीक्षा करें, सुझाव प्राप्त करें और डिज़ाइन में आवश्यक सुधार करें। यह सुनिश्चित करें कि डिज़ाइन उच्च-स्तरीय डिज़ाइन और आवश्यकताओं के अनुरूप है।
13. **परिष्करण (Refinements)** — आवश्यकतानुसार प्राप्त प्रतिक्रिया और आवश्यकता परिवर्तनों के अनुसार LLD को अद्यतन करें। परियोजना की प्रगति के साथ डिज़ाइन को अद्यतन बनाए रखें।
14. **कार्यान्वयन (Implementation)** — एक बार LLD को अंतिम रूप दे दिया जाए, तब डेवलपर्स इस डिज़ाइन के अनुसार सिस्टम का निर्माण आरंभ कर सकते हैं।

LLD (लो लेवल डिज़ाइन) में आवश्यक विवरण की मात्रा परियोजना की जटिलता और आपकी विकास टीम की प्राथमिकताओं के अनुसार भिन्न हो सकती है। यह आवश्यक है कि कार्यान्वयन हेतु पर्याप्त विवरण प्रदान करते हुए अनावश्यक जटिलता से बचने के बीच संतुलन स्थापित किया जाए।

6.7 उच्च-स्तरीय डिज़ाइन (High-Level Design – HLD) का परिचय

उच्च-स्तरीय डिज़ाइन (High-Level Design या HLD) सॉफ्टवेयर विकास एवं प्रणाली वास्तुकला प्रक्रिया का एक अत्यंत महत्वपूर्ण चरण है। यह प्रारंभिक आवश्यकताओं के संकलन और विश्लेषण के बाद तथा विस्तृत डिज़ाइन और कार्यान्वयन से पहले आने वाला एक बुनियादी कदम है। HLD किसी प्रणाली या सॉफ़्टवेयर अनुप्रयोग का एक वैचारिक रूपरेखा या वास्तु-दृष्टिकोण तैयार करने पर केंद्रित होता है।

6.7.1 उच्च-स्तरीय डिज़ाइन (HLD) का उद्देश्य

HLD आवश्यकताओं के संकलन और निम्न-स्तरीय डिज़ाइन (LLD) या कार्यान्वयन के बीच एक सेतु के रूप में कार्य करता है। यह प्रणाली का विहंगम दृश्य प्रदान करता है, जिससे हितधारकों को उसकी संरचना, घटकों और अंतःक्रियाओं को समझने में सहायता मिलती है। यह परियोजना के लिए वास्तुकला की नींव रखता है और महत्वपूर्ण डिज़ाइन निर्णयों को स्थापित करता है।

6.7.2 उच्च-स्तरीय डिज़ाइन के प्रमुख घटक

1. **प्रणाली वास्तुकला** — प्रणाली की समग्र संरचना निर्धारित करें, जिसमें प्रमुख घटक, उनके आपसी संबंध और उनके कार्य करने के तरीके शामिल हों।
2. **डेटा डिज़ाइन** — डेटा मॉडल को परिभाषित करें, जिसमें डेटाबेस, डेटा भंडारण और प्रणाली के अंदर डेटा प्रवाह शामिल हो।
3. **इंटरफेस डिज़ाइन** — उन बाहरी इंटरफेस और संचार प्रोटोकॉल (जैसे API, वेब सेवाएँ आदि) की पहचान करें जिन्हें प्रणाली उपयोग करेगी।
4. **तकनीकी स्टैक** — विकास के लिए प्रयुक्त की जाने वाली प्रोग्रामिंग भाषाओं, फ्रेमवर्क और उपकरणों का चयन करें।
5. **सुरक्षा विचार** — उच्च-स्तरीय सुरक्षा उपायों और पहुँच नियंत्रणों का विवरण दें।
6. **विस्तारणीयता और प्रदर्शन** — विचार करें कि प्रणाली भविष्य की वृद्धि को कैसे संभालेगी और सर्वोत्तम प्रदर्शन कैसे सुनिश्चित करेगी।
7. **त्रुटि प्रबंधन और अपवाद नियंत्रण** — परिभाषित करें कि प्रणाली त्रुटियों और अपवादों को कैसे दक्षता से संभालेगी।

6.7.3 उच्च-स्तरीय डिज़ाइन में प्रमुख गतिविधियाँ

1. **आवश्यकता विश्लेषण** — परियोजना आवश्यकताओं की समीक्षा करें और यह सुनिश्चित करें कि वे प्रस्तावित वास्तुकला के अनुरूप हैं।
2. **प्रणाली विघटन (System Decomposition)** — प्रणाली को प्रबंधन योग्य मॉड्यूलों या घटकों में विभाजित करें और उनकी जिम्मेदारियाँ परिभाषित करें।
3. **डेटा मॉडलिंग** — डेटा संरचना और संबंधों को दर्शाने के लिए एंटिटी-रिलेशनशिप आरेख या अन्य मॉडल तैयार करें।
4. **इंटरफेस डिज़ाइन** — उन API और बाह्य संचार विधियों को निर्दिष्ट करें जिनका उपयोग प्रणाली अन्य प्रणालियों या प्रयोक्ताओं के साथ संवाद करने हेतु करेगी।
5. **प्रौद्योगिकी चयन** — परियोजना की आवश्यकताओं के अनुरूप सर्वोत्तम प्रौद्योगिकियों, लाइब्रेरी और उपकरणों का चयन करें।
6. **प्रोटोटाइप निर्माण** — कुछ मामलों में विस्तृत कार्यान्वयन से पहले डिज़ाइन अवधारणा की पुष्टि हेतु प्रोटोटाइप या मॉक-अप तैयार करें।

6.7.4 उच्च-स्तरीय डिज़ाइन के निष्पाद्य (Deliverables)

1. **वास्तुकला आरेख** — प्रणाली की वास्तुकला को दर्शाने वाले दृश्य रूप, जैसे ब्लॉक डायग्राम, फ्लोचार्ट या UML आरेख।
2. **डेटा मॉडल** — डेटा संरचनाओं और प्रवाहों को चित्रित करने वाले एंटिटी-रिलेशनशिप आरेख (ERD) या डेटा प्रवाह आरेख (DFD)।
3. **इंटरफेस विनिर्देश** — बाहरी इंटरफेस, API तथा संचार प्रोटोकॉल का विस्तृत दस्तावेजीकरण।

4. **तकनीकी स्टैक दस्तावेजीकरण** — परियोजना के लिए चुनी गई प्रौद्योगिकियों, फ्रेमवर्क और उपकरणों की सूची, उनके औचित्य के साथ।
5. **उच्च-स्तरीय सुरक्षा योजना** — प्रणाली के लिए नियोजित सुरक्षा उपायों और पहुँच नियंत्रणों का अवलोकन।
6. **प्रदर्शन और विस्तार क्षमता विचार** — यह विवरण कि प्रणाली वृद्धि को कैसे संभालेगी और सर्वोत्तम प्रदर्शन कैसे सुनिश्चित करेगी।

उच्च-स्तरीय डिज़ाइन सॉफ़्टवेयर विकास का एक महत्वपूर्ण चरण है, जो परियोजना की वास्तुकला नींव स्थापित करता है। यह प्रणाली की संरचना, डेटा प्रवाह, इंटरफेस और तकनीकी विकल्पों को परिभाषित करता है तथा विस्तृत डिज़ाइन और कार्यान्वयन के आगामी चरणों के लिए एक स्पष्ट मार्ग प्रदान करता है। प्रभावी HLD यह सुनिश्चित करता है कि अंतिम प्रणाली आवश्यकताओं को पूरा करे, दक्षतापूर्वक कार्य करे और भविष्य में विस्तार व अनुरक्षण हेतु सक्षम बनी रहे।

अभ्यास 6.1

1. उच्च-स्तरीय डिज़ाइन (High-Level Design) की प्रमुख गतिविधियों को सूचीबद्ध कीजिए।
2. निम्न-स्तरीय डिज़ाइन (Low-Level Design) बनाने के किन्हीं पाँच चरणों की सूची बनाइए।
3. API डिज़ाइन के लिए किसी भी पाँच प्रमुख सिद्धांतों को सूचीबद्ध कीजिए।

सारांश

- निम्न-स्तरीय डिज़ाइन (LLD) सॉफ़्टवेयर क्रियान्वयन हेतु विस्तृत योजना से संबंधित होती है।
- उच्च-स्तरीय डिज़ाइन (HLD) प्रणाली की वास्तुकला और संरचना पर केंद्रित होती है।
- ऑब्जेक्ट-ओरिएंटेड डिज़ाइन (OOD) में कोड को व्यवस्थित करने के लिए ऑब्जेक्ट्स, क्लासेस और सिद्धांतों का प्रयोग किया जाता है।
- डेटाबेस डिज़ाइन में आवश्यकताओं का विश्लेषण, ईआर आरेख और डेटा सामान्यीकरण जैसे चरण शामिल होते हैं।
- API डिज़ाइन का उद्देश्य स्पष्ट होना चाहिए, उसमें सुसंगतता, RESTful प्रथाएँ और प्रलेखन होना आवश्यक है।
- उच्च-स्तरीय डिज़ाइन एक वास्तुकला अवलोकन के रूप में कार्य करती है जिसमें घटकों और तकनीकी चयन को परिभाषित किया जाता है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न (MCQs)—

1. निम्न-स्तरीय डिज़ाइन (LLD) का प्राथमिक फोकस क्या है?
   (क) उच्च-स्तरीय वास्तुकला
   (ख) सॉफ़्टवेयर घटकों के आंतरिक विवरण
   (ग) प्रणाली आवश्यकताएँ
   (घ) परियोजना प्रलेखन

2. निम्नलिखित में से कौन-सा UML आरेख सामान्यतः उच्च-स्तरीय डिज़ाइन (HLD) में प्रयुक्त होता है?
   (क) अनुक्रम आरेख
   (ख) क्लास आरेख

(ग) गतिविधि आरेख

(घ) ईआर आरेख

3. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन (OOD) में ऑब्जेक्ट्स क्या होते हैं?

(क) क्लासेस के उदाहरण

(ख) डेटा विशेषताएँ

(ग) विधियाँ

(घ) चर

4. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन में एन्कैप्सुलेशन क्या है?

(क) अन्य ऑब्जेक्ट्स से डेटा को छिपाना

(ख) कई क्लासेस को एक में सम्मिलित करना

(ग) नई क्लासेस उत्पन्न करने के लिए इनहेरिटेंस का उपयोग

(घ) विधियों में पॉलीमॉर्फिज़्म लागू करना

5. डेटाबेस डिज़ाइन का कौन-सा चरण डेटा पुनरावृत्ति को न्यूनतम करता है?

(क) एंटिटी-रिलेशनशिप आरेख

(ख) सामान्यीकरण

(ग) तालिका डिज़ाइन

(घ) सुरक्षा

6. API डिज़ाइन में रेट लिमिटिंग का उद्देश्य क्या है?

(क) सुरक्षित प्रमाणीकरण सुनिश्चित करना

(ख) दुरुपयोग रोकना और निष्पक्ष उपयोग सुनिश्चित करना

(ग) डेटाबेस प्रदर्शन का अनुकूलन

(घ) अनुरोध और प्रतिक्रिया प्रारूप परिभाषित करना

7. निम्न-स्तरीय डिज़ाइन बनाने का कौन-सा चरण त्रुटि कोड और संदेशों को परिभाषित करता है?

(क) मॉड्यूल/घटकों की पहचान करना

(ख) इंटरफेस परिभाषित करना

(ग) त्रुटि प्रबंधन

(घ) प्रलेखन

8. उच्च-स्तरीय डिज़ाइन (HLD) सॉफ़्टवेयर विकास के किन दो चरणों के बीच सेतु का कार्य करता है?

(क) आवश्यकता विश्लेषण और प्रोटोटाइप

(ख) विस्तृत डिज़ाइन और क्रियान्वयन

(ग) निम्न-स्तरीय डिज़ाइन और परीक्षण

(घ) प्रणाली वास्तुकला और डेटा डिज़ाइन

9. डेटाबेस डिज़ाइन के लिए संस्करण नियंत्रण (Version Control) को लागू करने का प्रमुख लाभ क्या है?

(क) सुरक्षा सुनिश्चित करना

(ख) रेट लिमिट हैंडल करना

(ग) समय के साथ स्कीमा में हुए परिवर्तनों का प्रबंधन

(घ) परीक्षण और प्रमाणीकरण

10. उच्च-स्तरीय डिज़ाइन (HLD) मुख्यतः किस पक्ष पर केंद्रित होती है?

(क) तकनीकी क्रियान्वयन विवरण

(ख) वास्तुकला अवलोकन

(ग) निम्न-स्तरीय घटक
(घ) एल्गोरिदम विकास

ख. रिक्त स्थान भरिए

1. उच्च-स्तरीय डिज़ाइन (HLD) प्रणाली की संरचना और घटकों का _____ दृश्य प्रदान करती है।
2. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन (OOD) में ऑब्जेक्ट्स, डेटा (गुण) और _____ दोनों को समाहित करते हैं।
3. डेटाबेस सामान्यीकरण में डेटा _____ को न्यूनतम करना और डेटा अखंडता में सुधार करना शामिल है।
4. API डिज़ाइन में रेट लिमिटिंग लागू करने से _____ को रोका जा सकता है और निष्पक्ष उपयोग सुनिश्चित होता है।
5. _____ डिज़ाइन बनाने में त्रुटि कोड, त्रुटि संदेश और कार्रवाई को परिभाषित किया जाता है।
6. उच्च-स्तरीय डिज़ाइन _____ संकलन और क्रियान्वयन के बीच सेतु का कार्य करती है।
7. API डिज़ाइन में रेट लिमिटिंग दुरुपयोग को रोकने और _____ सुनिश्चित करने के लिए लागू की जाती है।
8. निम्न-स्तरीय डिज़ाइन (LLD) का प्रमुख निष्कर्ष एक व्यापक _____ दस्तावेज़ होता है।
9. उच्च-स्तरीय डिज़ाइन प्रणाली की वास्तुकला, _____, घटक और तकनीकी विकल्पों को परिभाषित करती है।
10. API डिज़ाइन में संस्करण नियंत्रण (Versioning) भविष्य में बिना _____ मौजूदा क्लाइंट्स को अपडेट करने की सुविधा देता है।

ग. सही या गलत लिखिए

1. निम्न-स्तरीय डिज़ाइन (LLD) सॉफ़्टवेयर के अत्यंत तकनीकी पहलुओं पर केंद्रित होती है।
2. उच्च-स्तरीय डिज़ाइन (HLD), LLD की तुलना में कम विस्तृत होती है।
3. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन (OOD) में ऑब्जेक्ट्स केवल डेटा को समाहित करते हैं, विधियाँ नहीं।
4. डेटाबेस डिज़ाइन में सामान्यीकरण का अर्थ डेटा को अलग-अलग तालिकाओं में विभाजित करना होता है।
5. API डिज़ाइन में रेट लिमिटिंग का कोई महत्व नहीं है।
6. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन में इनहेरिटेंस से एक क्लास दूसरी क्लास की विशेषताएँ और व्यवहार प्राप्त कर सकती है।
7. उच्च-स्तरीय डिज़ाइन में क्लास आरेख और अनुक्रम आरेख जैसे विस्तृत UML आरेख शामिल होते हैं।
8. डेटाबेस डिज़ाइन एक पुनरावृत्त (iterative) प्रक्रिया है।
9. डेटाबेस डिज़ाइन में ERD का उपयोग एंटिटी और उनके संबंधों को दर्शाने के लिए किया जाता है।
10. उच्च-स्तरीय डिज़ाइन प्रणाली के तकनीकी क्रियान्वयन विवरणों पर केंद्रित होती है।

घ. लघु उत्तर प्रश्न

1. सॉफ़्टवेयर विकास में निम्न-स्तरीय डिज़ाइन (LLD) का प्रमुख उद्देश्य क्या है?
2. फोकस और विवरण के संदर्भ में उच्च-स्तरीय डिज़ाइन (HLD) निम्न-स्तरीय डिज़ाइन (LLD) से कैसे भिन्न है?
3. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन (OOD) में एन्कैप्सुलेशन की अवधारणा को समझाइए।
4. डेटाबेस डिज़ाइन प्रक्रिया में सामान्यीकरण (Normalization) का क्या महत्व है?
5. API डिज़ाइन में रेट लिमिटिंग क्यों महत्वपूर्ण है, और यह अनुप्रयोगों को कैसे लाभ पहुँचाती है?
6. ऑब्जेक्ट-ओरिएंटेड डिज़ाइन (OOD) में इनहेरिटेंस कोड पुन— उपयोग को कैसे बढ़ावा देता है?
7. पॉलीमॉर्फिज़्म ऑब्जेक्ट-ओरिएंटेड डिज़ाइन में गतिशील विधि प्रेषण (dynamic method dispatch) को कैसे सक्षम करता है?
8. सॉफ़्टवेयर डिज़ाइन में अमूर्तता (Abstraction) जटिल वास्तविकता को सरल बनाने में क्या भूमिका निभाती है?
9. सॉफ़्टवेयर विकास प्रक्रिया में निम्न-स्तरीय डिज़ाइन (LLD) के प्रमुख निष्कर्ष क्या हैं?
10. डेटाबेस डिज़ाइन के प्रभावी प्रबंधन में समय के साथ संस्करण नियंत्रण (Version Control) कैसे सहायक होता है?

सत्र 7

डिज़ाइन दोषों की जाँच और पहचान

मिलिए अंकित से, एक ऐसा बच्चा जिसे LEGO से घर बनाना और चित्र बनाना बहुत पसंद था। लेकिन कभी-कभी उनमें कुछ गलतियाँ हो जाया करती थीं। तब उसने "डिज़ाइन दोषों की जाँच और पहचान" के बारे में सीखा। उसे यह समझ में आया कि यह बिल्कुल एक जासूस बनने जैसा है—

जाँच करना— यह ऐसा है जैसे यह सुनिश्चित करना कि LEGO का घर ठीक से खड़ा है और चित्र सुंदर बना है। **जांचना**— यह ऐसा है जैसे किसी चीज़ को ध्यान से देखकर उसमें कोई गलती या धब्बा खोज निकालना। **गलतियाँ ढूँढना**— यह ऐसा है जैसे LEGO सेट में कोई टुकड़ा गायब हो और यह समझना कि उसे कैसे ठीक करें।

अंकित ने ये तकनीकें अपने निर्माण कार्य में अपनाईं। इससे उसकी बनाई चीज़ें और बेहतर हो गईं, जैसे कोई सुपर-जासूस मजेदार पहेली सुलझा रहा हो। उसने यह समझा कि किसी निर्माण कार्य में गलतियाँ ढूँढना और उन्हें सुधारना ही इस रोमांचक यात्रा का एक हिस्सा है, जैसा कि **चित्र 7.1** में दिखाया गया है।

चित्र 7.1— अंकित डिज़ाइन दोषों की पहचान करता हुआ

क्या आप जानते हैं?
डिज़ाइन दोष (Design Defect) उत्पाद की डिज़ाइन में ऐसी खामी होती है जो उसे प्रयोक्ता के लिए अनुचित रूप से खतरनाक बना देती है।

7.1 डिज़ाइन दोष

किसी उत्पाद या प्रणाली में डिज़ाइन दोष कई प्रकार की समस्याएँ उत्पन्न कर सकते हैं, जिनमें सुरक्षा जोखिम, कार्यदक्षता की कमी और ग्राहक असंतोष शामिल हैं। इन दोषों के कई कारण हो सकते हैं और इन दोषों तथा उनके अंतर्निहित कारणों को समझना उन्हें रोकने के लिए आवश्यक होता है।

कुछ सामान्य डिज़ाइन दोष और उनके संभावित कारण निम्नलिखित हैं—

1. **अपर्याप्त सुरक्षा उपाय**— उत्पादों या प्रणालियों में सुरक्षा सुविधाओं या उपायों की कमी दुर्घटनाओं और चोटों का कारण बन सकती है।
2. **खराब एर्गोनॉमिक्स (Ergonomics)**— ऐसे उत्पाद जो उपयोग में असुविधाजनक या कठिन होते हैं, वे प्रयोक्ता को असहज बना सकते हैं और उत्पादकता को घटा सकते हैं।

3. **अप्रभावी कार्यक्षमता**— ऐसे उत्पाद जो अपने नियत कार्यों को प्रभावी तरीके से नहीं करते, वे ग्राहक असंतोष और संसाधनों की बर्बादी का कारण बनते हैं।
4. **क्षमता से संबंधित समस्याएँ**— अन्य प्रणालियों या घटकों के साथ असंगतता एकीकरण समस्याओं और कार्यक्षमता में कमी का कारण बन सकती है।
5. **अपर्याप्त टिकाऊपन**— ऐसे उत्पाद जो जल्दी घिस जाते हैं या सामान्य उपयोग की स्थितियों को सहन करने के लिए नहीं बने होते, वे बार-बार प्रतिस्थापन और अधिक लागत उत्पन्न करते हैं।
6. **जटिलता**— अत्यधिक जटिल डिज़ाइन प्रयोक्ताओं को भ्रमित कर सकते हैं और त्रुटियों तथा रखरखाव की चुनौतियों की संभावना बढ़ा सकते हैं।
7. **प्रयोक्ता मित्रवत इंटरफेस की कमी**— खराब डिज़ाइन किए गए प्रयोक्ता इंटरफेस प्रयोक्ताओं को निराश कर सकते हैं और सॉफ़्टवेयर अनुप्रयोगों या उपकरणों की उपयोगिता को बाधित कर सकते हैं।
8. **लागत अधिक होना**— ऐसे डिज़ाइन जो बजट बॉर्डर्स को पार कर जाते हैं, वे वित्तीय समस्याओं और परियोजना में देरी का कारण बन सकते हैं।
9. **पर्यावरण पर प्रभाव**— ऐसे डिज़ाइन जो पर्यावरण पर नकारात्मक प्रभाव डालते हैं, जैसे अत्यधिक ऊर्जा खपत या हानिकारक सामग्री का उपयोग, स्थिरता की चिंताओं को जन्म देते हैं।

7.2 दोषों के प्रकार

सॉफ़्टवेयर विकास में पाए जाने वाले कुछ बुनियादी दोष निम्नलिखित हैं—

1. **अंकगणितीय दोष (Arithmetic Defects)**— यह वे दोष होते हैं जो डेवलपर द्वारा किसी अंकगणितीय अभिव्यक्ति में की गई गलती या उसके समाधान में की गई त्रुटि के कारण उत्पन्न होते हैं। यह दोष प्रोग्रामर द्वारा अधिक कार्यभार या अपर्याप्त ज्ञान के कारण होता है। कोड की भीड़-भाड़ (Code Congestion) के कारण भी अंकगणितीय दोष हो सकते हैं क्योंकि प्रोग्रामर द्वारा लिखे गए कोड को ठीक से देख पाना कठिन हो जाता है।
2. **तार्किक दोष (Logical Defects)**— यह कोड कार्यान्वयन में की गई गलतियाँ होती हैं। जब प्रोग्रामर समस्या को सही तरीके से नहीं समझता या गलत दिशा में सोचता है, तब ऐसे दोष उत्पन्न होते हैं। साथ ही, यदि कोड लिखते समय वह किन्हीं विशेष स्थितियों (Corner Cases) का ध्यान नहीं रखता तो यह दोष होते हैं। यह मुख्य रूप से सॉफ़्टवेयर के मूल से संबंधित होता है।
3. **वाक्य विन्यास दोष (Syntax Defects)**— यह कोड की लेखन शैली में की गई गलतियों को दर्शाता है। डेवलपर द्वारा कोड लिखते समय छोटे प्रतीकों के छूटने से यह दोष उत्पन्न होते हैं। उदाहरणस्वरूप, यदि कोई C++ कोड लिखते समय सेमीकोलन (;) लगाना भूल जाए तो वह वाक्य विन्यास दोष होगा।
4. **मल्टीथ्रेडिंग दोष (Multithreading Defects)**— मल्टीथ्रेडिंग का अर्थ है एक ही समय पर कई कार्यों का निष्पादन। इस प्रक्रिया में डिबगिंग जटिल हो सकती है। कभी-कभी इसमें डेडलॉक (Deadlock) या स्टार्वेशन (Starvation) की स्थिति बन जाती है, जिससे प्रणाली विफल हो सकती है।
5. **इंटरफेस दोष (Interface Defects)**— यह सॉफ़्टवेयर और प्रयोक्ता के बीच संपर्क में होने वाले दोष होते हैं। इसमें इंटरफेस जटिल होना, अस्पष्ट होना या प्लेटफॉर्म पर आधारित होना शामिल हो सकता है, जो इंटरफेस परीक्षण में बाधाएँ उत्पन्न करता है।
6. **प्रदर्शन दोष (Performance Defects)**— यह तब होते हैं जब प्रणाली या सॉफ़्टवेयर अनुप्रयोग अपेक्षित परिणाम देने में असमर्थ होता है। जब प्रणाली प्रयोक्ता की आवश्यकताओं को पूरा नहीं करती, तो वह प्रदर्शन दोष होता है। इसमें यह भी शामिल है कि सिस्टम विभिन्न लोड की स्थिति में कैसे प्रतिक्रिया करता है।
7. **सॉफ़्टवेयर त्रुटि (Software Error)**— यह सॉफ़्टवेयर विकास के दौरान होती है। यह व्याकरणिक त्रुटि, तार्किक त्रुटि या प्रयोक्ता की आवश्यकताओं की गलत व्याख्या के रूप में हो सकती है, जिससे वास्तविक कोड में अपेक्षित परिणाम नहीं मिलते। यह प्रयोक्ता

दस्तावेज़ और सॉफ़्टवेयर की वास्तविक कार्यप्रणाली में मेल न खाने के रूप में भी हो सकती है। यह त्रुटि कोडिंग या परीक्षण के दौरान ज्ञात हो भी सकती है या नहीं भी।

8. **सॉफ़्टवेयर दोष (Software Fault)—** यह उस त्रुटि का परिणाम होता है जो निष्पादनशील प्रोग्राम में बनी रहती है। सभी दोष ज्ञात नहीं होते, और कई बार सॉफ़्टवेयर वर्षों तक बिना किसी स्पष्ट समस्या के काम करता रहता है।
9. **सॉफ़्टवेयर विफलता (Software Failure)—** यह ऐसा दोष है जो एक स्पष्ट समस्या उत्पन्न करता है; इसलिए इसे विफलता कहा जाता है। यह एप्लीकेशन में ऐसी खराबी उत्पन्न करता है जिसे सिस्टम अनुरक्षण की आवश्यकता होती है।

7.3 डिज़ाइन दोषों के सामान्य कारण

1. **अपर्याप्त आवश्यकताओं का विश्लेषण** — प्रयोक्ता की आवश्यकताओं को पूरी तरह से समझने और परिभाषित करने में असफलता ऐसे डिज़ाइन का कारण बन सकती है जो प्रयोक्ता की आवश्यकताओं को पूरा नहीं करते।
2. **पर्याप्त परीक्षण और प्रमाणीकरण की कमी** — परीक्षण और प्रमाणीकरण प्रक्रियाओं को छोड़ना या अपर्याप्त रूप से करना ऐसे दोषों को अनदेखा कर सकता है जो उत्पाद के उपयोग में आने के बाद सामने आते हैं।
3. **विकास समय-सीमा का अत्यधिक दबाव** — निर्धारित समय-सीमा को पूरा करने के दबाव में डिज़ाइन प्रक्रिया में शॉर्टकट अपनाए जा सकते हैं, जिससे समस्याएँ अनदेखी रह जाती हैं।
4. **पारस्परिक कार्यात्मक सहयोग की कमी** — विभिन्न क्षेत्रों के संबंधित हितधारकों और विशेषज्ञों को शामिल न करना डिज़ाइन के महत्वपूर्ण पहलुओं की अनदेखी का कारण बन सकता है।
5. **अपर्याप्त अनुसंधान** — बाज़ार अनुसंधान या प्रौद्योगिकी संबंधी अनुसंधान की कमी ऐसे डिज़ाइनों को जन्म देती है जो वर्तमान प्रवृत्तियों या प्रयोक्ता की पसंद से मेल नहीं खाते।
6. **डिज़ाइन में परिवर्तन** — डिज़ाइन विनिर्देशों में बार-बार परिवर्तन से अंतिम उत्पाद में भ्रम और त्रुटियाँ उत्पन्न हो सकती हैं।
7. **अपर्याप्त परीक्षण वातावरण** — ऐसे वातावरण में परीक्षण करना जो वास्तविक परिस्थितियों का सटीक अनुकरण नहीं करते, कई दोषों को छिपा सकता है।
8. **अपर्याप्त प्रशिक्षण और कौशल स्तर** — आवश्यक कौशल या ज्ञान की कमी रखने वाली डिज़ाइन टीमें कम गुणवत्ता वाले डिज़ाइन का निर्माण कर सकती हैं।
9. **प्रोटोटाइपिंग की अनुपस्थिति** — प्रोटोटाइपिंग चरण को छोड़ने से ऐसे डिज़ाइन बन सकते हैं जिन्हें पर्याप्त रूप से परखा और परिष्कृत नहीं किया गया।

डिज़ाइन दोषों को कम करने के लिए संगठनों को आवश्यकताओं के विस्तृत विश्लेषण, समग्र परीक्षण और सभी हितधारकों के बीच निरंतर सहयोग को प्राथमिकता देनी चाहिए। इसके अतिरिक्त, सतत् सुधार प्रक्रियाएँ और प्रतिपुष्टि तंत्र विकास चक्र के प्रारंभिक चरणों में ही दोषों की पहचान और सुधार करने में सहायक होते हैं।

7.4 सॉफ्टवेयर विकास जीवन चक्र में दोषों के चरण

दोष, जिन्हें बग या समस्याएँ भी कहा जाता है, सॉफ्टवेयर विकास जीवन चक्र (SDLC) के विभिन्न चरणों में उत्पन्न हो सकते हैं। प्रक्रिया के आरंभिक चरणों में ही दोषों की पहचान और निवारण उच्च गुणवत्ता वाले सॉफ्टवेयर उत्पाद प्रदान करने के लिए अत्यंत आवश्यक होता है। नीचे SDLC में वे सामान्य चरण दिए गए हैं जहाँ दोष उत्पन्न और पहचाने जा सकते हैं—

1. **आवश्यकताओं का संकलन और विश्लेषण**
   दोष की उत्पत्ति — आवश्यकताओं की गलतफहमी या गलत संप्रेषण दोष उत्पन्न कर सकता है।
   पहचान — आवश्यकताओं की समीक्षा और हितधारकों के साथ प्रमाणीकरण से समस्याओं का पता लगाया जा सकता है।
2. **सिस्टम डिज़ाइन**

दोष की उत्पत्ति — अधूरी या गलत डिज़ाइन विनिर्देश दोष उत्पन्न कर सकते हैं।
पहचान — समकक्ष समीक्षा (peer review) और डिज़ाइन वॉकथ्रू डिज़ाइन से संबंधित दोषों की पहचान में सहायक होते हैं।

3. **कोडिंग/कार्यान्वयन**
   दोष की उत्पत्ति — लॉजिकल त्रुटियाँ, सिंटैक्स (वाक्यविन्यास) त्रुटियाँ और टाइपो जैसी कोडिंग गलतियाँ इस चरण में उत्पन्न होती हैं।
   पहचान — कोड समीक्षा, स्थैतिक कोड विश्लेषण उपकरण और स्वचालित परीक्षण कोडिंग दोषों की पहचान करने में सहायक होते हैं।
4. **यूनिट परीक्षण**
   दोष की उत्पत्ति — कोड की व्यक्तिगत इकाइयों या घटकों में दोष अनदेखे रह सकते हैं।
   पहचान — सामान्यतः डेवलपर यूनिट परीक्षण करते हैं, और स्वचालित यूनिट परीक्षण अलग-अलग कोड खंडों में दोष ढूँढ़ते हैं।
5. **इंटीग्रेशन (एकीकरण) परीक्षण**
   दोष की उत्पत्ति — घटकों के बीच गलत डेटा एक्सचेंज जैसे एकीकरण मुद्दे दोष उत्पन्न कर सकते हैं।
   पहचान — इंटीग्रेशन परीक्षण विभिन्न घटकों की पारस्परिक क्रिया से उत्पन्न दोषों की पहचान पर केंद्रित होता है।
6. **सिस्टम परीक्षण**
   दोष की उत्पत्ति — प्रणाली स्तर के दोष, जैसे कार्यात्मक और प्रदर्शन से संबंधित समस्याएँ उभर सकती हैं।
   पहचान — सिस्टम परीक्षण पूरे सॉफ़्टवेयर सिस्टम की कार्यक्षमता और प्रदर्शन से जुड़े दोषों की पहचान करता है।
7. **प्रयोक्ता स्वीकृति परीक्षण (UAT)**
   दोष की उत्पत्ति — वास्तविक प्रयोक्ताओं को प्रभावित करने वाले दोष इस चरण में सामने आ सकते हैं।
   पहचान — प्रयोक्ता या हितधारक UAT करते हैं ताकि प्रयोक्ता के दृष्टिकोण से दोषों की पहचान की जा सके।
8. **रिग्रेशन परीक्षण**
   दोष की उत्पत्ति — नए कोड परिवर्तन या सुधार अनजाने में नए दोष उत्पन्न कर सकते हैं या पुराने दोषों को फिर से सक्रिय कर सकते हैं।
   पहचान — स्वचालित रिग्रेशन परीक्षण यह सुनिश्चित करता है कि हालिया परिवर्तनों से मौजूदा कार्यक्षमता प्रभावित न हो।
9. **रिलीज का परिनियोजन (Deployment of Release)**
   दोष की उत्पत्ति — परिनियोजन के दौरान कॉन्फ़िगरेशन संबंधी समस्याएँ, परिनियोजन त्रुटियाँ या अनुकूलता संबंधी समस्याएँ उत्पन्न हो सकती हैं।
   पहचान — सावधानीपूर्वक परिनियोजन योजना, रोलबैक प्रक्रिया और निगरानी से इस चरण में दोषों को पकड़ा जा सकता है।
10. **रख-रखाव और सहायता (Maintenance and Support)**
    दोष की उत्पत्ति — लाइव सिस्टम में किए गए परिवर्तन या अद्यतन नए दोष उत्पन्न कर सकते हैं।
    पहचान — निरंतर निगरानी, ग्राहक प्रतिपुष्टि और सतत परीक्षण से रिलीज के बाद की अवस्था में दोषों का पता लगाया और निवारण किया जा सकता है।

यह ध्यान देना महत्त्वपूर्ण है कि उपरोक्त चरण एक सामान्य SDLC को दर्शाते हैं, जबकि विभिन्न विकास कार्यप्रणालियाँ (जैसे—Agile, Waterfall, DevOps) इन चरणों के क्रम और आवृत्ति में भिन्नता ला सकती हैं। फिर भी, किसी भी कार्यप्रणाली में **मजबूत गुणवत्ता आश्वासन प्रक्रिया** का होना प्रत्येक चरण पर दोषों की पहचान और समाधान के लिए अत्यावश्यक होता है।

7.5 सॉफ़्टवेयर परीक्षण के प्रकार

सॉफ़्टवेयर परीक्षण, सॉफ़्टवेयर विकास की एक महत्त्वपूर्ण प्रक्रिया है, जो किसी सॉफ़्टवेयर अनुप्रयोग में दोषों या समस्याओं की पहचान करने में सहायक होती है। सॉफ़्टवेयर परीक्षण की अनेक विधियाँ और तकनीकें होती हैं, जिनमें से प्रत्येक विभिन्न प्रकार के दोषों की

पहचान के लिए डिज़ाइन की गई होती है। नीचे सॉफ़्टवेयर परीक्षण के कुछ सामान्य प्रकार और वे दोष दिए गए हैं जिनकी वे पहचान करते हैं—

1. **यूनिट परीक्षण** — यह परीक्षण कोड की व्यक्तिगत इकाइयों या घटकों को पृथक रूप से परखने पर केंद्रित होता है। यह किसी विशेष कोड मॉड्यूल में मौजूद दोषों जैसे गलत लॉजिक या सिंटैक्स त्रुटियों की पहचान करता है।
2. **इंटीग्रेशन परीक्षण** — यह परीक्षण एप्लीकेशन के विभिन्न घटकों/मॉड्यूलों के एकीकरण के समय उनके परस्पर कार्य को परखता है। यह उनके आपसी संबंधों से उत्पन्न दोषों की पहचान करता है।
3. **कार्यात्मक परीक्षण** — यह परीक्षण सॉफ़्टवेयर के निर्दिष्ट कार्यों के अनुरूप कार्य करने की पुष्टि करता है। इसका उद्देश्य गलत व्यवहार, कार्यक्षमता की कमी या उपयोगिता संबंधी समस्याओं की पहचान करना होता है।
4. **रिग्रेशन परीक्षण** — इस परीक्षण में सुनिश्चित किया जाता है कि नए कोड परिवर्तन किसी भी पुराने कार्य में दोष उत्पन्न न करें। यह अनजाने में पुनः उत्पन्न हुए दोषों की पहचान में सहायक होता है।
5. **यूज़र इंटरफ़ेस (UI) परीक्षण** — यह परीक्षण सॉफ़्टवेयर के ग्राफिकल यूज़र इंटरफ़ेस (GUI) की जाँच करता है। यह इंटरफ़ेस के लेआउट, रूप और कार्यक्षमता से संबंधित दोषों की पहचान करता है।
6. **प्रदर्शन परीक्षण** — यह परीक्षण सॉफ़्टवेयर की गति, उत्तरदायित्व और विस्तार-क्षमता को परखता है। यह धीमे प्रदर्शन, संसाधन उपयोग या बॉटलनेक्स से जुड़े दोषों की पहचान करता है।
7. **तनाव परीक्षण (Stress Testing)** — यह परीक्षण सॉफ़्टवेयर को उसकी अपेक्षित बॉर्डर्स से परे धकेलता है ताकि अत्यधिक परिस्थितियों में उत्पन्न होने वाले दोषों जैसे सिस्टम क्रैश या मेमोरी लीक को उजागर किया जा सके।
8. **सुरक्षा परीक्षण** — यह परीक्षण सॉफ़्टवेयर की सुरक्षा प्रणालियों की कमजोरियों और खामियों की पहचान करता है। यह डेटा उल्लंघन या अनधिकृत पहुँच से संबंधित दोषों का पता लगाता है।
9. **उपयोगिता परीक्षण (Usability Testing)** — यह परीक्षण सॉफ़्टवेयर की प्रयोक्ता मित्रता और कार्य-संपादन की सरलता को परखता है। इसका उद्देश्य खराब प्रयोक्ता अनुभव से संबंधित दोषों की पहचान करना होता है।
10. **संगतता परीक्षण (Compatibility Testing)** — यह परीक्षण विभिन्न उपकरणों, ब्राउज़रों या ऑपरेटिंग सिस्टमों पर सॉफ़्टवेयर के प्रदर्शन की जाँच करता है। यह प्लेटफ़ॉर्म-विशिष्ट समस्याओं से संबंधित दोषों को उजागर करता है।
11. **सुलभता परीक्षण (Accessibility Testing)** — यह परीक्षण यह सुनिश्चित करता है कि सॉफ़्टवेयर दिव्यांग प्रयोक्ताओं के लिए सुलभ हो। यह पहुँच मानकों का पालन न करने से उत्पन्न दोषों की पहचान करता है।
12. **स्थानीयकरण परीक्षण (Localization Testing)** — यह परीक्षण यह सुनिश्चित करता है कि सॉफ़्टवेयर विभिन्न भाषाओं और संस्कृतियों के अनुसार अनुकूलित है। यह अनुवाद, सांस्कृतिक संवेदनशीलता और स्वरूप से जुड़े दोषों की पहचान करता है।
13. **अल्फा और बीटा परीक्षण** — इन परीक्षणों में वास्तविक प्रयोक्ता नियंत्रित वातावरण (अल्फा) या विस्तृत समूह (बीटा) में सॉफ़्टवेयर का परीक्षण करते हैं। ये परीक्षण ऐसे दोषों की पहचान करते हैं जो आंतरिक परीक्षण के दौरान स्पष्ट नहीं हो पाते।

इनमें से प्रत्येक परीक्षण प्रकार सॉफ़्टवेयर विकास जीवन चक्र में विशिष्ट भूमिका निभाता है और सॉफ़्टवेयर उत्पाद की गुणवत्ता तथा विश्वसनीयता सुनिश्चित करने में सहायक होता है। किस प्रकार के परीक्षण का चयन किया जाए, यह परियोजना की आवश्यकताओं, लक्ष्यों और बॉर्डर्स पर निर्भर करता है।

अभ्यास 7.5

- सॉफ़्टवेयर परीक्षण के प्रकारों की सूची बनाइए।
- डिज़ाइन दोषों के सामान्य कारणों की सूची बनाइए।
- सॉफ़्टवेयर विकास में दोषों के प्रकारों की सूची बनाइए।

सारांश

- डिज़ाइन दोष सुरक्षा जोखिम, अक्षमता और उपभोक्ता असंतोष का कारण बन सकते हैं।
- सामान्य डिज़ाइन दोषों में अपर्याप्त सुरक्षा उपाय, खराब एर्गोनॉमिक्स और जटिलता शामिल हैं।
- सॉफ़्टवेयर दोषों के प्रकारों में गणितीय, तर्कात्मक, सिंटैक्स और मल्टीथ्रेडिंग दोष सम्मिलित हैं।
- अपर्याप्त आवश्यकताओं का विश्लेषण और जल्दबाज़ी में किया गया विकास डिज़ाइन दोषों के सामान्य कारण हैं।
- विकास जीवन चक्र के विभिन्न चरणों में दोष उत्पन्न हो सकते हैं।
- सॉफ़्टवेयर परीक्षण के प्रकारों में यूनिट, इंटीग्रेशन, परफॉर्मेंस और यूज़ेबिलिटी परीक्षण शामिल हैं।
- संगतता, सुरक्षा और अभिगम्यता परीक्षण दोष पहचान के लिए महत्वपूर्ण हैं।
- अल्फा और बीटा परीक्षण क्रमशः नियंत्रित और व्यापक वातावरण में वास्तविक प्रयोक्ताओं को शामिल करते हैं।
- सॉफ़्टवेयर विकास प्रक्रिया के प्रारंभिक चरण में दोषों की पहचान और समाधान उच्च गुणवत्ता वाले उत्पाद प्रदान करने के लिए आवश्यक है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न (MCQs)

1. डिज़ाइन दोष सुरक्षा जोखिम और अक्षमता सहित विभिन्न समस्याओं का कारण बन सकते हैं। इस अध्याय में उल्लिखित एक सामान्य डिज़ाइन दोष क्या है?
   (क) अधिक लागत
   (ख) प्रयोक्ता संतोष
   (ग) संगतता
   (घ) स्थिरता

2. कौन-से दोष गणितीय अभिव्यक्तियों में त्रुटियों से संबंधित होते हैं और प्रायः कोड की भीड़ के कारण उत्पन्न होते हैं?
   (क) गणितीय दोष
   (ख) तर्कात्मक दोष
   (ग) सिंटैक्स दोष
   (घ) इंटरफ़ेस दोष
3. किस प्रकार का दोष तब उत्पन्न होता है जब प्रोग्रामर समस्या को स्पष्ट रूप से नहीं समझता या गलत तरीके से सोचता है?
   (क) गणितीय दोष
   (ख) तर्कात्मक दोष
   (ग) सिंटैक्स दोष
   (घ) मल्टीथ्रेडिंग दोष

4. सॉफ़्टवेयर विकास जीवन चक्र (SDLC) के किस चरण में कोडिंग त्रुटियों और टाइपिंग की ग़लतियों से संबंधित दोष उत्पन्न होते हैं?
   (क) आवश्यकताओं का संकलन और विश्लेषण
   (ख) प्रणाली डिज़ाइन
   (ग) कोडिंग/कार्यान्वयन
   (घ) इंटीग्रेशन परीक्षण

5. कौन-सा परीक्षण विभिन्न घटकों/मॉड्यूल्स के आपसी इंटरैक्शन से संबंधित दोषों को उजागर करने पर केंद्रित होता है?
   (क) यूनिट परीक्षण
   (ख) इंटीग्रेशन परीक्षण
   (ग) फंक्शनल परीक्षण
   (घ) सुरक्षा परीक्षण

6. कौन-सा परीक्षण सॉफ़्टवेयर के ग्राफिकल यूज़र इंटरफ़ेस (GUI) के लेआउट, स्वरूप और कार्यक्षमता से संबंधित दोषों की जाँच करता है?
   (क) रिग्रेशन परीक्षण
   (ख) यूज़ेबिलिटी परीक्षण
   (ग) सुरक्षा परीक्षण
   (घ) यूआई परीक्षण

7. नए कोड परिवर्तनों या सुधारों द्वारा अनजाने में पुनः लाए गए पुराने दोषों की पहचान किस प्रकार के परीक्षण से होती है?
   (क) यूनिट परीक्षण
   (ख) इंटीग्रेशन परीक्षण
   (ग) फंक्शनल परीक्षण
   (घ) रिग्रेशन परीक्षण

8. यूज़ेबिलिटी परीक्षण सॉफ़्टवेयर के किस आयाम का मूल्यांकन करता है?
   (क) गति और प्रत्युत्तर समय
   (ख) सुरक्षा कमजोरियाँ
   (ग) प्रयोक्ता अनुकूलता और प्रयोग में आसानी
   (घ) प्लेटफ़ॉर्म विशिष्ट समस्याएँ

9. सुरक्षा परीक्षण का उद्देश्य क्या पहचानना होता है?
   (क) गणितीय दोष
   (ख) प्रणाली विफलता
   (ग) डेटा उल्लंघन और कमजोरियाँ
   (घ) इंटरफ़ेस दोष

10. कौन-सा दोष निष्पादित कार्यक्रम में विद्यमान होता है, लेकिन तुरंत पहचान में नहीं आता और तत्काल समस्या उत्पन्न नहीं करता?
    (क) सॉफ़्टवेयर त्रुटि
    (ख) सॉफ़्टवेयर दोष
    (ग) सॉफ़्टवेयर विफलता
    (घ) सिंटैक्स दोष

ख. रिक्त स्थान भरें

1. _____ वे दोष होते हैं जो गणितीय अभिव्यक्तियों में डेवलपर द्वारा की गई गलतियों या समाधान में त्रुटियों के कारण होते हैं।

2. लॉजिकल या तर्कात्मक दोष तब उत्पन्न होते हैं जब प्रोग्रामर समस्या को स्पष्ट रूप से नहीं समझता या _____ रूप से सोचता है।
3. सिंटैक्स दोष कोड की लेखन _____ में की गई गलतियों से संबंधित होते हैं।
4. अपर्याप्त आवश्यकताओं का विश्लेषण ऐसे डिज़ाइन उत्पन्न कर सकता है जो प्रयोक्ता की _____ को पूरा नहीं करते।
5. इंटीग्रेशन परीक्षण उन दोषों की पहचान पर केंद्रित होता है जो विभिन्न _____ के इंटरैक्शन के दौरान उत्पन्न होते हैं।
6. सुरक्षा परीक्षण डेटा उल्लंघन और _____ पहुँच से संबंधित दोषों की पहचान में सहायक होता है।
7. नए कोड परिवर्तनों द्वारा उत्पन्न दोषों की पहचान _____ परीक्षण से की जाती है।
8. परफॉर्मेंस परीक्षण सॉफ़्टवेयर की गति, प्रत्युत्तर क्षमता और _____ का मूल्यांकन करता है।
9. संगतता परीक्षण यह जाँचता है कि सॉफ़्टवेयर विभिन्न उपकरणों, _____ या ऑपरेटिंग सिस्टम पर कैसा कार्य करता है।
10. दोषों की _____ उच्च गुणवत्ता वाले सॉफ़्टवेयर उत्पाद प्रदान करने के लिए आवश्यक होती है।

ग. सही या गलत बताइए

1. डिज़ाइन दोष सुरक्षा जोखिम जैसी समस्याएँ उत्पन्न कर सकते हैं, लेकिन वे प्रयोक्ता संतोष को प्रभावित नहीं करते।
2. तर्कात्मक दोष तब होते हैं जब प्रोग्रामर समस्या को पूरी तरह समझ लेता है।
3. यूज़ेबिलिटी परीक्षण सॉफ़्टवेयर के प्रदर्शन और विस्तारशीलता का मूल्यांकन करता है।
4. सुरक्षा परीक्षण का उद्देश्य सॉफ़्टवेयर की सुरक्षा प्रणालियों में कमजोरियों और त्रुटियों की पहचान करना होता है।
5. स्ट्रेस परीक्षण, परफॉर्मेंस परीक्षण का एक प्रकार है।
6. इंटरफ़ेस दोष गणितीय अभिव्यक्तियों में त्रुटियों से संबंधित होते हैं।
7. अल्फा और बीटा परीक्षणों में सॉफ़्टवेयर को विकास चरण के दौरान वास्तविक प्रयोक्ताओं द्वारा परखा जाता है।
8. अपर्याप्त आवश्यकताओं का विश्लेषण ऐसे डिज़ाइन उत्पन्न करता है जो प्रयोक्ता आवश्यकताओं को पूरी तरह पूरा करते हैं।
9. संगतता परीक्षण में यह जाँचा जाता है कि सॉफ़्टवेयर विभिन्न उपकरणों, ब्राउज़रों या ऑपरेटिंग सिस्टम पर कैसा प्रदर्शन करता है।
10. नए कोड परिवर्तनों द्वारा उत्पन्न दोषों की हमेशा तुरंत पहचान और समाधान हो जाता है।

घ. लघु उत्तर प्रश्न

1. डिज़ाइन दोषों के क्या संभावित परिणाम हो सकते हैं?
2. प्रयोक्ता को असुविधा पहुँचाने वाला एक सामान्य डिज़ाइन दोष बताइए।
3. तर्कात्मक दोष अन्य सॉफ़्टवेयर दोषों से किस प्रकार भिन्न होते हैं?
4. डिज़ाइन दोषों पर अपर्याप्त आवश्यकताओं के विश्लेषण का क्या प्रभाव होता है?
5. सॉफ़्टवेयर विकास जीवन चक्र के किस चरण में आवश्यकताओं के संकलन के दौरान दोष उत्पन्न हो सकते हैं?
6. इंटीग्रेशन परीक्षण का उद्देश्य क्या है और यह किस प्रकार के दोषों को उजागर करता है?
7. सॉफ़्टवेयर दोषों की पहचान में परफॉर्मेंस परीक्षण की क्या भूमिका होती है?
8. दोषों की पहचान में सुरक्षा परीक्षण क्यों आवश्यक है?
9. यूज़ेबिलिटी परीक्षण किन प्रकार के दोषों की पहचान करने में सहायक होता है?
10. अल्फा और बीटा परीक्षण में क्या अंतर है और ये दोषों की पहचान में कैसे सहायक हैं?

सत्र 8
डिज़ाइन दोषों का समाधान

आइए मिलते हैं अंकित से, जो LEGO से अवसंरचना बनाना और चित्र बनाना बहुत पसंद करता था। लेकिन कभी-कभी उसमें छोटी-छोटी गलतियाँ रह जाती थीं। इसलिए, उसने "डिज़ाइन दोषों का समाधान करें" के बारे में सीखा। अंकित ने खोजा कि यह एक समस्या-समाधानकर्ता बनने जैसा था—

त्रुटि को पहचानना— जैसे किसी पहेली में कोई टुकड़ा गायब हो, उसने ध्यानपूर्वक देखा कि कहाँ क्या गलत था। **समाधान के बारे में सोचना**— जैसे पहेली को पूरा करने का तरीका खोजते हैं, वैसे ही अंकित ने अपनी कल्पना का उपयोग करके त्रुटि को सुधारने के तरीके खोजे।

त्रुटि को सुधारना— जैसे पहेली का गायब टुकड़ा ठीक जगह पर रखा जाता है, वैसे ही उसने अपने निर्माण को और बेहतर बनाने के लिए आवश्यक परिवर्तन किए।

अंकित ने इन चरणों का पालन करके अपने LEGO निर्माण और चित्रों को बेहतर बनाया, जैसा कि **चित्र 8.1** में दिखाया गया है। यह जैसे कि वह अपनी रचनाओं के लिए एक सुपरहीरो बन गया हो, जो उन्हें और भी शानदार बना रहा था।

चित्र 8.1— अंकित डिज़ाइन दोषों को सुधारते हुए

इस अध्याय में, आप **डिज़ाइन दोषों का समाधान करने**, **भविष्य के डिज़ाइन में सुधार की संभावनाओं**, **कोडिंग उपकरणों**, तथा **डिज़ाइन दोषों के अभिलेखन और प्रलेखन** को समझेंगे।

8.1 डिज़ाइन दोषों का समाधान करना

डिज़ाइन दोषों को हल करना उत्पाद विकास प्रक्रिया का एक महत्वपूर्ण भाग है। ये दोष कार्यक्षमता, उपयोगिता और यहाँ तक कि सुरक्षा से संबंधित समस्याओं का कारण बन सकते हैं। डिज़ाइन दोषों को प्रभावी रूप से हल करने के लिए आप निम्नलिखित चरणों का पालन कर सकते हैं—

1. **दोषों की पहचान करें**— पहला चरण है सभी डिज़ाइन दोषों की पहचान करना और उन्हें प्रलेखित करना। यह कार्य उपयोगिता परीक्षण, कोड समीक्षाओं और ग्राहक प्रतिक्रिया जैसे विभिन्न तरीकों से किया जा सकता है। सुनिश्चित करें कि आपके पास दोषों की एक व्यापक सूची हो।

2. **दोषों को प्राथमिकता दें**— सभी दोष उत्पाद पर उनके प्रभाव के दृष्टिकोण से समान नहीं होते। उनकी गंभीरता और संभावित परिणामों के आधार पर प्राथमिकता दें। जो दोष सुरक्षा या संरक्षा को प्रभावित करते हैं, उन्हें तुरंत ठीक किया जाना चाहिए, जबकि कम गंभीर दोषों को बाद में शेड्यूल किया जा सकता है।
3. **मूल कारण विश्लेषण करें**— प्रत्येक दोष के लिए मूल कारण विश्लेषण करें ताकि यह समझा जा सके कि वह क्यों उत्पन्न हुआ। इसके लिए दोष को उसके स्रोत तक ट्रैक करना पड़ता है—चाहे वह कोडिंग त्रुटि हो, डिज़ाइन चरण में संप्रेषण में चूक हो या कोई अन्य कारण। यह चरण भविष्य में समान दोषों को रोकने के लिए आवश्यक है।
4. **पारस्परिक टीमों की भागीदारी करें**— डिज़ाइन दोषों को हल करने के लिए अक्सर विभिन्न टीमों जैसे डिज़ाइन, विकास, गुणवत्ता आश्वासन और उत्पाद प्रबंधन से इनपुट और सहयोग की आवश्यकता होती है। सुनिश्चित करें कि सभी संबंधित हितधारक इस प्रक्रिया में शामिल हों।
5. **डिज़ाइन में परिवर्तन करें**— यदि दोष स्वयं डिज़ाइन से संबंधित है, तो डिज़ाइन टीम के साथ मिलकर आवश्यक संशोधन करें। इसमें नए डिज़ाइन मॉकअप, वायरफ्रेम या प्रोटोटाइप बनाना शामिल हो सकता है, ताकि समस्याओं को संबोधित किया जा सके।
6. **कोड संबंधी सुधार करें**— यदि दोष कोड से संबंधित हैं, तो डेवलपर्स कोडिंग की सर्वोत्तम प्रथाओं का पालन करते हुए समस्याओं को ठीक करें। इसमें डिबगिंग, कोड का पुनर्लेखन (refactoring) या कुछ भागों को फिर से लिखना शामिल हो सकता है।
7. **परीक्षण करें**— डिज़ाइन और कोड में परिवर्तन करने के बाद, पूर्ण परीक्षण करें ताकि यह सुनिश्चित किया जा सके कि दोषों का समाधान हो गया है। इसमें यूनिट परीक्षण, इंटीग्रेशन परीक्षण और प्रयोक्ता स्वीकृति परीक्षण शामिल हैं।
8. **पुनरावृत्त परीक्षण (Regression Testing)**— ध्यान दें कि एक दोष को ठीक करने से कभी-कभी नए दोष उत्पन्न हो सकते हैं या मौजूदा कार्यक्षमता प्रभावित हो सकती है। इसलिए, पुनरावृत्त परीक्षण करें ताकि यह सुनिश्चित किया जा सके कि उत्पाद के अन्य भाग अप्रभावित रहें।
9. **प्रलेखन**— प्रयोक्ता पुस्तिका, डिज़ाइन विनिर्देश और कोड टिप्पणियों जैसे प्रलेखनों को अद्यतन करें ताकि उनमें किए गए परिवर्तनों का उल्लेख हो।
10. **प्रयोक्ता प्रतिक्रिया**— यदि संभव हो तो वास्तविक प्रयोक्ताओं या बीटा परीक्षकों को सत्यापन प्रक्रिया में शामिल करें। उनसे प्रतिक्रिया प्राप्त करें ताकि यह सुनिश्चित हो सके कि दोष का समाधान उनकी अपेक्षाओं के अनुरूप है और इससे कोई नई उपयोगिता समस्या उत्पन्न नहीं हुई है।
11. **संचार**— दोष समाधान की प्रगति के बारे में सभी हितधारकों को सूचित रखें। पारदर्शिता और स्पष्ट संचार आवश्यक हैं ताकि अपेक्षाओं का प्रबंधन किया जा सके और विश्वास बना रहे।
12. **निगरानी**— दोषों को हल करने के बाद भी उत्पाद की निगरानी करते रहें। कभी-कभी वास्तविक दुनिया के परिवेश में ही समस्याएँ स्पष्ट होती हैं, और उनका शीघ्र समाधान करना आवश्यक होता है।
13. **निरंतर सुधार**— दोष समाधान से प्राप्त अनुभवों का उपयोग अपने डिज़ाइन और विकास प्रक्रियाओं में सुधार हेतु करें। ऐसी रोकथाम संबंधी उपायों को लागू करें जिससे भविष्य में ऐसे दोषों की संभावना को कम किया जा सके।

याद रखें कि डिज़ाइन दोषों का समाधान एक दोहरावीय प्रक्रिया है, जिसमें पहचान, विश्लेषण और सुधार के कई चक्र लग सकते हैं। नियमित रूप से अपनी विकास और डिज़ाइन प्रक्रियाओं की समीक्षा और सुधार करने से दोषों की संभावना को प्रारंभिक स्तर पर ही न्यूनतम किया जा सकता है।

8.2 भविष्य के डिज़ाइन में सुधार की संभावनाएँ

भविष्य के डिज़ाइन में सुधार की संभावनाएँ व्यापक और सतत् विकसित होती जा रही हैं, क्योंकि डिज़ाइन अनुशासन नई तकनीकों, प्रवृत्तियों और प्रयोक्ता आवश्यकताओं के अनुरूप अनुकूलित होते जा रहे हैं। कुछ प्रमुख क्षेत्र जहाँ डिज़ाइनर भविष्य में अपने कार्य को बेहतर बनाने पर ध्यान केंद्रित कर सकते हैं, निम्नलिखित हैं—

1. **प्रयोक्ता केंद्रित डिज़ाइन**— डिज़ाइन में प्रयोक्ता की आवश्यकताओं और प्राथमिकताओं को सर्वोपरि रखना अनिवार्य है। अपने लक्षित प्रयोक्ताओं के साथ गहन अनुसंधान, उपयोगिता परीक्षण और फीडबैक विश्लेषण करके ऐसे डिज़ाइन बनाएं जो वास्तव में उनके साथ जुड़ाव बना सकें।
2. **सुलभता (Accessibility)**— डिज़ाइन करते समय सभी के लिए पहुँच सुनिश्चित करना आवश्यक है। सुनिश्चित करें कि आपका डिज़ाइन समावेशी है और पहुँच मानकों (जैसे WCAG) का पालन करता है ताकि विकलांग व्यक्ति भी उसका प्रयोग कर सकें।
3. **मोबाइल और उत्तरदायी डिज़ाइन**— जैसे-जैसे मोबाइल उपकरणों का प्रयोग बढ़ रहा है, डिज़ाइनरों को विभिन्न स्क्रीन आकारों और उपकरणों पर सहजता से कार्य करने वाले उत्तरदायी डिज़ाइन बनाने में निपुण होना चाहिए।
4. **डेटा-संचालित डिज़ाइन**— डिज़ाइन निर्णयों को बेहतर बनाने के लिए डेटा विश्लेषण और प्रयोक्ता व्यवहार की जानकारी का उपयोग करें। A/B परीक्षण और प्रयोक्ता डेटा के आधार पर डिज़ाइन को समय के साथ सुधारें और अनुकूलित करें।
5. **स्थिरता (Sustainability)**— डिज़ाइन विकल्पों के पर्यावरणीय प्रभाव पर विचार करें। स्थायी डिज़ाइन का उद्देश्य अपशिष्ट को कम करना, पर्यावरण-हितैषी सामग्रियों का उपयोग करना और न्यूनतम कार्बन उत्सर्जन वाले उत्पाद बनाना है।
6. **मानव-केंद्रित एआई डिज़ाइन**— एआई और मशीन लर्निंग के बढ़ते उपयोग के साथ, डिज़ाइनरों को ऐसे एआई-आधारित इंटरफेस तैयार करने चाहिए जो सहज और प्रयोक्ता अनुभव को बेहतर बनाते हों, न कि उसे प्रतिस्थापित करते हों।
7. **नैतिक डिज़ाइन**— सुनिश्चित करें कि आपके डिज़ाइन प्रयोक्ता की गोपनीयता का सम्मान करते हैं और नैतिक सिद्धांतों का पालन करते हैं। ऐसे डिज़ाइन तत्वों से बचें जो प्रयोक्ताओं को भ्रमित या प्रलोभित करें।
8. **सहयोग और अनुकूलनशीलता**— डेवलपर्स, विपणन विशेषज्ञों और उत्पाद प्रबंधकों सहित क्रॉस-फंक्शनल टीमों के साथ प्रभावी तरीके से सहयोग करें। जो डिज़ाइनर दूसरों के साथ मिलकर कार्य कर सकते हैं, वे अधिक सफल उत्पाद बनाते हैं। साथ ही, नई तकनीकों और डिज़ाइन प्रवृत्तियों के साथ स्वयं को अद्यतन रखना अनिवार्य है।
9. **निरंतर अधिगम**— डिज़ाइन एक सतत् परिवर्तनशील क्षेत्र है। कार्यशालाओं, सम्मेलनों में भाग लेकर तथा डिज़ाइन ब्लॉग और प्रकाशनों से जुड़े रहकर आजीवन अधिगम के प्रति प्रतिबद्ध रहें।

डिज़ाइन में सुधार की संभावनाएँ असीमित हैं, और यह आवश्यक है कि हम जिज्ञासु बने रहें, परिवर्तन के लिए खुले रहें, और डिज़ाइन की उत्कृष्टता की बॉर्डर्स को आगे बढ़ाने हेतु नए विचारों और तकनीकों के साथ प्रयोग करने को तत्पर रहें।

8.3 कोडिंग उपकरण (Coding Tools)

कोडिंग उपकरण ऐसे सॉफ़्टवेयर प्रोग्राम और यूटिलिटीज़ होते हैं जो डेवलपर्स को अधिक दक्षता से कोड लिखने, डिबग करने और प्रबंधित करने में सहायता करते हैं। ये उपकरण प्रोग्रामर के लिए उनके कार्य प्रवाह को सुव्यवस्थित करने और उच्च गुणवत्ता वाला सॉफ़्टवेयर तैयार करने में अत्यावश्यक होते हैं। कोडिंग के लिए कई प्रकार के उपकरण उपलब्ध हैं, जिनमें से प्रत्येक का एक विशिष्ट उद्देश्य होता है। नीचे कोडिंग उपकरणों की कुछ सामान्य श्रेणियाँ दी गई हैं—

1. एकीकृत विकास परिवेश (Integrated Development Environments - IDEs)—

Visual Studio Code — यह Microsoft द्वारा विकसित एक अत्यधिक अनुकूलन योग्य, मुक्त और मुक्त-स्रोत कोड एडिटर है। यह विभिन्न प्रोग्रामिंग भाषाओं और एक्सटेंशनों को समर्थन देता है।

PyCharm — विशेष रूप से Python विकास के लिए डिज़ाइन किया गया एक IDE, जो कोड विश्लेषण, डिबगिंग और संस्करण नियंत्रण जैसी विशेषताएँ प्रदान करता है।

2. कोड एडिटर (Code Editors)—

Sublime Text — यह एक हल्का किंतु शक्तिशाली टेक्स्ट एडिटर है जो अपनी गति और विस्तारक्षमता के लिए जाना जाता है।
Atom — GitHub द्वारा विकसित एक ओपन-सोर्स, अनुकूलन योग्य टेक्स्ट एडिटर है जिसमें कस्टमाइज़ेशन और समुदाय-संचालित पैकेजों पर बल दिया गया है।

3. संस्करण नियंत्रण प्रणाली (Version Control Systems - VCS)—

Git — यह एक वितरित संस्करण नियंत्रण प्रणाली है जिसका उपयोग सॉफ़्टवेयर विकास के दौरान स्रोत कोड में परिवर्तनों को ट्रैक करने के लिए किया जाता है।

GitHub — Git रिपॉज़िटरी को होस्ट करने और सहयोग के लिए एक वेब-आधारित मंच।

4. कोड सहयोग और साझाकरण (Code Collaboration and Sharing)—

GitHub/GitLab/Bitbucket — कोड रिपॉज़िटरी को होस्ट करने, समस्याओं का प्रबंधन करने और कोड समीक्षाएँ करने हेतु प्लेटफ़ॉर्म।

Slack — एक टीम संचार उपकरण जो कोडिंग प्लेटफ़ॉर्म्स के साथ एकीकृत होकर विकास टीमों को समन्वय में रखता है।

5. डिबगिंग उपकरण (Debugging Tools)—

GDB — यह GNU डिबगर है, जिसका उपयोग C, C++ और अन्य भाषाओं के लिए डिबगिंग में किया जाता है।

Xcode — macOS और iOS ऐप विकास के लिए Apple का एकीकृत विकास परिवेश, जिसमें डिबगिंग टूल्स भी शामिल हैं।

6. पैकेज प्रबंधक (Package Managers)—

Npm (Node Package Manager) — यह JavaScript पैकेजों के प्रबंधन और वितरण के लिए प्रयुक्त होता है।
Pip — यह Python के लिए एक पैकेज प्रबंधक है।

7. कंटेनरीकरण और वर्चुअलाइजेशन (Containerization and Virtualization)—

Docker — कंटेनरों में एप्लिकेशन विकसित करने, भेजने और चलाने का एक मंच।

VirtualBox — स्थानीय सिस्टम पर वर्चुअल मशीनें चलाने के लिए एक ओपन-सोर्स वर्चुअलाइजेशन टूल।

8. टर्मिनल के लिए टेक्स्ट एडिटर (Text Editors for Terminal)—

Vim — एक अत्यधिक अनुकूलन योग्य, टेक्स्ट-आधारित टेक्स्ट एडिटर, जिसे टर्मिनल वातावरण में कोडिंग के लिए व्यापक रूप से उपयोग किया जाता है।

Emacs — एक अन्य अत्यंत विस्तार योग्य टेक्स्ट एडिटर, जो कोडिंग और अन्य टेक्स्ट-आधारित कार्यों के लिए प्रयुक्त होता है।

9. कार्य निष्पादक और निर्माण उपकरण (Task Runners and Build Tools)—

Grunt — एक JavaScript टास्क रनर जो वेब विकास में दोहराए जाने वाले कार्यों को स्वचालित करता है।
Webpack — एक लोकप्रिय JavaScript मॉड्यूल बंडलर और निर्माण उपकरण।

10. कोड लिंटर्स और फॉर्मेटर्स (Code Linters and Formatters)—

ESLint — JavaScript कोड में समस्याओं की पहचान और सुधार के लिए एक लिंटर।

Prettier — एक कोड फॉर्मेटर जो परियोजनाओं में सुसंगत कोड शैली को लागू करता है।

11. डेटाबेस प्रबंधन उपकरण (Database Management Tools)—

phpMyAdmin — MySQL डेटाबेस के प्रबंधन के लिए एक वेब-आधारित टूल।

Sequel Pro — macOS के लिए MySQL डेटाबेस प्रबंधन हेतु एक एप्लिकेशन।

12. सतत एकीकरण/वितरण उपकरण(Continuous Integration/Continuous Deployment - CI/CD Tools)—

Jenkins — कोड को बिल्ड, टेस्ट और डिप्लॉय करने के लिए एक ओपन-सोर्स स्वचालन सर्वर।

Travis CI — कोड के परीक्षण और डिप्लॉयमेंट को स्वचालित करने के लिए एक क्लाउड-आधारित CI/CD सेवा।

13. टेक्स्ट और दस्तावेज़ीकरण निर्माण (Text and Documentation Generation)—

Doxygen — एनोटेटेड स्रोत कोड से दस्तावेज़ उत्पन्न करने के लिए एक उपकरण। Javadoc — Java के लिए एक दस्तावेज़ निर्माण उपकरण।

कोडिंग उपकरणों का चयन आपके द्वारा प्रयुक्त प्रोग्रामिंग भाषाओं, आपके विशिष्ट विकास आवश्यकताओं और व्यक्तिगत प्राथमिकताओं पर निर्भर करता है। कई डेवलपर इन उपकरणों के संयोजन का उपयोग करके दक्ष और उत्पादक कार्य प्रवाह तैयार करते हैं।

8.4 डिज़ाइन दोषों का अभिलेखन और प्रलेखन (Recording and Documentation of Design Defects)

डिज़ाइन दोषों का अभिलेखन और प्रलेखन उत्पाद विकास और गुणवत्ता आश्वासन प्रक्रिया का एक महत्वपूर्ण भाग है। उपयुक्त प्रलेखन यह सुनिश्चित करने में सहायता करता है कि दोषों की पहचान, ट्रैकिंग और समाधान प्रभावी तरीके से किया जा सके। डिज़ाइन दोषों को रिकॉर्ड और डॉक्युमेंट करते समय निम्नलिखित चरणों पर विचार किया जाना चाहिए—

1. पहला चरण डिज़ाइन दोष की पहचान और समझ होता है। इसमें परीक्षण करना, प्रदर्शन डेटा का विश्लेषण करना या प्रयोक्ताओं अथवा टीम के सदस्यों द्वारा रिपोर्ट की गई समस्याओं का अवलोकन करना शामिल हो सकता है।
2. बुनियादी जानकारी रिकॉर्ड करें, जैसे—दोष का स्पष्ट और संक्षिप्त विवरण दें, इसके लक्षण, प्रभाव तथा यह निर्दिष्ट करें कि डिज़ाइन में यह दोष कहाँ प्रकट होता है।
3. जहाँ संभव हो, उस प्रमाण को शामिल करें जो दोष की उपस्थिति को सिद्ध करता हो। इसमें स्क्रीनशॉट्स, लॉग फ़ाइलें या परीक्षण परिणाम शामिल हो सकते हैं।
4. उस डिज़ाइन या उत्पाद के संस्करण को निर्दिष्ट करें जिसमें दोष पाया गया है। यह यह ट्रैक करने के लिए आवश्यक है कि क्या इस दोष को बाद के संस्करणों में सुधारा गया है।
5. समान प्रकार के दोषों को श्रेणियों या वर्गों में समूहित करें ताकि विश्लेषण और प्राथमिकता निर्धारण में सहायता मिल सके। सामान्य दोष श्रेणियों में कार्यात्मक, प्रदर्शन-सम्बंधी, सुरक्षा और प्रयोज्यता दोष शामिल हैं।
6. दोषों को रिकॉर्ड और प्रबंधित करने के लिए किसी दोष ट्रैकिंग प्रणाली या सॉफ़्टवेयर का उपयोग करें। लोकप्रिय उपकरणों में JIRA, Bugzilla या आपके संगठन की आवश्यकताओं के अनुरूप बनाए गए कस्टम सिस्टम शामिल हैं।
7. प्रत्येक दोष की स्थिति को उसके प्रारंभिक खोज से लेकर उसके समाधान तक ट्रैक करें। सामान्य स्थिति टैग्स में "ओपन", "इन प्रोग्रेस", "रिज़ॉल्व्ड", और "क्लोज्ड" शामिल होते हैं।
8. दोषों को उनकी गंभीरता और प्रभाव के आधार पर प्राथमिकता दें। ऐसे गंभीर दोष जो सुरक्षा जोखिम उत्पन्न करते हैं या कार्यक्षमता पर गंभीर प्रभाव डालते हैं उन्हें तत्काल संबोधित किया जाना चाहिए, जबकि कम प्राथमिकता वाले दोषों को भविष्य के संस्करणों के लिए अनुसूचित किया जा सकता है।
9. जब कोई दोष ठीक कर दिया जाता है तो समाधान का विवरण प्रलेखित करें, जिसमें कोड में किए गए परिवर्तन, किए गए परीक्षण और वह संस्करण शामिल हो जिसमें यह सुधार लागू किया गया है।
10. किसी दोष के समाधान के पश्चात यह सुनिश्चित करें कि इसे पूर्ण रूप से पुनः परीक्षण किया गया है ताकि यह सत्यापित किया जा सके कि समस्या पूरी तरह से हल हो गई है और कोई नई समस्या उत्पन्न नहीं हुई है। जब किसी दोष को "रिज़ॉल्व्ड" घोषित कर सभी आवश्यक परीक्षणों में वह सफल सिद्ध हो जाए, तो उसे दोष ट्रैकिंग प्रणाली में "क्लोज्ड" किया जा सकता है।
11. नियमित रूप से दोष रिपोर्ट तैयार करें और संबंधित हितधारकों के साथ साझा करें ताकि दोष प्रबंधन प्रयासों की स्थिति स्पष्ट रूप से प्रदर्शित हो सके।
12. पश्चात विश्लेषण (Post-Mortem) या मूल कारण विश्लेषण (Root Cause Analysis) करें ताकि यह समझा जा सके कि दोष क्यों उत्पन्न हुए और भविष्य की डिज़ाइन या विकास परियोजनाओं में ऐसे दोषों की पुनरावृत्ति को कैसे रोका जा सकता है। दोष ट्रैकिंग से प्राप्त आंकड़ों और अंतर्दृष्टियों का उपयोग डिज़ाइन और विकास प्रक्रियाओं में निरंतर सुधार हेतु किया जाना चाहिए।

इन चरणों का पालन करते हुए और विस्तृत अभिलेख बनाए रखते हुए, संगठन डिज़ाइन दोषों का प्रभावी तरीके से प्रबंधन और शमन कर सकते हैं, जिससे उत्पाद की गुणवत्ता में वृद्धि और उपभोक्ता संतुष्टि में सुधार सुनिश्चित किया जा सकता है।

अभ्यास 8.1

- कोडिंग उपकरणों की सामान्य श्रेणियाँ सूचीबद्ध कीजिए।
- डिज़ाइन दोषों को हल करने के तीन चरण सूचीबद्ध कीजिए।

सारांश

- डिज़ाइन दोषों की पहचान करना और उनका प्रलेखन करना समाधान प्रक्रिया का पहला महत्त्वपूर्ण चरण है।
- दोषों की गंभीरता के आधार पर प्राथमिकता निर्धारित करने से प्रभावी समाधान में सहायता मिलती है।
- दोष क्यों उत्पन्न हुए, यह समझने और भविष्य में उनके दोहराव रोकने के लिए उनके मूल कारण का विश्लेषण आवश्यक होता है।
- डिज़ाइन, विकास और गुणवत्ता आश्वासन सहित पारकार्यात्मक टीमों के साथ सहयोग अत्यावश्यक है।
- डिज़ाइन दोषों को दूर करने के लिए डिज़ाइन में परिवर्तन और कोड सुधार किए जाते हैं।
- पुनः परीक्षण (Regression Testing) सहित गहन परीक्षण यह सुनिश्चित करता है कि दोषों का समाधान हो गया है और नए मुद्दे उत्पन्न नहीं हो रहे हैं।
- दस्तावेज़ों में अद्यतन परिवर्तन यह दर्शाते हैं कि किन बदलावों के माध्यम से दोषों को हल किया गया।
- प्रयोक्ता प्रतिक्रिया और सतत निगरानी, दोष समाधान की पुष्टि करते हैं।
- हितधारकों के साथ स्पष्ट संवाद और नियमित दोष रिपोर्टिंग अत्यंत महत्वपूर्ण है।
- दोष समाधान प्रक्रिया चक्रीय होती है और सतत सुधार इसकी कुंजी है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न (MCQs)

1. अनसुलझे डिज़ाइन दोषों के संभावित परिणाम क्या हो सकते हैं?
   (क) ग्राहक संतुष्टि में वृद्धि
   (ख) कार्यात्मकता में सुधार
   (ग) प्रयोज्यता में कमी
   (घ) सुरक्षा मानकों में गिरावट

2. डिज़ाइन दोषों के समाधान में मूल कारण विश्लेषण का उद्देश्य क्या है?
   (क) दोषों के लिए व्यक्तियों को दोषी ठहराना
   (ख) यह समझना कि दोष क्यों उत्पन्न हुए और समान समस्याओं को रोकना
   (ग) दोषों के स्रोत को छुपाना
   (घ) दोषों की गंभीरता के आधार पर प्राथमिकता तय करना

3. डिज़ाइन दोषों को हल करने में किन टीमों को शामिल किया जाना चाहिए?
   (क) केवल डिज़ाइन टीम
   (ख) विकास और गुणवत्ता आश्वासन टीमें
   (ग) केवल उत्पाद प्रबंधन टीम
   (घ) किसी अन्य टीम की आवश्यकता नहीं है

4. डिज़ाइन और कोड में परिवर्तन करने के बाद किन प्रकार के परीक्षण किए जाते हैं?
   (क) यूनिट परीक्षण और इंटीग्रेशन परीक्षण
   (ख) केवल यूनिट परीक्षण
   (ग) केवल प्रयोक्ता स्वीकृति परीक्षण
   (घ) कोई परीक्षण आवश्यक नहीं है

5. दोष समाधान के दौरान किस प्रकार के दस्तावेज़ों को अद्यतन किया जाना चाहिए?
   (क) बग रिपोर्ट
   (ख) बैठक कार्यवृत्त
   (ग) प्रयोक्ता पुस्तिकाएँ, डिज़ाइन विनिर्देश, और कोड टिप्पणियाँ
   (घ) कोई दस्तावेज़ अद्यतन आवश्यक नहीं है

6. भविष्य की डिज़ाइन में सुधार की दृष्टि से, डिज़ाइनरों को किसे प्राथमिकता देनी चाहिए?
   (क) प्रयोक्ता अनुसंधान और प्रतिक्रिया से बचना
   (ख) स्थिर रहना और नई तकनीकों के अनुकूल न होना
   (ग) प्रयोक्ता केंद्रित डिज़ाइन और प्रयोज्यता परीक्षण पर ध्यान केंद्रित करना
   (घ) पहुंच योग्य मानकों की उपेक्षा करना

7. सतत डिज़ाइन (Sustainable Design) किस पर केंद्रित होता है?
   (क) अपशिष्ट और कार्बन उत्सर्जन को अधिकतम करना
   (ख) अपशिष्ट को कम करना और पर्यावरण-अनुकूल सामग्री का उपयोग करना
   (ग) पर्यावरणीय प्रभाव की उपेक्षा करना
   (घ) स्थिरता की तुलना में सौंदर्यशास्त्र को प्राथमिकता देना

8. एआई डिज़ाइन का एक महत्वपूर्ण पहलू क्या है?
   (क) एआई से मानव अनुभव को प्रतिस्थापित करना
   (ख) प्रयोक्ता अनुभव को बेहतर बनाने के लिए एआई-संचालित इंटरफ़ेस बनाना
   (ग) डिज़ाइन में एआई से पूरी तरह बचना
   (घ) नैतिक डिज़ाइन

9. डिज़ाइनरों के लिए पारकार्यात्मक टीमों के साथ सहयोग क्यों महत्त्वपूर्ण है?
   (क) यह डिज़ाइन प्रक्रिया को धीमा करता है
   (ख) डिज़ाइन की गुणवत्ता पर कोई प्रभाव नहीं डालता
   (ग) प्रभावी सहयोग से अक्सर अधिक सफल उत्पाद बनते हैं
   (घ) पारकार्यात्मक टीमों के साथ सहयोग आवश्यक नहीं है

10. दोष समाधान के बाद भी निरंतर निगरानी क्यों महत्त्वपूर्ण है?
   (क) यह अनावश्यक है
   (ख) डिज़ाइन चरण में नए दोष पकड़ने के लिए
   (ग) कुछ समस्याएँ केवल वास्तविक दुनिया के वातावरण में ही स्पष्ट होती हैं
   (घ) दोषों के लिए विकास टीम को दोषी ठहराने हेतु

ख. रिक्त स्थान भरें

1. डिज़ाइन दोषों का समाधान _____ विकास प्रक्रिया का एक महत्वपूर्ण भाग है।

2. डिज़ाइन दोषों को हल करने की पहली प्रक्रिया है सभी दोषों की _____ और प्रलेखन करना।
3. गंभीरता के आधार पर दोषों को प्राथमिकता देना, सबसे महत्त्वपूर्ण समस्याओं को _____ हेतु आवश्यक है।
4. डिज़ाइन और कोड परिवर्तन के बाद, दोषों को सुनिश्चित करने हेतु गहन परीक्षण _____ किया जाता है।
5. पुनरावृत्त परीक्षण यह सुनिश्चित करने के लिए महत्त्वपूर्ण है कि दोष समाधान से कोई नया दोष उत्पन्न न हो या मौजूदा _____ न टूटे।
6. दोष समाधान के बाद भी उत्पादन में उत्पाद की निरंतर _____ आवश्यक है।
7. दोषों को हल करने से प्राप्त अनुभव का उपयोग अपनी डिज़ाइन और _____ प्रक्रियाओं को सुधारने हेतु करें।
8. डिज़ाइन में सुधार की सीमा असीमित है, इसलिए नए विचारों और तकनीकों के साथ _____ आवश्यक है।
9. डिज़ाइन दोषों का उपयुक्त दस्तावेज़ीकरण यह सुनिश्चित करता है कि दोषों की पहचान, ट्रैकिंग और _____ प्रभावी तरीके से हो सके।
10. यदि कोई दोष डिज़ाइन से संबंधित है, तो आवश्यक _____ करने के लिए डिज़ाइन टीम के साथ कार्य करना आवश्यक है।

ग. सही या गलत बताइए

1. डिज़ाइन दोषों का समाधान उत्पाद विकास प्रक्रिया का आवश्यक भाग नहीं है।
2. सभी डिज़ाइन दोषों की पहचान और उनका प्रलेखन समाधान प्रक्रिया के लिए आवश्यक नहीं है।
3. प्रभावी दोष समाधान हेतु दोषों को गंभीरता के आधार पर प्राथमिकता देना आवश्यक नहीं है।
4. यह समझने हेतु कि दोष क्यों उत्पन्न हुए और भविष्य में उन्हें रोकने के लिए मूल कारण विश्लेषण आवश्यक नहीं है।
5. सतत डिज़ाइन अपशिष्ट को कम करने और पर्यावरण-अनुकूल सामग्री के उपयोग पर केंद्रित होता है।
6. नैतिक डिज़ाइन में डार्क पैटर्न और भ्रामक प्रथाओं का उपयोग किया जाता है जो प्रयोक्ताओं को धोखा देते हैं।
7. पारकार्यात्मक टीमों के साथ सहयोग अक्सर अधिक सफल उत्पादों का निर्माण करता है।
8. डिज़ाइनरों को अपने कार्य में सुधार के लिए जिज्ञासु बने रहना और नए विचारों व तकनीकों के साथ प्रयोग करना टालना चाहिए।
9. एंड-यूज़र्स या बीटा परीक्षणकर्ताओं को मान्यता प्रक्रिया में शामिल करना यह सुनिश्चित करने के लिए आवश्यक नहीं है कि दोष समाधान उनकी अपेक्षाओं के अनुरूप है।
10. दोष समाधान के बाद उत्पाद की उत्पादन अवस्था में निरंतर निगरानी आवश्यक नहीं होती।

घ. लघु उत्तर प्रश्न

1. डिज़ाइन दोषों को हल करने का पहला चरण क्या है?
2. गंभीरता के आधार पर दोषों को प्राथमिकता देना क्यों आवश्यक है?
3. दोष समाधान में मूल कारण विश्लेषण का क्या महत्त्व है?
4. पारकार्यात्मक टीमों की भागीदारी प्रभावी दोष समाधान में किस प्रकार योगदान देती है?
5. डिज़ाइन से संबंधित दोषों को हल करने हेतु प्रमुख क्रियाएँ क्या होती हैं?
6. कोड से संबंधित दोषों को डेवलपर्स किस प्रकार संबोधित करते हैं?
7. दोष समाधान में गहन परीक्षण, विशेषतः पुनरावृत्त परीक्षण, क्यों आवश्यक होता है?
8. दोष समाधान प्रक्रिया के दौरान दस्तावेज़ों को किन प्रकारों में अद्यतन किया जा सकता है?
9. दोष समाधान के प्रमाणीकरण में प्रयोक्ता प्रतिक्रिया क्यों महत्त्वपूर्ण है?
10. डिज़ाइन दोषों के प्रबंधन में सतत सुधार की क्या भूमिका है?

मॉड्यूल 2

HTML और CSS

परिचय

इस मॉड्यूल में, आप सर्वप्रथम "HTML" के बारे में सीखेंगे, जिसमें HTML, HTML एलिमेंट्स, HTML गुण (Attributes), और डाक्यूमेंट संरचना सम्मिलित हैं। इसके बाद, हम "सूचियों की अवधारणा — अननुक्रमित सूची (Unordered Lists), अनुक्रमित सूची (Ordered Lists), परिभाषा सूची (Definition List)" की ओर बढ़ेंगे, जिसमें आप अननुक्रमित सूचियाँ, अनुक्रमित सूचियाँ, परिभाषा सूचियाँ, छवियाँ (Image) एवं इमेज मैपिंग (Image Mapping) तथा HTML हाइपरलिंक (Hyperlink) के बारे में जानेंगे। फिर, हम "वेबपृष्ठ में फ्रेम (Frame in Webpage)" पर चर्चा करेंगे, जिसमें फ्रेम, इसके लाभ एवं सीमाएँ, ऊर्ध्वाधर कॉलम (Vertical Columns) बनाना, वेबपृष्ठ में क्लाइंट-साइड फॉर्म बनाना तथा XHTML सम्मिलित है।

इसके पश्चात हम "CSS" की चर्चा करेंगे, जिसमें आप CSS, CSS के संस्करण, CSS सिंटैक्स, CSS के प्रकार, बहुविकल्पी स्टाइल शीट्स (Multiple Style Sheets) तथा शैली तत्वों की पृष्ठभूमि (Background) और बॉर्डर्स (Borders) को समझेंगे। अंततः, हम "CSS बॉक्स मॉडल" पर चर्चा करेंगे, जिसमें CSS बॉक्स मॉडल, CSS आयाम (Dimensions), CSS द्वारा प्रदर्शन स्थिति निर्धारण (Display Positioning), CSS दृश्यता (Visibility), CSS प्रदर्शन (Display) और CSS स्क्रॉलबार्स सम्मिलित हैं। यह मॉड्यूल आपको HTML में आवश्यक ज्ञान एवं कौशल से सुसज्जित करती है।

सत्र 9

HTML

आइए मिलते हैं अंकित से, जो इंटरनेट को बहुत पसंद करता है। वह यह जानना चाहता था कि वेबपृष्ठ कैसे कार्य करते हैं, इसलिए उसने "HTML" के बारे में सीखा। HTML उसके लिए एक जादुई नुस्खे (Recipe) की तरह था, जिसमें विशेष टैग "< >" होते थे, जो वेबपृष्ठ पर चीज़ों को उसी प्रकार सजाने में मदद करते थे जैसे कोई अपने खिलौने सजाता है। अंकित ने इसका उपयोग करके अपना वेबपृष्ठ बनाया, उसमें चित्र, कहानियाँ और लिंक जोड़े। यह एक डिजिटल कलाकार और कथावाचक बनने जैसा अनुभव था। HTML के माध्यम से उसने रोचक वेबपृष्ठ बनाए जिनके द्वारा वह अपने विचार और रचनात्मकता दूसरों से साझा कर सका — ठीक वैसे ही जैसे कोई चित्र बनाता है या कहानी लिखता है, जैसा कि चित्र 9.1 में दिखाया गया है।

चित्र 9.1— HTML सीखता हुआ अंकित

इस अध्याय में, आप HTML, HTML एलिमेंट्स, HTML गुण (Attributes), और डाक्यूमेंट संरचना के बारे में सीखेंगे।

9.1 HTML

HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज) एक टेक्स्ट-आधारित विधि है, जिसका उपयोग HTML दस्तावेज़ के अंदर सामग्री के संगठन को परिभाषित करने हेतु किया जाता है। यह एक मार्कअप भाषा है जो वेबपृष्ठों की रूपरेखा (Layout) और विषयवस्तु (Content) को परिभाषित करने में अनिवार्य है। प्रायः CSS (कैस्केडिंग स्टाइल शीट्स) और JavaScript के साथ प्रयुक्त होने वाला HTML, इंटरैक्टिव और दृश्य रूप से आकर्षक वेबसाइटें विकसित करने में एक महत्वपूर्ण भूमिका निभाता है।

मार्कअप निर्देशों का उपयोग करके, HTML वेब ब्राउज़र को यह निर्देश देता है कि किसी वेबपृष्ठ पर टेक्स्ट, चित्र, और विभिन्न मल्टीमीडिया तत्वों को कैसे प्रदर्शित किया जाए। इसका मुख्य कार्य वेबपृष्ठ की सामग्री को संरचित करना होता है, जिसमें इंटरैक्टिविटी और गत्यात्मक कार्यक्षमता जोड़ने हेतु स्क्रिप्टिंग — जैसे कि JavaScript — का विकल्प होता है।

HTML एक मान्यता प्राप्त मानक है जिसे **वर्ल्ड वाइड वेब कंसॉर्शियम (W3C)** द्वारा स्थापित किया गया है और इसे डेस्कटॉप व मोबाइल प्लेटफ़ॉर्म दोनों पर प्रमुख वेब ब्राउज़रों द्वारा व्यापक रूप से स्वीकार किया जाता है। विशेष रूप से, **HTML5** इसका नवीनतम संस्करण है।

क्या आप जानते हैं?
HTML का पूर्ण रूप **Hyper Text Markup Language** है। यह एक मार्कअप भाषा है जिसका उपयोग वेबपृष्ठ बनाने हेतु किया जाता है।

9.2 HTML एलिमेंट्स

HTML दस्तावेज़ विभिन्न एलिमेंट्स से मिलकर बने होते हैं, जो किसी वेबपृष्ठ की रूपरेखा और विषयवस्तु को परिभाषित करते हैं। ये एलिमेंट्स कोणीय ब्रैकेट्स (< >) के अंदर संलग्न होते हैं और सामान्यतः जोड़े के रूप में आते हैं — जिसमें एक प्रारंभिक टैग (Opening Tag) और एक समापन टैग (Closing Tag) होता है।

उदाहरण—

This is a paragraph.

Visit Example

HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज) एलिमेंट्स की एक विस्तृत श्रृंखला प्रदान करता है जो वेबपृष्ठों पर सामग्री को प्रारूपित (Format) और प्रदर्शित (Display) करने की सुविधा देते हैं। ये एलिमेंट्स किसी भी HTML दस्तावेज़ के मूलभूत घटक होते हैं। नीचे HTML के सामान्यतः प्रयुक्त कुछ प्रमुख एलिमेंट्स दिए गए हैं —

1. टेक्स्ट तत्व (Text Elements)—

 — पैराग्राफ परिभाषित करता है।

<h1> से <h6> — विभिन्न स्तरों (1 से 6 तक) की शीर्षक-पंक्तियाँ।

<a> — हाइपरलिंक बनाता है।

 और — टेक्स्ट को ज़ोर देना या प्रमुखता देने के लिए।

<blockquote> — उद्धृत किए गए टेक्स्ट का ब्लॉक परिभाषित करता है।

<code> — कोड को टेक्स्ट के रूप में दर्शाने हेतु।

<pre> — पूर्व-स्वरूपित टेक्स्ट (जैसे कोड ब्लॉक)।

<sub> और <sup> — सबस्क्रिप्ट और सुपरस्क्रिप्ट टेक्स्ट के लिए।

2. सूचियाँ (Lists)—

 — अननुक्रमित (बुलेटेड) सूची।

 — अनुक्रमित (क्रमांकित) सूची।

 — सूची का एकल आइटम, <ul> और <ol> के अंदर प्रयुक्त।

3. कंटेनर एलिमेंट्स (Container Elements)—

 — ब्लॉक स्तर का कंटेनर एलिमेंट।

 — इनलाइन स्तर का कंटेनर एलिमेंट।

4. लिंक और एंकर (Links and Anchors)—

<a> — हाइपरलिंक बनाता है।

<link> — वर्तमान डाक्यूमेंट और बाहरी संसाधनों (जैसे स्टाइलशीट) के बीच संबंध निर्दिष्ट करता है।

<nav> — नेविगेशन लिंक की अनुभागीय संरचना को दर्शाता है।

5. छवियाँ और मल्टीमीडिया (Images and Multimedia)—

<img> — वेबपृष्ठ में चित्र सम्मिलित करता है।

<audio> — ऑडियो सामग्री सम्मिलित करता है।

<video> — वीडियो सामग्री सम्मिलित करता है।

<iframe> — बाहरी वेब सामग्री (जैसे मानचित्र या अन्य वेबसाइट) को एम्बेड करता है।

6. फ़ॉर्म्स (Forms)—

<form> — प्रयोक्ता से इनपुट प्राप्त करने हेतु HTML फ़ॉर्म बनाता है।

<input> — विभिन्न प्रकार के इनपुट फ़ील्ड (जैसे टेक्स्ट, पासवर्ड, रेडियो बटन, चेकबॉक्स आदि)।

<textarea> — बहु-पंक्तियों वाला टेक्स्ट इनपुट क्षेत्र।

<select> — ड्रॉप-डाउन सूची।

<button> — क्लिक करने योग्य बटन।

<label> — फ़ॉर्म के तत्वों के लिए लेबल।

<option> — रेडियो बटन या ड्रॉप-डाउन के विकल्प।

7. तालिकाएँ (Tables)—

<table> — तालिका परिभाषित करता है।

<thead> — तालिका का शीर्षक भाग।

<tbody> — तालिका का मुख्य भाग।

<tfoot> — तालिका का अंतिम भाग (पाद)।

<tr> — तालिका की पंक्ति।

<th> — तालिका की शीर्षक-कोशिका।

<td> — तालिका की डेटा-कोशिका।

<caption> — तालिका के लिए शीर्षक या कैप्शन।

8. सेमान्टिक एलिमेंट्स (Semantic Elements — HTML5 में प्रारंभ)—

<header> — प्रस्तावना सामग्री हेतु कंटेनर।

<footer> — समापन सामग्री हेतु कंटेनर।

<nav> — नेविगेशन लिंक की अनुभागीय संरचना।

<section> — विषयवस्तु का खंड।

<article> — आत्म-निहित संरचना।

<aside> — सहायक या आस-पास की जानकारी।

<main> — डाक्यूमेंट का मुख्य भाग।

<figure> — सामान्यतः चित्र या चार्ट जैसी मॉड्यूल।

<figcaption> — <figure> का शीर्षक या विवरण।

9. मेटा जानकारी (Meta Information)—

<head> — दस्तावेज़ की मेटा-सूचना (जैसे शीर्षक, लिंक आदि)।

<meta> — वर्णमाला एन्कोडिंग जैसी मेटा जानकारी।

<title> — वेबपृष्ठ का शीर्षक।

10. टिप्पणियाँ (Comments)—

<!-- ... --> — HTML कोड में टिप्पणियाँ जोड़ने के लिए। ये वेबपृष्ठ पर प्रदर्शित नहीं होतीं।

ये HTML के कुछ मूल एलिमेंट्स हैं। HTML अत्यंत विस्तारशील (Extensible) है और डेवलपर्स वेब कॉम्पोनेंट्स के माध्यम से कस्टम एलिमेंट्स भी बना सकते हैं। किस एलिमेंट का उपयोग करना है, यह आपके वेबपृष्ठ की सामग्री, संरचना और अर्थवत्ता (Semantics) पर निर्भर करता है। इन एलिमेंट्स का उचित उपयोग वेबपृष्ठ की पहुँच, SEO और अनुरक्षण में सहायक होता है।

अभ्यास 9.1

निम्नलिखित HTML अवयवों को परिभाषित कीजिए —

• टेक्स्ट (Text)

- चित्र और मल्टीमीडिया (Images and Multimedia)
- फॉर्म (Forms)
- तालिकाएँ (Tables)
- लिंक और एंकर (Links and Anchors)

9.3 HTML गुण (Attributes)

गुण (Attributes) HTML अवयवों में अतिरिक्त विवरण जोड़ने या उनके व्यवहार को संशोधित करने का एक तरीका हैं। इन्हें किसी अवयव के आरंभिक टैग के अंदर निर्दिष्ट किया जाता है और ये सामान्यतः नाम–मूल्य (name-value) युग्म के रूप में होते हैं। HTML अवयवों के साथ गुणों का प्रायः उपयोग अतिरिक्त जानकारी प्रदान करने या उनकी कार्यक्षमता को समायोजित करने के लिए किया जाता है। कुछ सामान्य गुण निम्नलिखित हैं —

src — स्रोत URL को निर्दिष्ट करता है (उपयोग <img>, <script>, <a> आदि में)।

href — लक्ष्य URL को निर्दिष्ट करता है (उपयोग <a>, <link> आदि में)।

alt — छवियों के लिए वैकल्पिक टेक्स्ट प्रदान करता है (उपयोग <img> में)।

class — CSS वर्ग को निर्दिष्ट करता है, जिससे स्टाइलिंग संभव होती है।

id — अवयव के लिए एक अद्वितीय पहचानकर्ता प्रदान करता है।

style — इनलाइन CSS शैलियाँ परिभाषित करता है।

placeholder — इनपुट फ़ील्ड के लिए प्लेसहोल्डर टेक्स्ट निर्धारित करता है।

disabled — फॉर्म अवयवों को निष्क्रिय करता है।

title — HTML अवयव का शीर्षक प्रदान करता है।

required — निर्दिष्ट करता है कि किसी फ़ॉर्म फ़ील्ड को भरना अनिवार्य है।

HTML टैग और गुणों को मिलाकर वेब सामग्री की संरचना और स्वरूप तय किए जाते हैं, इंटरैक्टिव फॉर्म बनाए जाते हैं, तथा वेब पृष्ठ के व्यवहार और रूप को परिभाषित किया जाता है। इन अवयवों और गुणों को समझना वेब विकास के लिए अत्यंत महत्वपूर्ण है।

<img src="image.jpg" alt="Description of the image">

Visit Example

9.4 दस्तावेज़ संरचना (Document Structure)

एक HTML दस्तावेज़ एक विशिष्ट संरचना का पालन करता है —

<!DOCTYPE html> — यह दस्तावेज़ के प्रकार और संस्करण को सूचित करता है, जैसे कि HTML5। किसी HTML दस्तावेज़ की शुरुआत करने के लिए डॉक्युमेंट टाइप डिक्लेरेशन (DTD) को सम्मिलित करना उपयुक्त होता है। यह घोषणा यह निर्दिष्ट करती है कि आप किस HTML संस्करण का उपयोग कर रहे हैं, और यह दस्तावेज़ की संरचना और व्याख्या की नींव निर्धारित करती है।

<html> — <html> तत्व (एलिमेंट) HTML दस्तावेज़ का मूल तत्व होता है, जो पृष्ठ पर उपस्थित सभी अन्य तत्वों को समाहित करता है। सामान्यतः इसमें दो अवयव सम्मिलित होते हैं— <head> और <body>।

<head> — <head> अनुभाग दस्तावेज़ से संबंधित मेटा-जानकारी को स्टोर करता है, जैसे शीर्षक, अक्षर एन्कोडिंग, और बाहरी संसाधनों के संदर्भ। यह जानकारी वेब पृष्ठ पर दृश्य नहीं होती, परंतु खोज इंजनों और ब्राउज़रों के लिए महत्त्वपूर्ण होती है। <head> अनुभाग में सामान्यतः निम्नलिखित अवयव पाए जाते हैं—

- <meta> — मेटाडेटा जैसे अक्षर एन्कोडिंग और viewport सेटिंग्स को परिभाषित करता है।
- <title> — वेब पृष्ठ का शीर्षक निर्धारित करता है, जो ब्राउज़र की शीर्ष पट्टी या टैब में प्रदर्शित होता है।
- <link> — बाहरी संसाधनों जैसे स्टाइलशीट (CSS) और वेब फ़ॉन्ट से लिंक करता है।
- <script> — JavaScript फ़ाइलों को लिंक करता है या उन्हें सम्मिलित करता है।
- <style> — आंतरिक CSS शैलियों को समाहित करता है।

<body> — वेब पृष्ठ की दृश्य सामग्री को समाहित करता है। <body> अनुभाग में आपके वेब पृष्ठ की मुख्य दृश्य सामग्री होती है, जैसे टेक्स्ट, चित्र, फॉर्म और इंटरैक्टिव अवयव। यहीं पर आप HTML तत्वों जैसे शीर्षक (headings), अनुच्छेद (paragraphs), सूचियाँ (lists), लिंक (links) आदि का उपयोग करके पृष्ठ की संरचना बनाते हैं।

अभ्यास 9.2

1. निम्नलिखित HTML गुणों को परिभाषित कीजिए —

(क) class

(ख) id

(ग) style

2. निम्नलिखित HTML दस्तावेज़-विशिष्ट संरचनाओं को परिभाषित कीजिए —

(क) <meta>

(ख) <link>

(ग) <script>

9.5 HTML टैग बनाम तत्व (HTML Tags vs Elements)

HTML टैग और HTML तत्व आपस में संबंधित किन्तु भिन्न अवधारणाएँ हैं। HTML के साथ प्रभावी रूप से कार्य करने हेतु इनके बीच का अंतर समझना अत्यंत आवश्यक है।

9.5.1 HTML टैग (Tags)

परिभाषा— HTML टैग किसी HTML दस्तावेज़ के निर्माण खंड होते हैं, जो किसी HTML तत्व (एलिमेंट) की शुरुआत और अंत को दर्शाते हैं। ये टैग कोणीय ब्रैकेट (< >) में लिखे जाते हैं और सामान्यतः युग्म के रूप में होते हैं—एक आरंभिक टैग और एक समापन टैग। आरंभिक टैग तत्व (एलिमेंट) का नाम बताता है, जबकि समापन टैग में वही नाम होता है परंतु उसके पहले एक तिरछी रेखा (/) होती है। उदाहरण के लिए, और टैग का प्रयोग एक अनुच्छेद तत्व (एलिमेंट) को परिभाषित करने हेतु किया जाता है।

उद्देश्य— टैग ब्राउज़र को यह निर्देश देते हैं कि टैग के बीच स्थित सामग्री को किस प्रकार प्रदर्शित किया जाए। वे वेब पृष्ठ को संरचना और स्वरूप प्रदान करने के निर्देश होते हैं।

9.5.2 HTML तत्व (Elements)

परिभाषा— HTML तत्व एक आरंभिक टैग, एक समापन टैग और उनके बीच की सामग्री (यदि हो) से मिलकर बनते हैं। एक HTML तत्व में टैग और टैग के बीच की समस्त सामग्री सम्मिलित होती है। उदाहरण के लिए — Hello, World! एक HTML तत्व है जो "Hello, World!" नामक टेक्स्ट को समाहित करता है।

उद्देश्य— HTML तत्व एक HTML दस्तावेज़ के मौलिक घटक होते हैं, जो वेब पृष्ठ की सामग्री और संरचना को आकार देने में महत्वपूर्ण भूमिका निभाते हैं। इनमें शीर्षक, अनुच्छेद, चित्र, लिंक आदि जैसे कई आवश्यक तत्व सम्मिलित होते हैं। साथ ही, इन्हें गुणों (Attributes) के माध्यम से और अधिक जानकारी से समृद्ध किया जा सकता है।

सारांश— HTML टैग वे व्यक्तिगत मार्कअप घटक होते हैं जो किसी HTML तत्व की शुरुआत और अंत को चिन्हित करते हैं, जबकि HTML तत्व में टैग और टैग के बीच की सामग्री दोनों सम्मिलित होती हैं।

भेद को स्पष्ट करने हेतु कुछ उदाहरण —

• **टैग**— <h1> (आरंभिक टैग) और </h1> (समापन टैग)

  तत्व— <h1>Welcome to my website!</h1>

• **टैग**— (आरंभिक टैग) और (समापन टैग)

  तत्व— Visit Example

HTML के साथ कार्य करते समय, आप HTML तत्वों का निर्माण और उनका हेरफेर करेंगे, जिसमें टैग और उनकी सामग्री दोनों सम्मिलित होते हैं, ताकि आप अपने वेब पृष्ठ की सामग्री को प्रभावी रूप से संरचित और प्रस्तुत कर सकें। इसका एक सरल उदाहरण एक पूर्ण HTML दस्तावेज़ की मूल संरचना के रूप में **चित्र 9.2** में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Web Page</title>
    <link rel="stylesheet" href="styles.css">
    <script src="myscript.js"></script>
</head>
<body>
    <header>
        <h1>Welcome to My Website</h1>
        <nav>
            <!-- Navigation Links go here -->
        </nav>
    </header>
    <main>
        <article>
            <h2>Article Title</h2>
            <p>This is the content of the article.</p>
        </article>
        <!-- More content goes here -->
    </main>
    <footer>
        <p>&copy; 2023 My Website</p>
    </footer>
</body>
</html>
```

चित्र 9.2— मूल संरचना वाला एक पूर्ण HTML दस्तावेज़

उत्पादन (Output)

चित्र 9.3 — एक बुनियादी संरचना वाले पूर्ण HTML दस्तावेज़ का आउटपुट

9.6 HTML के मूल टैग्स

HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज) वेब पृष्ठ की सामग्री को संरचित करने के लिए विभिन्न प्रकार की मूल टैग्स का उपयोग करती है। ये टैग्स वेब दस्तावेज़ों को बनाने के लिए मूलभूत निर्माण खंड (building blocks) हैं।

1. <!DOCTYPE html> — उस दस्तावेज़ के प्रकार और उपयोग किए गए HTML संस्करण को निर्दिष्ट करता है। HTML5 के लिए यह घोषणा इस प्रकार होती है—
   <!DOCTYPE html>
2. <html> — यह मूल (root) तत्व (एलिमेंट) होता है जो पृष्ठ के सभी अन्य तत्वों को सम्मिलित करता है। इसमें <head> और <body> खंड शामिल होते हैं।
   <html>
   <!-- सामग्री यहाँ आती है -->
   </html>
3. <head> — इस खंड में दस्तावेज़ के बारे में मेटाडेटा होता है, जैसे कि अक्षर एन्कोडिंग, शीर्षक, तथा बाहरी संसाधनों (जैसे स्टाइलशीट्स व स्क्रिप्ट्स) के लिए लिंक।
   <head>
   <!-- मेटाडेटा और लिंक यहाँ होते हैं -->
   </head>
4. <title> — यह टैग वेब पृष्ठ का शीर्षक निर्धारित करता है, जो ब्राउज़र के शीर्षक पट्टी या टैब में प्रमुख रूप से प्रदर्शित होता है।
   <title>My Web Page</title>
5. <meta> — यह टैग दस्तावेज़ के मेटाडेटा को निर्दिष्ट करता है, जैसे कि अक्षर एन्कोडिंग और व्यूपोर्ट सेटिंग्स।
   <meta charset="UTF-8">
6. <body> — इस टैग में वेब पृष्ठ की दृश्य सामग्री होती है, जैसे कि टेक्स्ट, चित्र, लिंक आदि।
   <body>
   <!-- सामग्री यहाँ आती है -->
   </body>
7. <h1>, <h2>, <h3>, <h4>, <h5>, <h6> — ये शीर्षकों के विभिन्न स्तरों को दर्शाते हैं, जहाँ <h1> सबसे बड़ा और <h6> सबसे छोटा होता है।
   <h1>मुख्य शीर्षक</h1>
   <h2>उप-शीर्षक</h2>
8. — यह टैग किसी अनुच्छेद को परिभाषित करता है।
   यह एक अनुच्छेद है।

9.7 HTML फॉर्मेटिंग (Formatting) टैग्स

HTML में कई प्रकार के फॉर्मेटिंग टैग्स प्रदान किए जाते हैं, जो वेब पृष्ठ के अंदर टेक्स्ट और सामग्री की दृश्य शैली (visual styling) को नियंत्रित करने की सुविधा प्रदान करते हैं। नीचे कुछ सामान्यतः प्रयुक्त HTML फॉर्मेटिंग टैग्स दिए गए हैं—

टेक्स्ट फॉर्मेटिंग (Text Formatting)—

<b> — टेक्स्ट को मोटा (bold) बनाता है।

 — महत्वपूर्ण (strong importance) टेक्स्ट को दर्शाता है, और इसे आम तौर पर मोटे अक्षरों में प्रस्तुत किया जाता है।

<i> — टेक्स्ट को तिरछा (italic) करता है।

 — बलाघातयुक्त (एम्फेसाइज्ड emphasized) टेक्स्ट को दर्शाता है, जो सामान्यतः इटेलिक में दिखाया जाता है।

<u> — टेक्स्ट को रेखांकित (underline) करता है।

<s> या <strike> — टेक्स्ट पर काटने की रेखा (स्ट्राइकथ्रू strikethrough) दिखाता है।

<sup> — टेक्स्ट को अधिलेख (सुपरस्क्रिप्ट superscript) बनाता है (रेखाबिंदु से ऊपर उठाया गया)।

<sub> — टेक्स्ट को अधोलिखित (सबस्क्रिप्ट subscript) बनाता है (रेखाबिंदु से नीचे)।

कोड उदाहरण—

This is bold and italic text.

This is ^{superscript} and _{subscript} text.

फॉन्ट शैली और आकार (Font Style and Size) *(अनुशंसित नहीं; इसके लिए CSS का उपयोग करें)*

 — फॉन्ट का आकार, रंग और प्रकार निर्धारित करता है।

कोड उदाहरण—

Styled text

टेक्स्ट संरेखण (Text Alignment एलाइनमेंट)—

 — स्टाइलिंग उद्देश्यों हेतु विभाजन अथवा खंड को परिभाषित करता है, और टेक्स्ट संरेखण नियंत्रित करने के लिए उपयोग में लाया जा सकता है।

 — किसी तत्व (एलिमेंट) के अंदर टेक्स्ट और सामग्री को केंद्र में रखता है (HTML5 में अप्रचलित; इसके स्थान पर CSS का प्रयोग करें)।

कोड उदाहरण— Centered text

पंक्ति विभाजन एवं रिक्त स्थान (Line Breaks and Whitespace)—

•
 — टेक्स्ट के अंदर एक नई पंक्ति जोड़ता है।

• <pre> — पूर्व-स्वरूपित (preformatted) टेक्स्ट, जिसमें रिक्त स्थान और पंक्ति विभाजन संरक्षित रहते हैं।

कोड उदाहरण— This is some text.
Here is a new line.

<pre>
This text
has
preserved
line breaks.
</pre>

उद्धरण (Quotations)—

<blockquote> — एक बड़े उद्धरण खंड को परिभाषित करता है।

<q> — एक संक्षिप्त इनलाइन उद्धरण को परिभाषित करता है।

कोड उदाहरण—

<blockquote>
This is a blockquote.
</blockquote>

He said, <q>Quote me!</q>

संक्षेप और लघु रूप (Abbreviations and Acronyms)—

<abbr> — एक संक्षेप या लघुरूप को दर्शाता है, जिसमें शीर्षक (title) विशेषता द्वारा पूर्ण रूप प्रदान किया जा सकता है।

कोड उदाहरण— <abbr title="World Wide Web">WWW</abbr> is amazing!

ये कुछ मूल HTML फॉर्मेटिंग टैग्स हैं। यद्यपि HTML का उपयोग सरल फॉर्मेटिंग के लिए किया जा सकता है, किन्तु अधिक उन्नत और प्रतिक्रियाशील (responsive) शैली प्रायः CSS (कैस्केडिंग स्टाइल शीट्स) द्वारा प्राप्त की जाती है। CSS वेब सामग्री की प्रस्तुति पर अधिक नियंत्रण प्रदान करता है और स्टाइलिंग को सामग्री से अलग करता है, जिससे वेब पृष्ठों का प्रबंधन अधिक लचीला और दीर्घकालिक रूप से सुविधाजनक हो जाता है — जैसा कि **चित्र 9.4** में प्रदर्शित है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formatting Example</title>
</head>
<body>
    <h1>Text Formatting</h1>
    <p>This is a <b>bold</b> and <i>italic</i> text. Also <u>underline</u> and <s>strikethrough</s> text.</p>
    <p>This is <sup>superscript</sup> and <sub>subscript</sub> text.</p>

    <h1>Font Style and Size</h1>
    <p><font size="4" color="red" face="Arial">Styled text using &lt;font&gt;</font></p>

    <h1>Text Align</h1>
    <div style="text-align: center;">This text is centered using inline styling</div>
    <center>This text is centered using center (deprecated)</center>

    <h1>Line breaks and whitespaces</h1>
    <p>This is some text. <br> Here is a new line.</p>

    <pre>
        This text
        has
        preserved
        line breaks.
    </pre>

    <h1>Quotations</h1>
    <blockquote>
        This is a blockquote.
    </blockquote>
    <p>He said, <q>Quote me!</q></p>

    <h1>Abbrevitations and Acronyms</h1>
    <p><abbr title="World Wide Web">WWW</abbr>is amazing!</p>
</body>
</html>
```

चित्र 9.4 — HTML के मूल फॉर्मेटिंग टैग्स

आउटपुट

Text Formatting

This is a **bold** and *italic* text. Also underline and ~~strikethrough~~ text.

This is superscript and subscript text.

Font Style and Size

Styled text using <font>

Text Align

This text is centered using inline styling
This text is centered using center (deprecated)

Line breaks and whitespaces

This is some text.
Here is a new line.

```
This text
has
preserved
line breaks.
```

Quotations

> This is a blockquote.

He said, “Quote me!”

Abbrevitations and Acronyms

WWWis amazing!

चित्र 9.5 — HTML के मूल फॉर्मेटिंग टैग्स का आउटपुट

अभ्यास 9.3

- HTML के मूल टैगों की सूची बनाएँ।
- HTML टेक्स्ट फॉर्मेटिंग टैगों की सूची बनाएँ।
- HTML टेक्स्ट एलाइनमेंट (Text Alignment) टैगों की सूची बनाएँ।

9.8 HTML में रंग कोडिंग

HTML में आप रंग को विभिन्न तरीकों से निर्दिष्ट कर सकते हैं, जिनमें रंगों के नाम, हेक्साडेसीमल रंग कोड, RGB मान और HSL मान शामिल हैं। नीचे इन विधियों का एक संक्षिप्त विवरण प्रस्तुत है—

रंगों के नाम—

HTML में पहले से परिभाषित रंगों के नामों का एक संग्रह उपलब्ध है, जो आपको HTML कोड में सीधे रंग निर्दिष्ट करने की सुविधा देता है। उदाहरण के लिए—

```
<p style="color: red;">This text is reघ.</p>
<p style="color: blue;">This text is blue.</p>
```

आम तौर पर उपयोग होने वाले रंगों के नामों में red, blue, green, black, white, yellow, purple, orange आदि शामिल हैं।

9.8.1 हेक्साडेसीमल रंग कोड

HTML में रंग निर्दिष्ट करने की एक सामान्य विधि हेक्साडेसीमल रंग कोड है, जो एक # चिह्न के बाद छह हेक्साडेसीमल अंकों से मिलकर बना होता है। ये अंक रंग के लाल (Red), हरे (Green) और नीले (Blue) घटकों को दर्शाते हैं।

उदाहरण—

```
<p style="color: #FF0000;">This text is reघ.</p>
<p style="color: #0000FF;">This text is blue.</p>
```

हर दो हेक्साडेसीमल अंक क्रमशः लाल, हरे और नीले चैनलों की तीव्रता को दर्शाते हैं। उदाहरण स्वरूप, #FF0000 शुद्ध लाल रंग को, और #0000FF शुद्ध नीले रंग को दर्शाता है।

9.8.2 RGB रंग मान

आप RGB मानों का उपयोग करके रंग निर्दिष्ट कर सकते हैं, जिसमें लाल, हरे और नीले चैनलों की तीव्रता को अलग-अलग परिभाषित किया जाता है। RGB मानों को rgb() फंक्शन द्वारा निर्दिष्ट किया जाता है।

उदाहरण—

```
<p style="color: rgb(255, 0, 0);">This text is reघ.</p>
<p style="color: rgb(0, 0, 255);">This text is blue.</p>
```

rgb() फंक्शन के अंदर प्रत्येक मान 0 से 255 के बीच होता है और संबंधित रंग चैनल की तीव्रता को दर्शाता है।

9.8.3 HSL रंग मान

HSL (Hue, Saturation, Lightness) HTML में रंग निर्दिष्ट करने की एक अन्य विधि है, जो RGB की तुलना में रंग गुणों पर अधिक नियंत्रण प्रदान करती है। HSL मानों को hsl() फंक्शन द्वारा परिभाषित किया जाता है।

उदाहरण—

```
<p style="color: hsl(0, 100%, 50%);">This text is reघ.</p>
<p style="color: hsl(240, 100%, 50%);">This text is blue.</p>
```

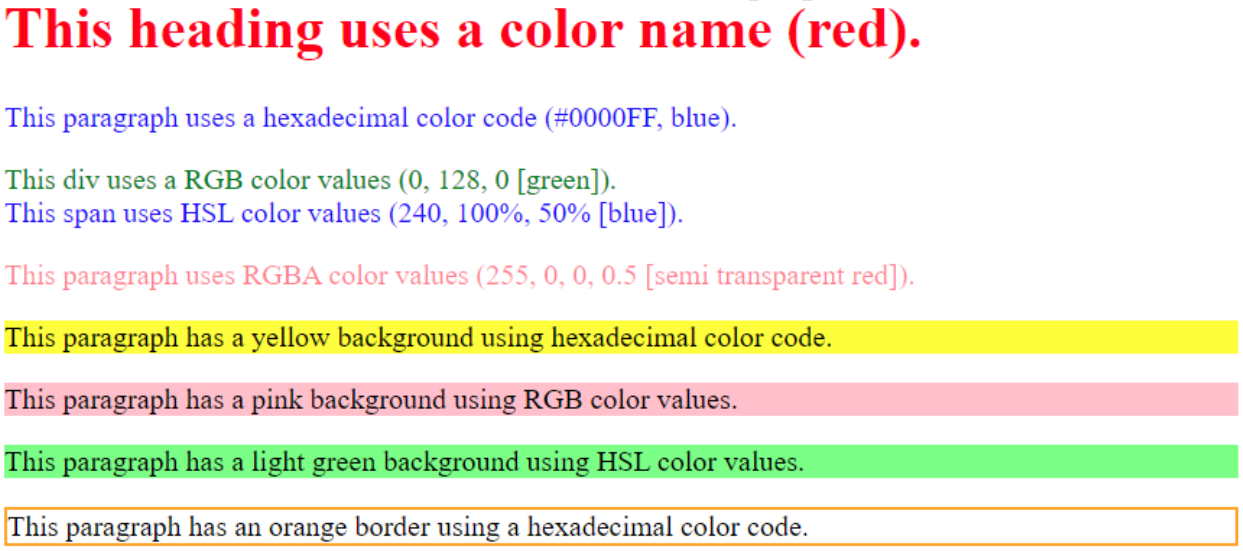
पहला मान (0 से 360 तक) रंग की आभा (Hue) को दर्शाता है, दूसरा मान (0% से 100% तक) संतृप्ति (Saturation) को दर्शाता है, और तीसरा मान (0% से 100% तक) प्रकाशता (Lightness) को दर्शाता है।

ये HTML में रंग निर्दिष्ट करने की प्रमुख विधियाँ हैं। आप इन रंग कोडों और मानों का उपयोग color, background-color या CSS स्टाइल्स में कर सकते हैं ताकि अपने वेब पृष्ठों पर टेक्स्ट, पृष्ठभूमि, बॉर्डर या अन्य तत्वों के रंग निर्धारित कर सकें जैसा कि **चित्र 9.6** में दिखाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Color Example</title>
</head>
<body>
    <h1 style="color: red;">This heading uses a color name (red).</h1>
    <p style="color: #0000ff">This paragraph uses a hexadecimal color code (#0000FF, blue).</p>
    <div style="color: rgb(0, 128, 0);">This div uses a RGB color values (0, 128, 0 [green]).</div>
    <span style="color: hsl(240, 100%, 50%);">This span uses HSL color values (240, 100%, 50% [blue]).</span>
    <p style="color: rgba(255, 0, 0, 0.5);">This paragraph uses RGBA color values (255, 0, 0, 0.5 [semi transparent red]).</p>
    <p style="background-color: #ffff00;">This paragraph has a yellow background using hexadecimal color code.</p>
    <p style="background-color: rgb(255, 192, 203);">This paragraph has a pink background using RGB color values.</p>
    <p style="background-color: hsl(120, 100%, 75%);">This paragraph has a light green background using HSL color values.</p>
    <p style="border: 2px solid #ffa500;">This paragraph has an orange border using a hexadecimal color code.</p>
</body>
</html>
```

चित्र 9.6— HTML में HSL रंग मान

आउटपुट



चित्र 9.7— HTML में HSL रंग मान के लिए आउटपुट

9.8.4 Div और Span टैग समूह बनाने के लिए

 और टैग आम तौर पर HTML में सामग्री को समूहित करने और उसकी संरचना हेतु प्रयोग किए जाते हैं, परंतु इनका उपयोग भिन्न संदर्भों में भिन्न उद्देश्यों के लिए किया जाता है, जैसा कि **चित्र 9.8** में दिखाया गया है।

** (डिवीजन)—**

 एक ब्लॉक-स्तरीय कंटेनर है जिसका उपयोग वेबपेज पर बड़ी सामग्री के समूहन और उसकी संरचना के लिए किया जाता है। इसका उपयोग अक्सर किसी सामग्री अनुभाग को एक मॉड्यूल के रूप में स्टाइल और फॉर्मेट करने हेतु किया जाता है।

 टैग का उपयोग पृष्ठ की लेआउट संरचना परिभाषित करने हेतु किया जाता है, जैसे कि हैडर, फुटर, साइडबार और सामग्री क्षेत्र। इन टैगों के साथ CSS का प्रयोग करके पृष्ठभूमि, बॉर्डर, मार्जिन और पैडिंग जैसी शैलियाँ एप्लाइ की जा सकती हैं।

 टैग एक ब्लॉक-स्तरीय बॉक्स बनाता है जो सामान्यतः एक नई पंक्ति से शुरू होता है और अपने पेरेंट कंटेनर की पूरी चौड़ाई लेता है।

```
<div id="header">
    <h1>Welcome to My Website</h1>
</div>
<div id="sidebar">
    <!-- Sidebar content goes here -->
</div>
<div id="content">
    <!-- Main content goes here -->
</div>
<div id="footer">
    &copy; 2024 My Website
</div>
```

चित्र 9.8— Div और Span टैग

9.8.5 (इनलाइन स्पैन)

 टैग एक इनलाइन-स्तरीय कंटेनर है जिसका उपयोग ब्लॉक-स्तरीय तत्वों के अंदर छोटे-छोटे टेक्स्ट अंशों को समूहित करने और उन पर शैलियाँ लागू करने हेतु किया जाता है।

इसका उपयोग सामान्यतः किसी टेक्स्ट या इनलाइन सामग्री के किसी विशिष्ट भाग को CSS शैलियाँ, फॉर्मेटिंग या JavaScript क्रियाओं के लिए लक्षित करने हेतु किया जाता है।

 टैग स्वयं पंक्ति विच्छेद (line break) उत्पन्न नहीं करता; यह केवल अपने पेरेंट तत्व (एलिमेंट) के अंदर की सामग्री को प्रभावित करता है।

यह तब उपयोगी होता है जब आप किसी बड़े टेक्स्ट खंड के अंदर किसी विशेष शब्द या वाक्यांश को शैलीबद्ध या नियंत्रित करना चाहते हैं।

उदाहरण—

This is a blue word in a sentence.
Click here to trigger an alert.

संक्षेप में, का उपयोग बड़ी सामग्री को संरचित करने और ब्लॉक-स्तरीय कंटेनर बनाने हेतु किया जाता है, जबकि का उपयोग इनलाइन-स्तरीय समूह बनाने तथा किसी ब्लॉक-स्तरीय तत्व (एलिमेंट) के अंदर छोटे भागों को शैलीबद्ध करने हेतु किया जाता है। वेब पृष्ठ की सामग्री को प्रभावी रूप से व्यवस्थित करने और शैलीबद्ध करने के लिए दोनों टैग आवश्यक हैं, जैसा कि **चित्र 9.9** में दिखाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Div and Span Example</title>
</head>
<body>
    <div style="background-color: #f0e60c;">Header</div>

    <div>
        <p>This is a <span style="background-color: yellow; font-weight: bold;">highlighted</span> word in a paragraph.</p>
        <p>Click <span style="background-color: yellow; font-weight: bold; cursor: pointer;"
            onclick="alert('hello')"
            >here</span>to triger a alert.</p>
    </div>

    <div style="background-color: #f0e60c;">&copy; 2024 My Website</div>
</body>
</html>
```

चित्र 9.9— HTML में (इनलाइन स्पैन)

आउटपुट

Header

This is a **highlighted** word in a paragraph.

Click **here**to triger a alert.

© 2024 My Website

चित्र 9.10— HTML में (इनलाइन स्पैन) के लिए आउटपुट

सारांश

- HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज) वेब पृष्ठों की संरचना के लिए आवश्यक है और प्रायः CSS व JavaScript के साथ प्रयोग की जाती है।
- HTML तत्व कोणीय ब्रैकेट्स (< >) के अंदर रखे जाते हैं और इनका प्रारंभिक तथा समापन टैग होता है।
- HTML में सामग्री की संरचना के लिए कई तत्व (एलिमेंट) होते हैं, जैसे हेडिंग्स, लिंक्स, सूचियाँ, चित्र, फॉर्म, तालिकाएँ आदि।
  एट्रिब्यूट्स का उपयोग HTML तत्वों के व्यवहार को परिभाषित या विस्तारित करने हेतु किया जाता है।

- HTML दस्तावेज़ एक विशेष संरचना का पालन करते हैं, जिनमें <!DOCTYPE html>, <html>, <head>, और <body> जैसे तत्व सम्मिलित होते हैं।
- HTML टैग किसी तत्व की शुरुआत और अंत को दर्शाते हैं, जबकि HTML तत्व में टैग तथा उसकी सामग्री दोनों सम्मिलित होते हैं।
- <b>, <i>, <u> जैसे फॉर्मेटिंग टैग्स का उपयोग टेक्स्ट को शैलीबद्ध करने हेतु किया जाता है; <font> टैग से फ़ॉन्ट की विशेषताएँ परिभाषित की जा सकती हैं।
- HTML में रंग निर्दिष्ट करने हेतु रंग के नाम, हेक्साडेसीमल कोड, RGB मान और HSL मानों का उपयोग किया जा सकता है।
- टैग का प्रयोग बड़ी सामग्री को समूहित और संरचित करने के लिए किया जाता है, जबकि टैग का प्रयोग इनलाइन-स्तरीय स्टाइलिंग हेतु किया जाता है।
- HTML तत्वों और गुणों की उचित समझ वेब विकास और सामग्री प्रस्तुति हेतु अत्यंत महत्वपूर्ण है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. HTML का पूर्ण रूप क्या है?
   (क) हाइपरटेक्स्ट मार्कअप लैंग्वेज
   (ख) हाइपर ट्रांसफर मार्कअप लैंग्वेज
   (ग) हाई-लेवल टेक्स्ट मार्कअप लैंग्वेज
   (घ) हाइपरटेक्स्ट टेक्स्ट मार्कअप लैंग्वेज

2. किस संस्था ने HTML को एक मान्यता प्राप्त मानक के रूप में स्थापित किया?
   (क) डब्ल्यू3सी (वर्ल्ड वाइड वेब कंसॉर्शियम)
   (ख) आईईईई (इंस्टीट्यूट ऑफ इलेक्ट्रिकल एंड इलेक्ट्रॉनिक्स इंजीनियर्स)
   (ग) आईएसओ (इंटरनेशनल ऑर्गनाइज़ेशन फॉर स्टैंडर्डाइज़ेशन)
   (घ) आईईटीएफ (इंटरनेट इंजीनियरिंग टास्क फोर्स)

3. HTML विनिर्देश का नवीनतम संस्करण कौन-सा है?
   (क) HTML4
   (ख) HTML5
   (ग) XHTML
   (घ) HTMLX

4. कौन-सा HTML तत्व मूल (root) तत्व के रूप में कार्य करता है और सभी अन्य तत्वों को समाहित करता है?
   (क) <body>
   (ख) <head>
   (ग) <html>
   (घ) <root>

5. किसी अनुच्छेद को परिभाषित करने के लिए कौन-सा HTML टैग प्रयोग किया जाता है?
   (क) <para>
   (ख) <text>
   (ग) <p>
   (घ) <line>

6. किसी अन्य वेब पृष्ठ या संसाधन से हाइपरलिंक जोड़ने के लिए कौन-सा HTML टैग उपयोग किया जाता है?
   (क) <link>
   (ख) <a>
   (ग) <hyper>
   (घ) <href>

7. कौन-सा HTML तत्व जोर दिए गए (emphasized) टेक्स्ट को प्रदर्शित करता है, जो सामान्यतः तिरछे अक्षरों (italics) में दिखाया जाता है?
   (क) <italic>
   (ख)
   (ग)
   (घ) <emphasis>
8. HTML में <blockquote> तत्व का उद्देश्य क्या है?
   (क) नेविगेशन लिंक की एक धारा बनाना
   (ख) टेक्स्ट को मोटा (bold) करना
   (ग) उद्धृत टेक्स्ट (quoted text) के खंड को परिभाषित करना
   (घ) संक्षेप या लघुरूप (abbreviation) को दर्शाना

9. किसी टेक्स्ट के अंदर पंक्ति-विराम जोड़ने के लिए कौन-सा HTML टैग उपयोग किया जाता है?
   (क) <line>
   (ख) <lb>
   (ग)
   (घ) <newline>

10. HTML में तत्व का उद्देश्य क्या है?
    (क) सामग्री के बड़े वर्गों को समूहबद्ध और संरचित करना
    (ख) टेक्स्ट के इनलाइन खंडों को परिभाषित करना
    (ग) हाइपरलिंक बनाना
    (घ) किसी तालिका की कक्ष (cell) को प्रदर्शित करना

ख. रिक्त स्थान भरें

1. HTML का पूर्ण रूप _____ Language है।
2. HTML विनिर्देश का नवीनतम संस्करण _____ है।
3. HTML दस्तावेज़ उन तत्वों (एलिमेंट्स) से मिलकर बने होते हैं जो _____ कोष्ठकों में लिखे जाते हैं।

4. किसी अनुच्छेद को परिभाषित करने के लिए प्रयुक्त HTML टैग _____ है।
5. किसी HTML तत्व के आरंभ और समापन टैग मिलकर _____ बनाते हैं।
6. अन्य वेब पृष्ठों या संसाधनों से _____ जोड़ने के लिए HTML टैग <a> प्रयोग किया जाता है।
7. _____ तत्व कोट किए गए टेक्स्ट के खंड को परिभाषित करने के लिए प्रयोग किया जाता है।
8. _____ तत्व सामान्यतः बड़ी सामग्री के खंडों को समूहबद्ध और संरचित करने हेतु प्रयोग किया जाता है।
9. _____ तत्व छोटे टेक्स्ट खंडों को समूहबद्ध और शैली प्रदान करने के लिए प्रयोग किया जाता है।
10. _____ के माध्यम से "red" या "blue" जैसे रंग नामों का उपयोग कर रंग निर्दिष्ट किए जा सकते हैं।

ग. सही या गलत

1. HTML एक टेक्स्ट-आधारित विधि है जिसका उपयोग किसी दस्तावेज़ की सामग्री के संगठन को परिभाषित करने के लिए किया जाता है।
2. HTML तत्व सामान्यतः बड़ी सामग्री को समूहबद्ध करने और उन पर शैलियाँ लागू करने के लिए प्रयोग किए जाते हैं।
3. <meta> टैग HTML दस्तावेज़ के मेटाडेटा को निर्दिष्ट करता है, जैसे कि कैरेक्टर एनकोडिंग और व्यूपोर्ट सेटिंग्स।
4. तत्व एक ब्लॉक-स्तरीय कंटेनर होता है, जिसका उपयोग सामग्री की संरचना के लिए किया जाता है।
5. हेक्साडेसिमल रंग कोड छह अंकों से बने होते हैं जो रंग के रेड, ग्रीन और ब्लू (RGB) घटकों का प्रतिनिधित्व करते हैं।
6. <blockquote> तत्व विभिन्न स्तरों की शीर्षकों को परिभाषित करने के लिए प्रयोग किया जाता है।
7. <head> अनुभाग वेब पृष्ठ की दृश्य सामग्री जैसे टेक्स्ट, चित्र और लिंक को समाहित करता है।
8. टेक्स्ट को स्टाइल करने के लिए <font> टैग का उपयोग करना चाहिए, CSS का नहीं।
9. <sub> टैग उपस्क्रिप्ट (subscript) टेक्स्ट बनाने के लिए प्रयोग किया जाता है।
10. <abbr> तत्व का उपयोग टेक्स्ट के अंदर पंक्ति-विराम जोड़ने के लिए किया जाता है।

घ. लघु उत्तर प्रश्न

1. HTML का पूर्ण रूप क्या है और वेब विकास में इसका प्रमुख उद्देश्य क्या है?
2. वेब विकास में HTML किस प्रकार CSS और JavaScript के साथ कार्य करता है?
3. किसी HTML तत्व की संरचना का वर्णन कीजिए, जिसमें आरंभ और समापन टैग सम्मिलित हों।
4. किसी अनुच्छेद को परिभाषित करने वाले HTML तत्व का एक उदाहरण दीजिए।
5. HTML टैग और HTML तत्व में क्या अंतर होता है?
6. HTML विशेषताओं (attributes) का उद्देश्य क्या होता है और इन्हें किसी HTML तत्व में कैसे निर्दिष्ट किया जाता है?
7. HTML दस्तावेज़ को अर्थपूर्ण (semantic) तत्वों से उचित रूप से संरचित करना क्यों आवश्यक होता है?
8. किसी HTML दस्तावेज़ में <head> अनुभाग की भूमिका क्या होती है और इसमें किस प्रकार की जानकारी होती है?
9. HTML में और तत्वों में क्या अंतर है, तथा किसका उपयोग कब करना चाहिए?
10. HTML में रंग निर्दिष्ट करने की तीन विभिन्न विधियों के नाम बताइए और संक्षेप में समझाइए कि वे कैसे कार्य करती हैं।

सत्र 10

सूचियों की अवधारणा — अनुक्रमहीन सूची, क्रमित सूची, परिभाषा सूची

आइए मिलते हैं अमन से, एक जिज्ञासु बालक जो चीज़ों को व्यवस्थित करना बहुत पसंद करता था। उसने "सूचियों की अवधारणा" के बारे में एक मज़ेदार तरीके से सीखा। उसे पता चला कि सूचियों के तीन प्रकार होते हैं—

अनुक्रमहीन सूची — जैसे कि खिलौनों को बिना किसी क्रम के एक शेल्फ़ पर रखना, केवल चीज़ों का एक समूह।

क्रमित सूची — यह ऐसा है जैसे कि खिलौनों को बड़े से छोटे या पुराने से नए के क्रम में सजाना।

परिभाषा सूची — इस सूची में चीज़ों को जोड़ा जाता है, जैसे खिलौनों की गाड़ियों को उनके रंगों से मिलाना, और प्रत्येक के साथ एक विशेष व्याख्या होती है।

अमन ने जाना कि सूचियाँ विचारों को व्यवस्थित करने में बेहद सहायक होती हैं, जैसे कि खिलौनों को सुव्यवस्थित करना या उन्हें जोड़ना। सूचियों से चीज़ों को याद रखना और उन्हें क्रम में रखना आसान हो जाता है, जैसे कि संगठित रहने का एक गुप्त उपकरण — जैसा कि चित्र 10.1 में दिखाया गया है।

चित्र 10.1— अमन खिलौनों को व्यवस्थित करता हुआ

इस अध्याय में, आप अनुक्रमहीन सूचियाँ, क्रमित सूचियाँ, परिभाषा सूचियाँ, छवि एवं छवि मानचित्रण (Image Mapping) तथा HTML हाइपरलिंक के बारे में सीखेंगे।

HTML में, आप विभिन्न प्रकार की सूचियाँ बनाकर किसी वेब पृष्ठ पर सामग्री को संरचित एवं प्रदर्शित कर सकते हैं। इन सूचियों में प्रमुख रूप से अनुक्रमहीन सूचियाँ, क्रमित सूचियाँ तथा परिभाषा सूचियाँ होती हैं।

10.1 अनुक्रमहीन सूची (<ul>)

अनुक्रमहीन सूचियाँ उन वस्तुओं की सूची को दर्शाने के लिए उपयोग की जाती हैं जिनका कोई विशेष संख्यात्मक या क्रमबद्ध अनुक्रम नहीं होता। अनुक्रमहीन सूचियों में वस्तुओं से पहले सामान्यतः बुलेट बिंदु (या अन्य चिह्न) होते हैं, जो प्रत्येक सूची आइटम को इंगित करते हैं।

 टैग का उपयोग अनुक्रमहीन सूची को परिभाषित करने के लिए किया जाता है, और प्रत्येक सूची आइटम <li> (list item) टैग में संलग्न होता है।

इनका सामान्य उपयोग नेविगेशन मेनू, बिंदुवार सूची, और जहाँ आइटमों का क्रम महत्त्वपूर्ण नहीं होता, वहाँ किया जाता है।

उदाहरण—

```
<ul>
        <li>Item 1</li>
```

```
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

क्या आप जानते हैं?
अनुक्रमहीन सूचियाँ ऐसी होती हैं जिनका कोई निश्चित क्रम नहीं होता और प्रत्येक आइटम बुलेट बिंदु से चिह्नित होता है।

10.2 क्रमित सूची (<ol>)

क्रमित सूचियाँ उन स्थितियों में प्रयुक्त होती हैं जहाँ वस्तुओं को विशिष्ट संख्यात्मक या क्रमिक रूप से दर्शाना आवश्यक होता है। क्रमित सूची में प्रत्येक आइटम के आगे सामान्यतः संख्या या अक्षर होता है, जो उनकी स्थिति को दर्शाता है।
क्रमित सूची बनाने के लिए <ol> टैग का उपयोग किया जाता है, और प्रत्येक आइटम <li> टैग में रखा जाता है। क्रमित सूचियाँ उन परिस्थितियों में उपयोगी होती हैं जहाँ क्रम महत्वपूर्ण होता है — जैसे चरणबद्ध प्रक्रिया, रैंकिंग, आदि।

उदाहरण—

```
<ol>
    <li>First step</li>
    <li>Second step</li>
    <li>Third step</li>
</ol>
```

क्या आप जानते हैं?
क्रमित सूचियों में स्वाभाविक रूप से क्रम होता है और प्रत्येक आइटम क्रमांकित होता है।

10.3 परिभाषा सूची (<dl>, <dt>, <dd>)

परिभाषा सूचियों का उपयोग शब्दों और उनके अर्थों या विवरणों की जोड़ी को प्रदर्शित करने के लिए किया जाता है।
<dl> टैग परिभाषा सूची को परिभाषित करता है। प्रत्येक शब्द <dt> (definition term) टैग में रखा जाता है, और उसका विवरण <dd> (definition description) टैग में लिखा जाता है।
इन सूचियों का सामान्य उपयोग शब्दावली (glossary), शब्दकोश (dictionary), और उन सभी स्थानों पर होता है जहाँ शब्द और उनके स्पष्टीकरण प्रस्तुत करने की आवश्यकता होती है।

उदाहरण—

```
<dl>
    <dt>HTML</dt>
    <dd>Hypertext Markup Language</dd>
    <dt>CSS</dt>
    <dd>Cascading Style Sheets</dd>
    <dt>JavaScript</dt>
    <dd>A programming language for the web</dd>
</dl>
```

क्या आप जानते हैं?
HTML विवरण सूची या परिभाषा सूची (Description List) तत्वों को शब्दकोश जैसी परिभाषात्मक रूप में प्रदर्शित करती है। <dl>, <dt> और <dd> टैग का प्रयोग विवरण सूची को परिभाषित करने के लिए किया जाता है।

HTML में सूची के प्रकार, जैसा कि चित्र 10.2 में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>List Examples</title>
</head>
<body>
    <h1>Type of Lists</h1>

    <!-- Unordered Lists -->
    <h2>Unordered Lists</h2>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>

    <!-- Ordered Lists -->
    <h2>Ordered Lists</h2>
    <ol>
        <li>First Step</li>
        <li>Second Step</li>
        <li>Third Step</li>
    </ol>

    <!-- Definition Lists -->
    <h2>Definition Lists</h2>
    <dl>
        <dt>HTML</dt>
        <dd>Hypertext Markup Language</dd>
        <dt>CSS</dt>
        <dd>Cascading Style Sheets</dd>
        <dt>Jaascript</dt>
        <dd>A programming language for the web.k</dd>
    </dl>

</body>
</html>
```

चित्र 10.2— HTML में सूची के प्रकार

आउटपुट—

Type of Lists

Unordered Lists

- Item 1
- Item 2
- Item 3

Ordered Lists

1. First Step
2. Second Step
3. Third Step

Definition Lists

HTML
: Hypertext Markup Language

CSS
: Cascading Style Sheets

Jaascript
: A programming language for the web.

चित्र 10.3— HTML में सूची के प्रकार का आउटपुट

10.4 चित्र और चित्र मैपिंग

HTML में <img> तत्व का उपयोग किसी वेब पृष्ठ में चित्र प्रदर्शित करने के लिए किया जाता है। इसके अतिरिक्त, आप इमेज मैप्स (Image Maps) को लागू कर सकते हैं, जो किसी चित्र के अंदर इंटरैक्टिव क्षेत्र को चिन्हित करने की सुविधा प्रदान करते हैं। इन क्षेत्रों में से प्रत्येक को किसी विशिष्ट URL से जोड़ा जा सकता है या किसी विशेष क्रिया को सक्रिय कर सकता है, जैसा कि चित्र 10.4 में दर्शाया गया है।

टिप्पणी— चित्र को <img> टैग का उपयोग करके जोड़ा जाता है।

यहाँ <img> तत्व और चित्र मैपिंग की अवधारणा का संक्षिप्त विवरण प्रस्तुत है—

10.4.1 <img> तत्व (चित्र समावेशन)—

<img> तत्व का प्रयोग वेब पृष्ठ में सीधे चित्र जोड़ने के लिए किया जाता है। यह एक स्वतः पूर्ण टैग होता है, अतः इसका कोई समापन </img> टैग नहीं होता।

src विशेषता (attribute) का उपयोग चित्र फ़ाइल के स्रोत URL को निर्दिष्ट करने के लिए किया जाता है। यह URL पूर्ण (absolute) या सापेक्षिक (relative) दोनों हो सकता है।

alt विशेषता वैकल्पिक टेक्स्ट (Alternative Text) प्रदान करती है। जब चित्र लोड नहीं हो पाता है या पहुँच-योग्य (Accessibility) की दृष्टि से यह टेक्स्ट प्रदर्शित होता है।

width और height जैसी विशेषताएँ चित्र के आकार को नियंत्रित करने हेतु प्रयोग की जा सकती हैं। फिर भी, स्टाइल संबंधी उद्देश्यों के लिए CSS का उपयोग करना अधिक उपयुक्त माना जाता है।

उदाहरण—

<img src="example.jpg" alt="Example Image" width="300" height="200">

10.4.2 चित्र मैपिंग (Image Mapping)—

चित्र मैपिंग के माध्यम से आप किसी चित्र पर क्लिक करने योग्य क्षेत्रों को परिभाषित कर सकते हैं, जिनमें से प्रत्येक क्षेत्र किसी विशिष्ट क्रिया या URL से संबद्ध होता है। यह तकनीक इंटरैक्टिव आरेख, नेविगेशन मेनू या चित्र आधारित प्रपत्र (forms) बनाने में सहायक होती है।

जब आप एक चित्र मैप तैयार करते हैं, तो <map> टैग का उपयोग करके मैप को परिभाषित किया जाता है। इस <map> टैग के अंदर <area> तत्वों का उपयोग उन चित्र क्षेत्रों को चिन्हित करने के लिए किया जाता है, जो क्लिक करने योग्य होते हैं।

<map> टैग में name विशेषता होती है, जो संबद्ध <img> टैग की usemap विशेषता से मेल खानी चाहिए।

प्रत्येक <area> तत्व में उस क्लिक क्षेत्र के आकार के अनुसार विभिन्न विशेषताएँ हो सकती हैं—

- आयताकार क्षेत्रों के लिए— shape, coords और href
- वृत्ताकार क्षेत्रों के लिए— shape, coords और href
- बहुभुजाकार क्षेत्रों के लिए— shape, coords और href

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Maps</title>
</head>
<body>
    <h2>Image Maps</h2>

    <img src="https://cdn.pixabay.com/photo/2017/05/11/11/15/workplace-2303851_960_720.jpg"
    alt="Desk" usemap="#deskmap" width="500" height="380">

    <map name="deskmap">
        <area shape="rect" coords="175, 242, 420, 358"
        href="https://en.wikipedia.org/wiki/Computer_keyboard" alt="Keyboard" target=_blank>

        <area shape="rect" coords="444, 251, 481, 357"
        href="https://en.wikipedia.org/wiki/Computer_mouse" alt="mouse" target="_blank">

        <area shape="rect" coords="375, 14, 481, 357"
        href="https://en.wikipedia.org/wiki/Book" alt="Diary" target="_blank">
    </map>
</body>
</html>
```

चित्र 10.4— HTML में चित्र और चित्र मैपिंग

आउटपुट

Image Maps

चित्र 10.5— HTML में चित्र और चित्र मैपिंग का आउटपुट

कोड को अपने कोड एडिटर (code editor) में चलाएँ और कीबोर्ड, माउस, तथा डायरी पर क्लिक करके देखें। आप जिस वस्तु पर क्लिक करेंगे, वह आपको अलग-अलग वेब पृष्ठों पर ले जाएगा।

10.5 HTML हाइपरलिंक

HTML में आप हाइपरलिंक (या लिंक) बनाकर वेब पृष्ठों, संसाधनों या इंटरनेट पर विभिन्न स्थानों के बीच नेविगेशन कर सकते हैं। हाइपरलिंक <a> (एंकर) टैग का उपयोग करके बनाए जाते हैं।

HTML में हाइपरलिंक बनाने का तरीका निम्नलिखित है—

Link Text

HTML हाइपरलिंक के महत्त्वपूर्ण भागों का विश्लेषण—

<a>: <a> तत्व हाइपरलिंक को परिभाषित करने के लिए प्रयुक्त होता है।

href: href विशेषता लिंक के URL या गंतव्य को निर्दिष्ट करती है। यह पूर्ण (absolute) URL (जैसे— "https://www.example.com") या सापेक्षिक (relative) URL (जैसे— "page.html") हो सकता है।

Link Text— यह टेक्स्ट वेब पृष्ठ पर दृश्य रूप में प्रदर्शित होता है और एक क्लिक करने योग्य लिंक के रूप में कार्य करता है। यह टेक्स्ट <a> टैग की शुरुआत और समापन टैग के बीच में रखा जाता है और प्रयोक्ता इसी पर क्लिक कर लिंक गंतव्य तक पहुँचते हैं।

HTML में हाइपरलिंक का उपयोग करने के कुछ उदाहरण निम्नलिखित हैं, जैसा कि चित्र 10.6 में दर्शाया गया है—

- **बाहरी वेबसाइट से लिंक करना—**

```
<a href="https://www.example.com">Visit Example.com</a>
```

- **आंतरिक पृष्ठ से लिंक करना (सापेक्षिक URL)—**

```
<a href="about.html">Learn About Us</a>
```

- **ईमेल पते से लिंक करना (mailto)—**

```
<a href="mailto:info@example.com">Contact Us</a>
```

• **फ़ोन नंबर से लिंक करना (tel)—**

Call Us

• **फ़ाइल से लिंक करना (जैसे PDF)—**

Download PDF

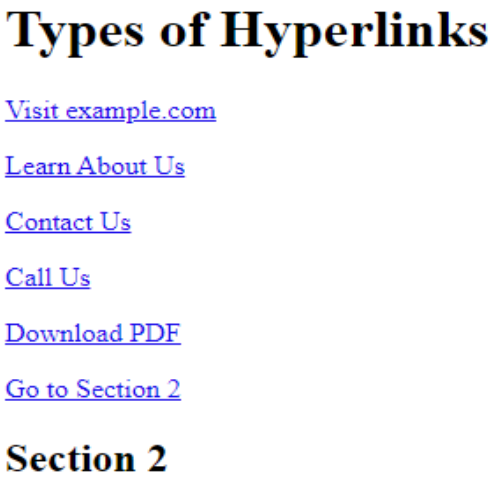
• **किसी पृष्ठ के विशिष्ट अनुभाग से लिंक करना (आंतरिक एंकर)—**

Go to Section 2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hyperlinks Examples</title>
</head>
<body>
    <h1>Types of Hyperlinks</h1>

    <!-- Link to an External Website -->
    <p><a href="https://www.example.com">Visit example.com</a></p>

    <!-- Link to an Internal Page (Relative URL) -->
    <p><a href="about.html">Learn About Us</a></p>

    <!-- Link to an Email Address (mailto) -->
    <p><a href="mailto:info@example.com">Contact Us</a></p>

    <!-- Link to a Phone Number (tel) -->
    <p><a href="tel: +919000000000">Call Us</a></p>

    <!-- Link to a File (e.g., PDF) -->
    <p><a href="document.pdf">Download PDF</a></p>

    <!-- Link to a specific section of a page (Internal Anchor) -->
    <p><a href="#section2">Go to Section 2</a></p>

    <!-- Anchor Point for Internal Link -->
    <h2 id="section2">Section 2</h2>
</body>
</html>
```

चित्र 10.6— HTML में हाइपरलिंक के प्रकार

आउटपुट



चित्र 10.6— HTML में हाइपरलिंक के प्रकार का आउटपुट

हमने विभिन्न <a> (एंकर) टैग्स का उपयोग करके कई प्रकार के हाइपरलिंक बनाए हैं—
• एक बाहरी वेबसाइट (https://www.example.com) से लिंक।
• उसी वेबसाइट के आंतरिक पृष्ठ (about.html) से लिंक।
• ईमेल भेजने के लिए एक मेलटू लिंक (mailto:info@example.com)।
• फ़ोन कॉल के लिए एक टेल लिंक (tel:+91 9000000000)।
• PDF फ़ाइल डाउनलोड करने हेतु लिंक (document.pdf)।
• उसी पृष्ठ पर एक विशेष अनुभाग (section2) तक जाने हेतु आंतरिक एंकर लिंक।
आंतरिक एंकर लिंक के लिए, हमने गंतव्य अनुभाग को परिभाषित करने हेतु उपयुक्त id विशेषता (id="section2") को भी परिभाषित किया है।

अभ्यास 10.1
• विभिन्न HTML हाइपरलिंक की सूची बनाएँ।
• इमेज मैपिंग में <area> एलिमेंट के विभिन्न गुणों (attributes) की सूची बनाएँ।

10.6 समान संसाधन स्थानकर्ता (URL) और URL एन्कोडिंग

समान संसाधन स्थानकर्ता (Uniform Resource Locator - URL) एक मानकीकृत पता होता है, जिसका उपयोग इंटरनेट पर संसाधनों की पहचान और उन्हें ढूँढ़ने के लिए किया जाता है। URL वेब नेविगेशन में आधारभूत भूमिका निभाता है, जिससे प्रयोक्ता वेबसाइट, वेब पृष्ठ, फ़ाइलें और अन्य विविध ऑनलाइन संसाधनों तक पहुँच सकते हैं। एक मानक URL में कई आवश्यक घटक होते हैं—

Scheme (योजना) — यह उस प्रोटोकॉल या विधि को निर्दिष्ट करता है जिसका उपयोग संसाधन तक पहुँचने के लिए किया जाता है। सामान्य योजनाओं में "http", "https", "ftp", "mailto" आदि शामिल हैं।

Domain (डोमेन) — यह उस सर्वर का इंटरनेट पता होता है जहाँ संसाधन होस्ट किया गया होता है। इसे IP पता (जैसे "192.168.1.1") या डोमेन नाम (जैसे "www.example.com") के रूप में दर्शाया जा सकता है।

Port (पोर्ट) — यह एक वैकल्पिक घटक होता है जो सर्वर पर कनेक्शन स्थापित करने के लिए पोर्ट नंबर को दर्शाता है। जब इसे छोड़ा जाता है, तो यह स्वतः चयनित योजना से जुड़े मानक पोर्ट (जैसे HTTP के लिए 80 और HTTPS के लिए 443) को मान लेता है।

Path (पथ) — यह सर्वर पर विशिष्ट संसाधन या स्थान की पहचान करता है। इसे अक्सर निर्देशिका या फ़ाइल पथ (जैसे "/folder/file.html") के रूप में दर्शाया जाता है।

Query (प्रश्नवाचक भाग) — यह एक वैकल्पिक भाग होता है जो संसाधन को अतिरिक्त पैरामीटर प्रदान करता है, और अक्सर वेब एप्लिकेशन में डेटा भेजने के लिए उपयोग किया जाता है (जैसे "?search=keyword")।

Fragment (खंड) — यह भी एक वैकल्पिक घटक होता है जो संसाधन के अंदर किसी विशेष खंड को निर्दिष्ट करता है (जैसे "#section2")।

एक पूर्ण URL इस प्रकार हो सकता है—

https://www.example.com:8080/path/to/resource?param1=value1&param2=value2#section3

10.7 URL एन्कोडिंग

URL एन्कोडिंग, जिसे प्रतिशत एन्कोडिंग (Percent Encoding) भी कहा जाता है, एक ऐसी विधि है जिसका उपयोग URL के अंदर आरक्षित या विशेष वर्णों (कैरेक्टर) को प्रदर्शित करने के लिए किया जाता है। URL में प्रयुक्त वर्णों (कैरेक्टर) का एक विशिष्ट सेट होता है, और सभी वर्णों को सीधे प्रयोग नहीं किया जा सकता। आरक्षित वर्णों जैसे स्पेस या विशेष चिह्नों को इस प्रकार एन्कोड किया जाता है कि वे सही तरीके से संप्रेषित हो सकें।

इस प्रक्रिया में प्रत्येक आरक्षित वर्ण को "%" चिन्ह के साथ दो हेक्साडेसिमल अंकों से प्रतिस्थापित किया जाता है, जो उस वर्ण के ASCII कोड का प्रतिनिधित्व करते हैं। उदाहरण के लिए—

• Space (स्पेस) ⟶ %20

• Ampersand (&) ⟶ %26

• Question mark (?) ⟶ %3F

URL एन्कोडिंग तब अत्यावश्यक होती है जब डेटा को क्वेरी पैरामीटर के माध्यम से URL में भेजा जा रहा हो, या जब फ़ाइल पथों में विशेष वर्ण सम्मिलित हों। अधिकांश प्रोग्रामिंग भाषाएँ और वेब फ्रेमवर्क URL एन्कोडिंग और डिकोडिंग के लिए अंतर्निर्मित कार्य (functions) प्रदान करते हैं, जिससे यह प्रक्रिया स्वचालित रूप से पूरी हो जाती है।

> **क्या आप जानते हैं?**
> URL एन्कोडिंग गैर-ASCII वर्णों (कैरेक्टर) को एक ऐसे स्वरूप में परिवर्तित करता है जिसे इंटरनेट पर स्थानांतरित किया जा सके।

10.8 वेब पृष्ठ में तालिका

वेब पृष्ठों में तालिकाओं का उपयोग डेटा को एक सुव्यवस्थित ग्रिड प्रारूप में संरचित और प्रदर्शित करने के लिए किया जाता है। वे पंक्तियों और स्तंभों से बनी होती हैं, जिनका प्रतिच्छेदन एक कोष्ठक (cell) बनाता है। तालिकाओं का उपयोग विभिन्न परिदृश्यों में किया जाता है, जैसे सारणीबद्ध डेटा प्रदर्शित करना, ग्रिड बनाना, सूचना व्यवस्थित करना और पृष्ठ लेआउट तैयार करना।

तालिका के घटक—

HTML में तालिका बनाने के लिए विशिष्ट HTML तत्वों के एक सेट का उपयोग करना होता है जो तालिका की संरचना और सामग्री को परिभाषित करते हैं। HTML तालिका के प्रमुख तत्व इस प्रकार हैं—

<table> — यह तत्व संपूर्ण तालिका को परिभाषित करता है। यह तालिका से संबंधित सभी तत्वों का कंटेनर होता है, जैसे पंक्तियाँ, कॉलम और कोष्ठक।

<tr> — "टेबल रो" तत्व तालिका में एक पंक्ति को परिभाषित करता है। प्रत्येक पंक्ति में एक या अधिक कोष्ठक होते हैं।

<th> — "टेबल हेडर सेल" तत्व किसी तालिका पंक्ति के अंदर शीर्षक कोष्ठक को परिभाषित करता है। ये सामान्यतः कॉलम शीर्षक होते हैं और डिफ़ॉल्ट रूप से मोटे (bold) अक्षरों में प्रदर्शित होते हैं।

<td> — "टेबल डेटा सेल" तत्व किसी तालिका पंक्ति के अंदर सामान्य डेटा कोष्ठक को परिभाषित करता है। इसमें तालिका की वास्तविक सामग्री होती है।

<caption> — यह तत्व तालिका के लिए शीर्षक या कैप्शन प्रदान करता है। यह सामान्यतः <table> के अंदर, लेकिन किसी पंक्ति के बाहर स्थित होता है।

उदाहरण 10.7— एक साधारण तालिका बनाना

यहाँ एक साधारण HTML तालिका का उदाहरण दिया गया है जिसमें दो पंक्तियाँ और तीन कॉलम हैं, जैसा कि चित्र 10.7 में दर्शाया गया है।

```
<table>
    <caption>Sample HTML Table</caption>
    <thead>
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
            <th>Header 3</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
            <td>Data 3</td>
        </tr>
    </tbody>
</table>
```

चित्र 10.7— दो पंक्तियों और तीन स्तंभों वाली साधारण तालिका

इस उदाहरण में—

• <table> संपूर्ण तालिका को परिभाषित करता है।
• <caption> तालिका का शीर्षक प्रदान करता है।
• <tr> दो पंक्तियाँ परिभाषित करता है।
• <thead> तालिका शीर्षक को परिभाषित करता है।
• <tbody> तालिका की मुख्य सामग्री को परिभाषित करता है।
• <th> कॉलम शीर्षकों के लिए शीर्षक कोष्ठक को परिभाषित करता है।
• <td> वास्तविक सामग्री वाले डेटा कोष्ठक को परिभाषित करता है।

यह संरचना तालिका के पहले पंक्ति में शीर्षक और दूसरे पंक्ति में डेटा प्रदर्शित करती है।3

Sample HTML Table

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3

चित्र 10.8— दो पंक्तियों और तीन स्तंभों वाली साधारण तालिका का आउटपुट

अभ्यास 10.2

• तालिका बनाने में प्रयुक्त HTML तत्वों की सूची बनाएँ।
• URL के आवश्यक घटकों की सूची बनाएँ।

HTML में जटिल तालिका का उदाहरण — जैसा कि चित्र 10.9 में दिखाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Complex Table Example</title>
</head>
<body>
    <h1>Complex Table Example</h1>
    <table style="width: 100%; border-collapse: collapse;">
        <caption>Exployee Information</caption>
        <thead>
            <tr>
                <th style="background-color: #f2f2f2; border: 1px solid #ddd; padding: 8px; text-align: left;">Employee ID</th>
                <th style="background-color: #f2f2f2; border: 1px solid #ddd; padding: 8px; text-align: left;">First Name</th>
                <th style="background-color: #f2f2f2; border: 1px solid #ddd; padding: 8px; text-align: left;">Last Name</th>
                <th style="background-color: #f2f2f2; border: 1px solid #ddd; padding: 8px; text-align: left;">Department</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">0001</td>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">Kunal</td>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">Singh</td>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">Engineering</td>
            </tr>
            <tr>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">0002</td>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">Rajesh</td>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">Kahar</td>
                <td style="border: 1px solid #ddd; padding: 8px; text-align: left;">Engineering</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="2" style="border: 1px solid #ddd; padding: 8px; text-align: left;">Total Employees:</td>
                <td colspan="2" style="border: 1px solid #ddd; padding: 8px; text-align: left;">2</td>
            </tr>
        </tfoot>

    </table>
</body>
</html>
```

चित्र 10.9— HTML में जटिल तालिका का उदाहरण

आउटपुट —

Complex Table Example

Exployee Information

Employee ID	First Name	Last Name	Department
0001	Kunal	Singh	Engineering
0002	Rajesh	Kahar	Engineering
Total Employees:		2	

चित्र 10.10— HTML में जटिल तालिका के उदाहरण का आउटपुट

10.9 <tfoot>, <tbody>, <colgroup> और <caption>

10.9.1 <tfoot> एलिमेंट

<tfoot> एलिमेंट का उपयोग तालिका के लिए एक फूटर अनुभाग को परिभाषित करने के लिए किया जाता है। यह सामान्यतः <tbody> अनुभाग के बाद और </table> टैग से पहले प्रकट होता है। <tfoot> के अंदर की सामग्री में अक्सर तालिका से संबंधित सारांश या फूटर में जानकारी होती है। यह तालिका के अंत में संदर्भ या सारांश डेटा प्रदान करने हेतु उपयोगी होता है। <tfoot> का प्रयोग कैसे किया जाता है, यह चित्र 10.11 में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Table</title>
</head>
<body>
    <table>
        <thead>
            <tr>
                <th>Header 1</th>
                <th>Header 2</th>
                <th>Header 3</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Data 1</td>
                <td>Data 2</td>
                <td>Data 3</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="3">Footer Text</td>
            </tr>
        </tfoot>
    </table>
</body>
</html>
```

चित्र 10.11— HTML में <tfoot> एलिमेंट

आउटपुट

Header 1 Header 2 Header 3

Data 1 Data 2 Data 3

Footer Text

चित्र 10.12— HTML में <tfoot> एलिमेंट का आउटपुट

10.9.2 <tbody> एलिमेंट

<tbody> एलिमेंट तालिका के केंद्रीय भाग को परिभाषित करने में महत्वपूर्ण भूमिका निभाता है। यह सामान्यतः तालिका की पंक्तियों और डेटा कोष्ठकों को सम्मिलित करता है। यद्यपि इसका उपयोग अनिवार्य नहीं है, परंतु तालिका की मुख्य सामग्री को व्यवस्थित करने के लिए इसका प्रयोग करना उत्तम अभ्यास माना जाता है।

10.9.3 <caption> एलिमेंट

<caption> एलिमेंट का उपयोग तालिका के लिए शीर्षक या कैप्शन प्रदान करने हेतु किया जाता है। यह तालिका के ऊपर या नीचे प्रकट हो सकता है, और तालिका की सामग्री के लिए संदर्भ या संक्षिप्त विवरण प्रदान कर सकता है। यद्यपि यह वैकल्पिक है, परंतु पहुँच (accessibility) और स्पष्टता बढ़ाने हेतु इसे प्रयोग करने की सिफारिश की जाती है। <caption> एलिमेंट का उपयोग कैसे किया जाता है, यह चित्र 10.13 में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Employee List</title>
</head>
<body>
    <table>
        <caption>Employee List</caption>
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Department</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>001</td>
                <td>Kunal Singh</td>
                <td>Engineering</td>
            </tr>
            <tr>
                <td>002</td>
                <td>Rajesh Kahar</td>
                <td>Engineering</td>
            </tr>
            <!-- More rows... -->
        </tbody>
    </table>
</body>
</html>
```

चित्र 10.13— HTML में <caption> एलिमेंट

आउटपुट

Employee List

ID	Name	Department
001	Kunal Singh	Engineering
002	Rajesh Kahar	Engineering

चित्र 10.14— HTML में <caption> एलिमेंट का आउटपुट

10.9.4 <colgroup> एलिमेंट

<colgroup> एलिमेंट का उपयोग तालिका में स्तंभों के समूह को परिभाषित करने और उन पर सामूहिक रूप से स्टाइल या अन्य गुण लागू करने हेतु किया जाता है। यह प्रायः <col> एलिमेंट के साथ मिलकर प्रयोग किया जाता है, ताकि किसी व्यक्तिगत कॉलम या स्तंभों के समूह पर विशेष गुण निर्दिष्ट किए जा सकें। <colgroup> एलिमेंट का प्रयोग कैसे किया जाता है, यह चित्र 10.15 में दर्शाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Understanding colgroup</title>
</head>
<body>
    <table>
        <colgroup>
            <col style="background-color: #f2f2f2;">
            <col style="font-weight: bold;">
        </colgroup>
        <thead>
            <tr>
                <th>Header 1</th>
                <th>Header 2</th>
                <th>Header 3</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Data 1</td>
                <td>Data 2</td>
                <td>Data 3</td>
            </tr>
            <!-- More rows... -->
        </tbody>
    </table>
</body>
</html>
```

चित्र 10.15— HTML में <colgroup> एलिमेंट

आउटपुट

Header 1	Header 2	Header 3
Data 1	Data 2	Data 3

चित्र 10.16— HTML में <colgroup> एलिमेंट का आउटपुट

इस उदाहरण में, <colgroup> एलिमेंट का उपयोग पहले दो स्तंभों को एक समूह में बाँधने हेतु किया गया है, और इन स्तंभों पर पृष्ठभूमि का रंग एवं फ़ॉन्ट का भार निर्धारित करने हेतु इनलाइन स्टाइल्स लागू किए गए हैं। <colgroup> के अंदर के <col> एलिमेंट, समूह के अंदर प्रत्येक कॉलम पर अतिरिक्त गुण लागू करने की सुविधा देते हैं।

अभ्यास 10.3

1. निम्नलिखित HTML तत्वों का कोड लिखिए—
   - <tbody>
   - <colgroup>
   - <caption>
   - <tfoot>

10.10 वेबपेज में क्लाइंट-साइड फ़ॉर्म बनाना

वेबपेज में क्लाइंट-साइड फ़ॉर्म डिज़ाइन करना HTML का उपयोग करके फ़ॉर्म की रूपरेखा तैयार करना और टेक्स्ट फ़ील्ड, रेडियो बटन, चेकबॉक्स तथा सबमिट बटन जैसे फ़ॉर्म घटकों को सम्मिलित करना शामिल होता है।

HTML <form> एलिमेंट को प्रस्तुत करता है, जो वेबपेज में इंटरैक्टिव फ़ॉर्म तत्वों को समाहित करने वाले एक संगठित खंड का निर्माण करने के लिए आधार का कार्य करता है। फ़ॉर्म वेब विकास में एक महत्वपूर्ण भूमिका निभाते हैं, क्योंकि ये प्रयोक्ता पंजीकरण, खोज सुविधा, डेटा सबमिशन और अन्य कई कार्यों के लिए प्रयुक्त होते हैं। <form> एलिमेंट सभी फ़ॉर्म नियंत्रणों (controls) को एकीकृत करने वाला कंटेनर होता है और प्रयोक्ताओं द्वारा प्रविष्ट किए गए डेटा के संचालन के लिए नियमों को भी परिभाषित करता है। यह वेब अनुप्रयोगों और वेबसाइटों के संदर्भ में प्रयोक्ता इनपुट को एकत्र करने और संसाधित करने की रूपरेखा प्रदान करता है।

नीचे <form> एलिमेंट के प्रमुख गुणों और उद्देश्यों का विवरण दिया गया है—

10.10.1 *action* गुण (Attribute)

action गुण उस URL या स्क्रिप्ट को निर्दिष्ट करता है जहाँ प्रयोक्ता द्वारा फ़ॉर्म सबमिट करने पर डेटा भेजा जाएगा।

उदाहरण— <form action="submit_form.php" method="post">

10.10.2 *method* गुण (Attribute)

method गुण यह निर्धारित करता है कि फ़ॉर्म डेटा सर्वर को भेजने के लिए कौन-सी HTTP विधि (method) प्रयुक्त होगी। सामान्यतः प्रयुक्त विधियाँ हैं — "get" और "post"।

GET फ़ॉर्म डेटा को URL में जोड़ता है और सरल क्वेरी के लिए उपयुक्त होता है।

POST फ़ॉर्म डेटा को HTTP अनुरोध बॉडी में भेजता है और जटिल या संवेदनशील डेटा के लिए प्रयुक्त होता है।

उदाहरण— <form action="submit_form.php" method="post">

10.10.3 फ़ॉर्म नियंत्रण (Form Controls)

फ़ॉर्म नियंत्रण वे इंटरैक्टिव तत्व होते हैं जैसे टेक्स्ट इनपुट, रेडियो बटन, चेकबॉक्स, चयन सूची (select dropdown) और बटन, जो प्रयोक्ता को डेटा दर्ज करने या विकल्प चुनने की सुविधा देते हैं। ये नियंत्रण <form> एलिमेंट के अंदर रखे जाते हैं।

10.10.4 *name* गुण

name गुण प्रत्येक फ़ॉर्म नियंत्रण को पहचानने के लिए प्रयोग किया जाता है, जब फ़ॉर्म सबमिट किया जाता है। यह नाम प्रयोक्ता द्वारा प्रविष्ट डेटा के साथ भेजा जाता है।

उदाहरण— <input type="text" name="username">

10.10.5 *id* गुण

id गुण प्रत्येक फ़ॉर्म नियंत्रण के लिए एक विशिष्ट पहचानकर्ता (unique identifier) प्रदान करता है। इसका उपयोग प्रायः लेबल के साथ किया जाता है ताकि वे संबंधित फ़ॉर्म नियंत्रणों से समन्वित हो सकें, जिससे पहुँच-योग्यता (accessibility) बढ़ती है।

उदाहरण— <label for="username">Username:</label><input type="text" id="username" name="username">

10.10.6 *for* गुण

<label> एलिमेंट में प्रयुक्त for गुण यह दर्शाता है कि यह किस फ़ॉर्म नियंत्रण से संबद्ध है। यह फ़ॉर्म नियंत्रण के id से मेल खाता है।

उदाहरण— <label for="username">Username:</label><input type="text" id="username" name="username">

10.10.7 *required* गुण

required गुण को फ़ॉर्म नियंत्रणों में जोड़ा जा सकता है ताकि यह दर्शाया जा सके कि प्रयोक्ता को फ़ॉर्म सबमिट करने से पूर्व आवश्यक रूप से इनपुट देना होगा। यह क्लाइंट-साइड पर इनपुट वैधता (validation) लागू करता है।

उदाहरण—<input type="text" name="name" required>

10.10.8 सबमिट बटन (<input type="submit">) या सबमिट क्रिया

एक सबमिट बटन या फ़ॉर्म क्रिया (जैसे— JavaScript फ़ंक्शन) का उपयोग सर्वर को फ़ॉर्म डेटा भेजने की प्रक्रिया को प्रारंभ करने के लिए किया जाता है।

उदाहरण— <input type="submit" value="Submit">

10.10.9 रीसेट बटन (<input type="reset">) (वैकल्पिक)

रीसेट बटन प्रयोक्ताओं को फ़ॉर्म के इनपुट फ़ील्ड्स को साफ़ करके प्रारंभिक स्थिति में पुनः स्थापित करने की सुविधा देता है।

उदाहरण—<input type="reset" value="Reset">

<form> एलिमेंट अपने संबंधित फ़ॉर्म नियंत्रणों और गुणों के साथ, वेबपृष्ठों पर डेटा एकत्र करने और सबमिट करने का एक संगठित और मानकीकृत तरीका प्रदान करता है। फ़ॉर्म डेटा को सर्वर-साइड स्क्रिप्ट्स (जैसे PHP, Python, Node.js) के माध्यम से संसाधित किया जा सकता है या क्लाइंट-साइड पर JavaScript द्वारा नियंत्रित किया जा सकता है।

10.10.10 टेक्स्ट इनपुट (<input type="text">)

<label for="name">Name:</label> — <label> एलिमेंट इनपुट फ़ील्ड के लिए लेबल प्रदान करता है। for गुण संबंधित इनपुट फ़ील्ड के id से मेल खाता है ताकि पहुँच-योग्यता सुनिश्चित हो सके।

<input type="text" id="name" name="name" required> — यह एक टेक्स्ट इनपुट फ़ील्ड बनाता है जिसमें id="name" है, name फ़ॉर्म सबमिशन के दौरान फ़ील्ड को पहचानने हेतु है और required यह दर्शाता है कि यह फ़ील्ड आवश्यक है।

10.10.11 ईमेल इनपुट (<input type="email">)

यह टेक्स्ट इनपुट के समान होता है, परंतु इसका type="email" होता है ताकि केवल वैध ईमेल इनपुट ही स्वीकार किया जाए।

10.10.12 रेडियो बटन (<input type="radio">)

रेडियो बटन का उपयोग परस्पर बहिष्कृत विकल्पों (mutually exclusive options) के लिए किया जाता है (जैसे— लिंग)। प्रत्येक रेडियो बटन का एक अद्वितीय id, एक name गुण (सभी को एक समूह में रखने हेतु), और एक value गुण होता है जो विकल्प से संबंधित मान को निर्दिष्ट करता है।

10.10.13 चेकबॉक्स (<input type="checkbox">)

चेकबॉक्स का उपयोग बहुविकल्प चयन के लिए किया जाता है (जैसे— रुचियाँ)। प्रत्येक चेकबॉक्स का एक अद्वितीय id, एक name गुण (आरेख सरणी के रूप में), और एक value गुण होता है जो विकल्प से संबंधित मान को निर्दिष्ट करता है।

10.10.14 टेक्स्टएरिया (<textarea>)

<label for="message">Message:</label> — टेक्स्टएरिया के लिए लेबल।

<textarea id="message" name="message" rows="4" required></textarea> — यह एक बहु-पंक्ति इनपुट फ़ील्ड बनाता

है जिसमें id="message" है, name फ़ॉर्म में फ़ील्ड की पहचान हेतु है, rows="4" दृश्य पंक्तियों की संख्या को दर्शाता है, और required दर्शाता है कि इनपुट आवश्यक है।

10.10.15 सबमिट बटन (<input type="submit">)

यह इनपुट एलिमेंट एक बटन बनाता है, जो क्लिक किए जाने पर फ़ॉर्म को सबमिट करता है।

10.10.16 <select>

HTML में <select> एलिमेंट का उपयोग एक ड्रॉपडाउन सूची (dropdown list) या चयन बॉक्स (select box) बनाने हेतु किया जाता है, जिससे प्रयोक्ता पूर्वनिर्धारित विकल्पों की सूची में से एक या अधिक विकल्पों का चयन कर सकते हैं। <select> एलिमेंट को अक्सर <option> एलिमेंट्स के साथ जोड़ा जाता है, जो ड्रॉपडाउन सूची के अंदर उपलब्ध विकल्पों को परिभाषित करते हैं जैसा कि **चित्र 10.17** में दर्शाया गया है।

```
<select>
    <option value="option1">Option 1</option>
    <option value="option2">Option 2</option>
    <option value="option3">Option 3</option>
    <!-- More options... -->
</select>
```

चित्र 10.17— <select> का कोड

आइए <select> एलिमेंट और इससे संबंधित तत्वों के प्रमुख घटकों को समझें— *(यहाँ अगला भाग — <select> के अंदर <option> एलिमेंट आदि का विवरण — यदि आप चाहें तो अगली किस्त में प्रस्तुत किया जा सकता है।)*

10.10.17 <select> एलिमेंट

<select> एलिमेंट ड्रॉपडाउन सूची कंटेनर बनाता है। इसमें name और id जैसे गुण (attributes) हो सकते हैं, जो HTML या JavaScript में इस चयन बॉक्स की पहचान और संदर्भ के लिए प्रयुक्त होते हैं।

उदाहरण— <select name="gender" id="gender">

10.10.18 <option> एलिमेंट्स

<select> एलिमेंट के अंदर, आप एक या एक से अधिक <option> एलिमेंट सम्मिलित कर सकते हैं, जो ड्रॉपडाउन सूची में अलग-अलग विकल्पों को परिभाषित करते हैं। प्रत्येक <option> एलिमेंट एक एकल विकल्प का प्रतिनिधित्व करता है। प्रत्येक <option> एलिमेंट का value गुण उस मान को निर्दिष्ट करता है जो फ़ॉर्म सबमिट किए जाने पर सर्वर को भेजा जाएगा। <option> टैग के प्रारंभ और अंत टैग के बीच का टेक्स्ट वही होता है जो प्रयोक्ता को विकल्प के रूप में दिखता है।

10.10.19 बहुविकल्प चयन (वैकल्पिक)

आप <select> एलिमेंट में multiple गुण जोड़कर प्रयोक्ताओं को एक से अधिक विकल्पों को चुनने की अनुमति दे सकते हैं। यह एक मल्टी-सिलेक्ट ड्रॉपडाउन बनाता है।

उदाहरण— <select name="interests" multiple>

10.10.20 निष्क्रिय विकल्प (वैकल्पिक)

आप <option> एलिमेंट में disabled गुण जोड़कर विशिष्ट विकल्पों को निष्क्रिय कर सकते हैं। निष्क्रिय विकल्प प्रयोक्ता द्वारा चयनित नहीं किए जा सकते।

उदाहरण— <option value="disabled_option" disabled>Disabled Option</option>

10.10.21 पूर्वनिर्धारित विकल्प

आप <option> एलिमेंट में selected गुण जोड़कर यह निर्दिष्ट कर सकते हैं कि पृष्ठ लोड होते समय कौन-सा विकल्प पहले से चयनित हो।

उदाहरण—<option value="preselected_option"selected>Preselected Option</option>

कुल मिलाकर, ये फ़ॉर्म एलिमेंट प्रयोक्ताओं को डेटा इनपुट और सबमिट करने की सुविधा प्रदान करते हैं, और इनसे जुड़े गुण (attributes) फ़ॉर्म फ़ील्ड के लिए निर्देश और प्रमाणीकरण नियम निर्धारित करते हैं। जब फ़ॉर्म सबमिट किया जाता है, तो डेटा <form> एलिमेंट के action और method गुणों के अनुसार सर्वर पर प्रोसेसिंग के लिए भेजा जाता है, जैसा कि **चित्र 10.18** में दर्शाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Client Side Form Example</title>
</head>
<body>
    <h1>Contact Us</h1>

    <!-- Form Element -->
    <form action="submit_form.php" method="POST">
        <!-- Text Input -->
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required>
        <br>
        <!-- Email Input -->
        <label for="email">Email Address</label>
        <input type="email" id="email" name="email" required>
        <br>
        <!-- Select -->
        <label for="color">Select your favourite color:</label>
        <select name="color" id="color">
            <option value="red">Red</option>
            <option value="green">Green</option>
            <option value="blue">Blue</option>
            <option value="yellow">Yellow</option>
        </select>
        <br>
        <!-- Radio Buttons -->
        <label>Gender:</label>
        <input type="radio" id="male" name="gender" value="male">
        <label for="male">Male</label>
        <input type="radio" id="female" name="gender" value="female">
        <label for="female">Female</label>
        <br>
        <!-- Textarea -->
        <label for="message">Message</label>
        <textarea name="message" id="message" cols="30" rows="10" required></textarea>
        <br>
        <!-- Submit Button -->
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

चित्र 10.18— HTML में Contact Us

आउटपुट

चित्र 10.19— HTML में Contact Us का आउटपुट

अभ्यास 10.4

- चेकबॉक्स
- सबमिट बटन
- टेक्स्ट इनपुट
- नाम

10.11 दस्तावेज़ के हेडर और मेटाडेटा का उपयोग करें

10.11.1 दस्तावेज़ का मेटाडेटा

मेटाडेटा का मुख्य उद्देश्य दस्तावेज़ से संबंधित जानकारी प्रदान करना होता है, जिसमें दस्तावेज़ का शीर्षक, कैरेक्टर एनकोडिंग, लेखनकर्ता और अन्य आवश्यक गुण सम्मिलित होते हैं। मेटाडेटा एलिमेंटों का यह समूह वेब विकास का अभिन्न अंग होता है और वेब दस्तावेज़ों की प्रभावी व्याख्या और संप्रेषण में अत्यंत आवश्यक होता है। सामान्यतः उपयोग किए जाने वाले मेटाडेटा एलिमेंटों में निम्नलिखित सम्मिलित हैं—

Title (<title>)— <title> एलिमेंट HTML दस्तावेज़ के शीर्षक को परिभाषित करता है, जो ब्राउज़र के शीर्षक पट्टी या टैब में प्रमुख रूप से प्रदर्शित होता है। प्रयोक्ता अनुभव को बेहतर बनाने के अलावा, यह SEO (Search Engine Optimization) में भी महत्वपूर्ण भूमिका निभाता है, जिससे यह वेब विकास का एक अनिवार्य भाग बन जाता है।

Base (<base>)— <base> एलिमेंट दस्तावेज़ में प्रयुक्त सापेक्ष (relative) URL के लिए एक आधार URL निर्दिष्ट करता है। यह लिंक किए गए संसाधनों के लिए सापेक्ष URLs का समाधान करने में सहायता करता है।

Link (<link>)— <link> एलिमेंट का प्रयोग मुख्य रूप से बाह्य संसाधनों जैसे स्टाइलशीट्स (CSS) और आइकन फ़ाइलों (favicon) से लिंक करने के लिए किया जाता है। यह अन्य दस्तावेजों से लिंक स्थापित करने और संबंधों को परिभाषित करने के लिए भी प्रयोग किया जा सकता है।

Style (<style>)— <style> एलिमेंट का प्रयोग HTML दस्तावेज़ के अंदर आंतरिक CSS शैलियाँ (styles) सम्मिलित करने के लिए किया जाता है। यह सामग्री को सजाने हेतु CSS नियमों को परिभाषित करता है।

Script (<script>)— <script> एलिमेंट का उपयोग HTML दस्तावेज़ में JavaScript कोड सम्मिलित करने के लिए किया जाता है। इसे दस्तावेज़ के <head> या <body> खंड में रखा जा सकता है।

10.11.2 HTML मेटा टैग

<meta> टैग HTML दस्तावेज़ से संबंधित मेटाडेटा प्रदान करने के लिए एक बहुउद्देशीय उपकरण है। यह प्रायः दस्तावेज़ के <head> खंड में स्थित होता है और आवश्यक जानकारी प्रस्तुत करता है। <meta> टैग के प्रमुख गुण इस प्रकार हैं—

charset— दस्तावेज़ की कैरेक्टर एनकोडिंग निर्दिष्ट करता है (उदा. <meta charset="UTF-8">)।

name और content— विभिन्न प्रकार के मेटाडेटा को परिभाषित करने के लिए प्रयुक्त होते हैं, जैसे कि लेखक, विवरण, कीवर्ड्स, और viewport सेटिंग्स (उदा. <meta name="author" content="John Doe">)।

10.11.3 XHTML (एक्स्टेंसिबल हाइपरटेक्स्ट मार्कअप लैंग्वेज)

XHTML, HTML का एक कठोर और अधिक XML-अनुपालन संस्करण है। यह XML सिंटैक्स नियमों का पालन करता है, जिसमें टैग्स की उचित नेस्टिंग और क्लोजिंग शामिल होती है। XHTML की कुछ प्रमुख विशेषताएँ इस प्रकार हैं—

- सभी टैग और गुण छोटे अक्षरों (lowercase) में होने चाहिएं।
- सभी टैग्स को ठीक से नेस्ट और बंद किया जाना चाहिए (उदा. ...)।
- गुणों (attributes) के मान को डबल कोट्स (" ") में लिखना अनिवार्य है।
- <img> और
 जैसे एलिमेंट्स के लिए सेल्फ-क्लोजिंग टैग्स का प्रयोग करना होता है (उदा.)।

10.11.4 HTML में अप्रचलित टैग्स और गुण (Deprecated Tags and Attributes)

HTML के अप्रचलित टैग्स और गुण ऐसे होते हैं जिन्हें अब आधुनिक वेब विकास में प्रयोग नहीं किया जाना चाहिए। वे अभी भी कुछ ब्राउज़रों में कार्य करते हैं, लेकिन उपयोग की अनुशंसा नहीं की जाती। इनका स्थान आधुनिक तकनीकों द्वारा ले लिया गया है। कुछ उदाहरण निम्नलिखित हैं, जैसा कि **चित्र 10.20** में दर्शाया गया है।

अप्रचलित टैग्स— <font>, , <strike>, <frame>, <frameset> आदि।

अप्रचलित गुण— align, bgcolor, border, cellspacing, cellpadding आदि।

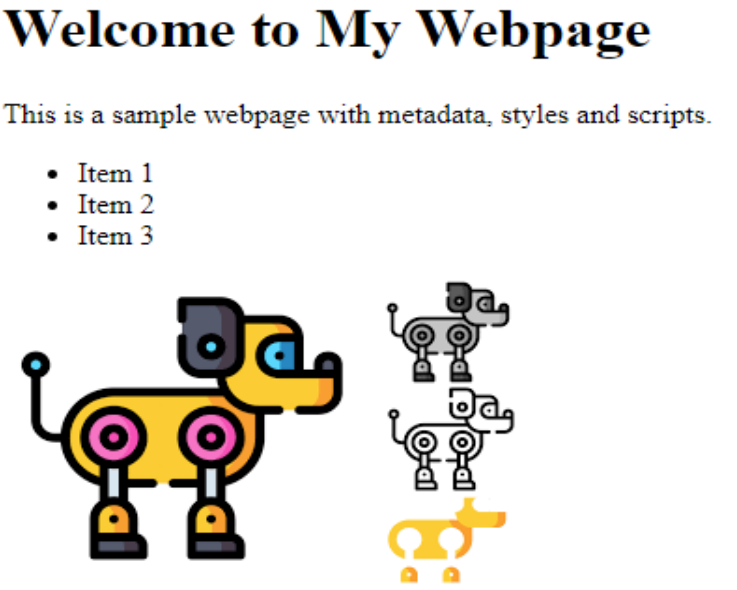
आधुनिक वेब विकास में स्टाइलिंग और लेआउट के लिए CSS का प्रयोग किया जाता है, तथा इंटरएक्टिविटी के लिए JavaScript का उपयोग किया जाता है, न कि अप्रचलित HTML गुणों और टैग्स पर निर्भर किया जाता है।

यह अत्यंत आवश्यक है कि हम HTML और वेब विकास के अद्यतन मानकों के अनुरूप कार्य करें ताकि विभिन्न डिवाइस और ब्राउज़र में अनुकूलता, पहुँच और सकारात्मक प्रयोक्ता अनुभव सुनिश्चित किया जा सके।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="author" content="Vijay Goswami">
    <meta name="description" content="A sample webpage with metadata">
    <meta name="keywords" content="HTML, CSS, JavaScript, Metadata">
    <title>Document</title>
    <link rel="stylesheet" href="metadata.css">
</head>
<body>
    <h1>Welcome to My Webpage</h1>
    <p>This is a sample webpage with metadata, styles and scripts.</p>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
    <img src="sample-image.png" alt="Sample Image">
    <script>
        // JavaScript code goes here
        console.log("Hello World!")
    </script>
</body>
</html>
```

चित्र 10.20— HTML के अप्रचलित टैग्स और गुण

आउटपुट



चित्र 10.21— HTML के अप्रचलित टैग्स और गुणों का आउटपुट

व्याख्या

इस उदाहरण में— <meta> टैग दस्तावेज़ के मेटाडेटा प्रदान करते हैं, जिसमें कैरेक्टर एनकोडिंग (charset), viewport सेटिंग्स (viewport), लेखनकर्ता (author), विवरण (description) और कीवर्ड्स (keywords) सम्मिलित होते हैं।

<title> एलिमेंट उस वेब पृष्ठ का शीर्षक परिभाषित करता है जो ब्राउज़र की शीर्ष पट्टी या टैब में प्रदर्शित होता है। <link> एलिमेंट "styles.css" नामक बाह्य CSS स्टाइलशीट को संदर्भित करता है, जो वेबपृष्ठ की स्टाइलिंग हेतु प्रयोग की जाती है।

<img> एलिमेंट एक सेल्फ-क्लोजिंग टैग है जो चित्र प्रदर्शित करता है। <script> एलिमेंट वेब पृष्ठ में JavaScript कोड सम्मिलित करता है। इस उदाहरण में, यह ब्राउज़र कंसोल में एक संदेश लॉग करता है।

यह उदाहरण विभिन्न मेटाडेटा-संबंधी एलिमेंटों और गुणों, <meta> टैग, तथा XHTML सिंटैक्स जैसे सेल्फ-क्लोजिंग टैग्स (<img />) के प्रयोग को प्रदर्शित करता है।

सारांश

- HTML सूचियाँ तीन प्रकार की होती हैं — अननुक्रमित, अनुक्रमित और परिभाषा सूची।
- अननुक्रमित सूचियों में और <li> का प्रयोग होता है, जहाँ क्रम का कोई महत्व नहीं होता।
- अनुक्रमित सूचियों में और <li> का प्रयोग होता है, जहाँ क्रम महत्वपूर्ण होता है (जैसे क्रमांक)।
- परिभाषा सूचियों में <dl>, <dt>, और <dd> का प्रयोग शब्दों और उनके विवरण के लिए किया जाता है।
- चित्रों को <img> टैग द्वारा सम्मिलित किया जाता है, जिसमें "src" और "alt" जैसे गुण होते हैं।
- इमेज मैप्स में <map> और <area> का प्रयोग किया जाता है जिससे चित्रों में क्लिक करने योग्य क्षेत्र बनाए जाते हैं।
- हाइपरलिंक्स <a> टैग द्वारा बनाए जाते हैं और "href" गुण का प्रयोग गंतव्य URL के लिए किया जाता है।
- URL के घटक होते हैं — स्कीम, डोमेन, पोर्ट, पथ (path), क्वेरी और अंश (fragment)।
- URL एन्कोडिंग में आरक्षित अक्षरों को "%xx" कोड्स में बदला जाता है।
- HTML में तालिकाएँ <table>, <tr>, <th>, और <td> एलिमेंट्स से संरचित होती हैं।
- <caption> एलिमेंट तालिका का शीर्षक प्रदान करता है।
- <tfoot>, <tbody>, और <colgroup> जैसे एलिमेंट्स तालिका की संरचना और स्टाइलिंग को बेहतर बनाते हैं।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. HTML में एक अव्यवस्थित सूची (unordered list) बनाने के लिए किस टैग का उपयोग किया जाता है?
   (क)
   (ख)
   (ग)
   (घ) <dl>
2. किस प्रकार की सूची का उपयोग ऐसे बिंदुओं के लिए किया जाता है जिनका एक निश्चित क्रम होता है?
   (क) अव्यवस्थित सूची
   (ख) क्रमबद्ध सूची
   (ग) परिभाषा सूची
   (घ) हाइपरलिंक सूची

3. HTML की क्रमबद्ध सूची में प्रत्येक आइटम से पहले सामान्यतः क्या होता है?
   (क) बुलेट बिंदु
   (ख) संख्याएँ या अक्षर
   (ग) हाशिया (इंडेंटेशन)
   (घ) हाइपरलिंक

4. HTML में परिभाषा सूची को परिभाषित करने के लिए किस टैग का उपयोग किया जाता है?
   (क)
   (ख)
   (ग)
   (घ) <dl

5. HTML में परिभाषा सूची का उद्देश्य क्या होता है?
   (क) नेविगेशन मेनू बनाना
   (ख) सूचीबद्ध आइटम बनाना
   (ग) शब्दों व उनके विवरण प्रस्तुत करना
   (घ) तालिकाएँ बनाना

6. किस HTML टैग का उपयोग वेब पृष्ठ में सीधे चित्र प्रदर्शित करने के लिए किया जाता है?
   (क) <image>
   (ख) <img>
   (ग) <picture>
   (घ) <link>

7. <img> टैग का "alt" गुण किसका संकेत करता है?
   (क) चित्र स्रोत URL
   (ख) चित्र का आकार
   (ग) चित्र के लिए वैकल्पिक टेक्स्ट
   (घ) चित्र का कैप्शन

8. चित्र मैपिंग का उपयोग किसलिए किया जाता है?
   (क) वेब पृष्ठों में चित्र डालने हेतु
   (ख) चित्र के अंदर क्लिक करने योग्य क्षेत्र बनाने हेतु
   (ग) चित्र के रंग बदलने हेतु
   (घ) चित्र के आयाम समायोजित करने हेतु

9. एक इमेज मैप बनाने के लिए, HTML में किस टैग का प्रयोग चित्र को सीमांकित करने के लिए किया जाता है?
   (क) <img>
   (ख) <map>
   (ग) <area>
   (घ) <a>

10. HTML हाइपरलिंक में "href" गुण क्या निर्दिष्ट करता है?
    (क) चित्र स्रोत URL
    (ख) लिंक का गंतव्य

(ग) हाइपरलिंक का टेक्स्ट

(घ) लिंक का रंग

ख. रिक्त स्थान भरें

1. HTML में एक अव्यवस्थित सूची को परिभाषित करने के लिए _____ टैग का उपयोग किया जाता है।
2. एक _____ सूची में प्रत्येक आइटम के आगे सामान्यतः उसकी स्थिति को दर्शाने के लिए संख्या या अक्षर होते हैं।
3. क्रमबद्ध सूची बनाने के लिए _____ टैग का उपयोग किया जाता है।
4. _____ सूची का उपयोग शब्दों और उनके विवरण या परिभाषाओं को प्रदर्शित करने हेतु किया जाता है।
5. परिभाषा सूची को परिभाषित करने के लिए _____ टैग का प्रयोग होता है।
6. वेब पृष्ठ में चित्र प्रदर्शित करने के लिए _____ टैग का उपयोग किया जाता है।
7. <img> टैग का "alt" गुण चित्र के लिए _____ प्रदान करता है।
8. चित्र _____ आपको किसी चित्र पर क्लिक योग्य क्षेत्र परिभाषित करने की सुविधा देता है।
9. HTML में, <a> टैग का उपयोग करके आप _____ बना सकते हैं।
10. _____ टैग तालिका के मुख्य भाग को परिभाषित करता है।

ग. सत्य या असत्य लिखिए

1. अव्यवस्थित सूची का उपयोग मुख्यतः ऐसे बिंदुओं के लिए किया जाता है जिनमें विशेष क्रम आवश्यक होता है।
2. HTML में क्रमबद्ध सूची बनाने के लिए <ol> टैग का उपयोग किया जाता है।
3. परिभाषा सूचियों का उपयोग सामान्यतः नेविगेशन मेनू बनाने हेतु किया जाता है।
4. <img> टैग का उपयोग वेब पृष्ठ में वीडियो जोड़ने के लिए किया जाता है।
5. <img> टैग का "alt" गुण चित्र को शीर्षक प्रदान करने हेतु होता है।
6. चित्र मैपिंग का उपयोग इंटरेक्टिव आरेख जैसे चित्रों पर क्लिक करने योग्य क्षेत्र बनाने हेतु किया जाता है।
7. चित्र मैपिंग में <map> टैग चित्र पर क्लिक करने योग्य क्षेत्र निर्दिष्ट करता है।
8. HTML में <a> टैग का उपयोग हाइपरलिंक बनाने के लिए किया जाता है।
9. हाइपरलिंक में "href" गुण वेब पृष्ठ पर दिखाई देने वाले लिंक टेक्स्ट को निर्दिष्ट करता है।
10. URL एन्कोडिंग का उपयोग URL में आरक्षित अक्षरों को इस प्रकार निरूपित करने हेतु किया जाता है जिससे उनका सही रूप से प्रसारण हो सके।

घ. लघु उत्तर प्रश्न

1. HTML में अव्यवस्थित सूची का उद्देश्य क्या होता है और इसे किस प्रकार परिभाषित किया जाता है?
2. अव्यवस्थित सूची बनाने में और <li> टैग का उपयोग किस प्रकार किया जाता है, समझाइए।
3. HTML में आप किस स्थिति में क्रमबद्ध सूची का उपयोग करेंगे और इसे कैसे परिभाषित किया जाता है?
4. क्रमबद्ध सूची बनाने के लिए प्रयुक्त तत्वों और उनके टैगों का वर्णन कीजिए।
5. परिभाषा सूचियों का क्या उपयोग है और HTML में इन्हें बनाने के लिए किन टैगों का प्रयोग किया जाता है?
6. परिभाषा सूची की संरचना समझाइए, जिसमें <dl>, <dt>, और <dd> टैग शामिल हैं।
7. HTML वेब पृष्ठ में <img> टैग का उपयोग करके चित्र किस प्रकार जोड़े जाते हैं?
8. <img> टैग के "alt" गुण का क्या उद्देश्य है और यह क्यों आवश्यक है?
9. HTML में चित्र मैपिंग की अवधारणा को समझाइए तथा उसका उपयोग स्पष्ट कीजिए।
10. HTML में हाइपरलिंक कैसे बनाए जाते हैं और इसके लिए किस टैग का प्रयोग होता है?

सत्र 11
सीएसएस (CSS)

आइए मिलते हैं अंकित से, जो एक ऐसा बच्चा है जिसे इंटरनेट बहुत पसंद है। उसने "CSS" के बारे में सीखा। CSS वेब पृष्ठों के लिए एक जादुई रंग की तरह थी। इसकी मदद से वह कर सकता था— रंग बदलना, फॉन्ट्स को स्टाइल करना, लेआउट को सजाना, और अलंकरण जोड़ना। अंकित ने CSS के साथ अपने वेब पेज को आश्चर्यजनक रूप से सुंदर बना दिया, जैसे वह एक डिजिटल कलाकार हो। वह मज़ेदार और स्टाइलिश वेब पेज बना सकता था। CSS इंटरनेट के लिए एक जादुई ब्रश की तरह थी, जो उसे रंगीन और रोमांचक बना देती थी। यह ऐसा था जैसे वेब पेजों को शानदार बनाने के लिए एक विशेष उपकरण हो!

चित्र 11.1— CSS सीखता हुआ अंकित

इस अध्याय में, आप CSS, इसके संस्करणों, सिंटैक्स, CSS के प्रकार, एकाधिक स्टाइल शीट्स, तथा स्टाइल एलिमेंट्स की पृष्ठभूमि और बॉर्डर के बारे में जानेंगे।

11.1 CSS (कैस्केडिंग स्टाइल शीट्स)

CSS, जिसका पूर्ण रूप **कैस्केडिंग स्टाइल शीट्स** है, वेब विकास के क्षेत्र में अत्यंत महत्वपूर्ण स्टाइलशीट भाषा है। इसका मुख्य उद्देश्य HTML (हाइपरटेक्स्ट मार्कअप लैंग्वेज) दस्तावेजों की प्रस्तुति और शैलीगत विशेषताओं को परिभाषित और नियंत्रित करना है। एक आधारभूत तकनीक के रूप में, CSS वेब पृष्ठों की व्यवस्था, सौंदर्यशास्त्र और दृश्य स्वरूप पर अधिकार रखने में महत्वपूर्ण भूमिका निभाती है।

क्या आप जानते हैं?
CSS सामान्यतः HTML के साथ उपयोग में ली जाती है ताकि वेब पृष्ठों और यूज़र इंटरफेस की शैली को बदला जा सके।

11.1.1 CSS सिंटैक्स (वाक्यविन्यास)

CSS नियम की मूल संरचना एक चयनक (Selector), एक घोषणापत्र ब्लॉक (Declaration Block) और घोषणापत्र ब्लॉक के अंदर गुण (Properties) और मान (Values) से मिलकर बनती है, जैसा कि **चित्र 11.2** में दिखाया गया है।

```css
selector {
  property: value;
}
```

चित्र 11.2— CSS का वाक्यविन्यास

- **चयनक (Selector)** — यह उस HTML तत्व को लक्षित करता है जिसे आप स्टाइल देना चाहते हैं। चयनक HTML तत्वों (प्रकार चयनक), क्लास, ID, गुण आदि से बने हो सकते हैं।
- **घोषणापत्र ब्लॉक (Declaration Block)** — इसे { } में लिखा जाता है, जिसमें एक या अधिक घोषणाएँ अर्धविराम (;) से अलग की जाती हैं।
- **गुण (Property)** — यह वह स्टाइल विशेषता है जिसे आप बदलना चाहते हैं (जैसे— रंग, चौड़ाई, फ़ॉन्ट-आकार)।
- **मान (Value)** — यह उस गुण को दिया गया मान होता है (जैसे— #333, 15px, bold)।

1. चयनक (Selectors)

चयनक ऐसे पैटर्न होते हैं जिनका उपयोग उन HTML तत्वों को चुनने और लक्षित करने के लिए किया जाता है जिन्हें आप स्टाइल देना चाहते हैं। CSS नियम इन्हीं चयनकों के आधार पर तत्वों पर लागू होते हैं। कुछ सामान्य चयनक इस प्रकार हैं—

तत्व चयनक (Element Selector) — किसी विशेष HTML तत्व के सभी उदाहरणों का चयन करता है। उदाहरण के लिए—

```
css
p {
  /* सभी <p> तत्वों के लिए शैली */
}
```

क्लास चयनक (Class Selector) — किसी विशिष्ट class विशेषता वाले तत्वों का चयन करता है। यह बिंदु (.) से आरंभ होता है।

```
css
.highlight {
  /* class="highlight" वाले तत्वों के लिए शैली */
}
```

ID चयनक (ID Selector) — किसी विशिष्ट id विशेषता वाले तत्व का चयन करता है। यह हैश (#) से आरंभ होता है।

```
css
#header {
  /* id="header" वाले तत्व के लिए शैली */
}
```

वंशज चयनक (Descendant Selector) — किसी विशेष तत्व के अंदर आने वाले तत्वों का चयन करता है।

```
css

ul li {
  /* <ul> के अंदर <li> तत्वों के लिए शैली */
}
```

छद्म-क्लास चयनक (Pseudo-Class Selector) — किसी विशेष अवस्था या स्थिति में तत्वों का चयन करता है।

```
css
a:hover {
  /* जब लिंक पर माउस लाया जाए तो शैली */
}
```

यूनिवर्सल चयनक (Universal Selector) — CSS में यूनिवर्सल चयनक को * से दर्शाया जाता है। यह एक विशेष चयनक होता है जो HTML दस्तावेज़ के सभी तत्वों का चयन करता है।

```
css

/* सभी तत्वों पर एक सामान्य शैली लागू करें */
* {
  margin: 0;
  padding: 0;
  border: none;
}
```

2. गुण (Properties)

गुण वे CSS विशेषताएँ होती हैं जो यह परिभाषित करती हैं कि किसी HTML तत्व को किस प्रकार स्टाइल किया जाए। प्रत्येक CSS नियम में एक या अधिक गुण होते हैं, जो कोलन (:) के बाद मान के साथ दिए जाते हैं। कुछ सामान्य गुण निम्नलिखित हैं—

- **color** — टेक्स्ट का रंग सेट करता है।
- **font-size** — फ़ॉन्ट का आकार निर्धारित करता है।
- **background-color** — पृष्ठभूमि का रंग सेट करता है।
- **margin, padding** — तत्वों के चारों ओर रिक्त स्थान को नियंत्रित करता है।
- **border** — तत्वों के चारों ओर बॉर्डर परिभाषित करता है।
- **text-align** — किसी तत्व के अंदर टेक्स्ट को संरेखित करता है।
- **width, height** — किसी तत्व की चौड़ाई और ऊँचाई निर्धारित करता है।

3. मान (Values)

मान गुणों को सौंपे जाते हैं और यह परिभाषित करते हैं कि चयनित तत्वों को किस प्रकार स्टाइल किया जाएगा। मान विशिष्ट मान (जैसे— 12px, #FF0000), कुंजीशब्द (जैसे— bold, center) या क्रियात्मक स्वरूप (जैसे— rgba(255, 0, 0, 0.5)) हो सकते हैं। प्रत्येक मान अर्धविराम (;) से समाप्त होता है जिससे अनेक गुण-मूल्य युग्म अलग-अलग लिखे जा सकें।

यहाँ CSS नियम का एक मूल उदाहरण प्रस्तुत है—

```
css
/* चयनक— सभी <p> तत्वों का चयन करता है */
p {
   color: blue; /* टेक्स्ट का रंग नीला किया गया */
   font-size: 18px; /* फ़ॉन्ट का आकार 18px */
   margin-bottom: 10px; /* नीचे की ओर 10px का मार्जिन */
}
```

व्याख्या —

इस उदाहरण में चयनक p सभी <p> तत्वों को लक्षित करता है, और गुण (color, font-size, margin-bottom) यह निर्धारित करते हैं कि उन तत्वों को कैसे स्टाइल किया जाए। प्रत्येक गुण के बाद कोलन (:) और उसके बाद वांछित मान लिखा गया है।

CSS आपको एकल या संयुक्त चयनकों, गुणों और मानों के संयोजन द्वारा जटिल नियम बनाने की सुविधा देता है ताकि आप वेब तत्वों को इच्छानुसार स्टाइल कर सकें। CSS फाइलें प्रायः HTML से अलग रखी जाती हैं और एक ही स्टाइलशीट में अनेक CSS नियम परिभाषित किए जा सकते हैं, जिससे **संगतता (consistency)** और **रखरखाव (maintainability)** आसान होता है।

11.1.2 CSS की प्रमुख अवधारणाएँ और विशेषताएँ

अलगाव का सिद्धांत (Separation of Concerns) — CSS, सामग्री (HTML) को उसकी प्रस्तुति से अलग करता है, जिससे वेब डेवलपर्स को संरचित और अर्थयुक्त (semantic) HTML दस्तावेज़ बनाने की सुविधा मिलती है, जबकि सौंदर्यात्मक रूप से सज्जा (styling) को अलग से लागू किया जा सकता है।

सेलेक्टर्स (Selectors) — CSS, HTML तत्वों को लक्षित करने के लिए सेलेक्टर्स का उपयोग करता है। सेलेक्टर्स यह परिभाषित करते हैं कि किन तत्वों पर स्टाइल लागू होगी। सामान्य सेलेक्टर्स में एलिमेंट सेलेक्टर्स (जैसे— p, h1, a), क्लास सेलेक्टर्स (जैसे— .button), और आईडी सेलेक्टर्स (जैसे— #header) शामिल हैं।

प्रॉपर्टीज़ (Properties) — CSS प्रॉपर्टीज़ यह निर्धारित करती हैं कि HTML तत्व किस प्रकार स्टाइल किए जाएँगे। इनमें रंग (color), फॉन्ट आकार (font-size), मार्जिन (margin), पैडिंग (padding), पृष्ठभूमि रंग (background-color) आदि जैसे अनेक गुण शामिल होते हैं।

वैल्यूज़ (Values) — CSS प्रॉपर्टीज़ को निर्धारित करने के लिए मान (value) दिए जाते हैं, जो यह तय करते हैं कि चयनित तत्व कैसे दिखाई देंगे। उदाहरण के लिए, color: blue; टेक्स्ट का रंग नीला निर्धारित करता है।

कैस्केडिंग (Cascading) — CSS में "कैस्केडिंग" शब्द का तात्पर्य प्राथमिकता के क्रम से है, जब किसी एक ही तत्व पर कई स्टाइल लागू की जाती हैं। कैस्केडिंग आपको विभिन्न स्तरों पर स्टाइल निर्धारित करने की सुविधा देता है, जैसे प्रयोक्ता द्वारा निर्धारित शैली, लेखक शैली (author styles), और ब्राउज़र डिफॉल्ट्स।

इनहेरिटेंस (Inheritance) — कुछ CSS प्रॉपर्टीज़ पैरेंट (parent) तत्व से चाइल्ड (child) तत्वों में स्वतः स्थानांतरित हो जाती हैं। उदाहरण के लिए, यदि आप body टैग पर फॉन्ट परिवार निर्धारित करते हैं, तो वह उस टैग के अंदर उपस्थित सभी पाठों पर लागू हो जाएगा, जब तक कि उसे किसी विशेष शैली द्वारा अधिलेखित (override) न किया गया हो।

सेलेक्टर्स और कॉम्बिनेटर्स (Selectors and Combinators) — CSS, विशिष्ट तत्वों या तत्व समूहों को लक्षित करने के लिए कई प्रकार के सेलेक्टर्स और कॉम्बिनेटर्स प्रदान करता है। इनमें वंशज सेलेक्टर (descendant selectors), चाइल्ड सेलेक्टर (child selectors), आसन्न सहोदर सेलेक्टर (adjacent sibling selectors) आदि शामिल हैं।

उत्तरदायी डिज़ाइन (Responsive Design) — CSS का प्रयोग उत्तरदायी (responsive) वेब डिज़ाइन बनाने के लिए अत्यंत महत्वपूर्ण है, जो विभिन्न स्क्रीन आकारों और यंत्रों के अनुरूप अनुकूलित होता है। मीडिया क्वेरी (media queries) का उपयोग सामान्यतः दृश्यपटल (viewport) आयामों के आधार पर स्टाइल लागू करने के लिए किया जाता है।

विभिन्न CSS फ्रेमवर्क्स — Bootstrap, Foundation, TailwindCSS, Materialize, SemanticUI आदि।

CSS का उदाहरण — यहाँ एक सरल CSS कोड उदाहरण प्रस्तुत है जो एक अनुच्छेद (paragraph) तत्व को स्टाइल करता है—

```
css
/* बाहरी स्टाइलशीट (styles.css) में प्रयुक्त CSS कोड */
p {
  color: #333; /* टेक्स्ट का रंग (गहरा स्लेटी) */
```

```
  font-size— 16px; /* फॉन्ट आकार (16 पिक्सेल) */
  line-height— 1.5; /* पंक्ति की ऊँचाई (फॉन्ट आकार का 1.5 गुना) */
  margin-bottom— 20px; /* नीचे की मार्जिन (20 पिक्सेल) */
}
```

इस उदाहरण में, CSS कोड वेबपृष्ठ पर उपस्थित सभी <p> तत्वों को लक्षित करता है और उनके टेक्स्ट का रंग, फॉन्ट आकार, पंक्ति ऊँचाई और नीचे की मार्जिन निर्धारित करता है। ये स्टाइल बाहरी स्टाइलशीट फाइल "styles.css" में परिभाषित हैं और HTML दस्तावेज़ में <link> तत्व द्वारा जोड़ी जाती हैं।

CSS एक बहुपयोगी और शक्तिशाली भाषा है जो आकर्षक और उत्तरदायी (responsive) वेब डिज़ाइन बनाने में सहायक है। यह वेब विकास में अत्यंत महत्वपूर्ण भूमिका निभाती है क्योंकि यह वेब सामग्री के रूप और संरचना पर नियंत्रण प्रदान करती है।

11.2 CSS के संस्करण

• समय के साथ CSS के विभिन्न संस्करणों और मॉड्यूल्स के रूप में विकास किया हुआ है। इसके कुछ प्रमुख संस्करण इस प्रकार हैं —

•**CSS1** — CSS का पहला संस्करण जिसमें मूलभूत स्टाइलिंग प्रॉपर्टीज़ और सेलेक्टर्स शामिल थे।

• **CSS2** — इसमें अधिक उन्नत सेलेक्टर्स, स्थिति निर्धारण (positioning), और मीडिया प्रकारों का समर्थन जोड़ा गया।

• **CSS3** — यह एक मॉड्यूलर संस्करण है जिसमें CSS3 सेलेक्टर्स, CSS3 रंग, CSS3 टेक्स्ट, CSS3 पृष्ठभूमियाँ आदि जैसे मॉड्यूल शामिल हैं। इसमें ग्रेडिएंट्स (gradients), ट्रांज़िशन (transitions), और ऐनिमेशन (animations) जैसी विशेषताएँ सम्मिलित की गईं।

• **CSS4** — यह एक अनौपचारिक शब्द है जिसका प्रयोग CSS के सतत विकास के लिए किया जाता है। इसमें नवीन विशेषताएँ और संवर्धन शामिल हैं, परंतु यह अभी सार्वजनिक रूप से जारी नहीं हुआ है।

क्या आप जानते हैं?

CSS के तीन प्रमुख संस्करण हैं—

• **CSS1** — सबसे पुराना संस्करण, वर्ष 1996 में जारी

• **CSS2** — वर्ष 1998 में जारी

• **CSS3** — नवीनतम संस्करण जिसमें कई नई विशेषताएँ और कार्यक्षमताएँ शामिल हैं

अभ्यास 11.1

निम्नलिखित सेलेक्टर्स का CSS सिंटैक्स लिखिए —

• वंशज सेलेक्टर (Descendant Selector)

• आईडी सेलेक्टर (ID Selector)

• स्यूडो-क्लास सेलेक्टर (Pseudo-Class Selector)

11.3 CSS सिंटैक्स

CSS (कैस्केडिंग स्टाइल शीट्स) का सिंटैक्स सीधा और स्पष्ट होता है, जिसमें मुख्यतः **सेलेक्टर्स, प्रॉपर्टीज़ और वैल्यूज़** शामिल होती हैं। आइए, CSS सिंटैक्स के प्रत्येक भाग को विस्तार से समझते हैं —

क्या आप जानते हैं?

इनलाइन CSS का उपयोग त्वरित और विशिष्ट स्टाइलिंग के लिए किया जाता है। इनलाइन CSS को ब्राउज़र बाहरी CSS की तुलना में अधिक तेजी से प्रोसेस करता है।

11.4 CSS के प्रकार

आइए हम आंतरिक CSS और इनलाइन CSS की अवधारणाओं को समझें, जो दोनों HTML तत्वों पर शैलियाँ लागू करने के लिए उपयोग किए जाते हैं, लेकिन इनकी कार्यप्रणाली और स्थान में अंतर होता है, जैसा कि चित्र 11.3 में दर्शाया गया है।

1. इनलाइन CSS (Inline CSS)

इनलाइन CSS में, CSS शैलियों को सीधे HTML तत्व पर "style" विशेषता (attribute) के माध्यम से लागू किया जाता है। इस विधि का उपयोग इस प्रकार किया जाता है—

प्रयोग — HTML तत्व में style विशेषता जोड़कर CSS नियमों को परिभाषित किया जाता है। यह शैली केवल उसी विशिष्ट तत्व पर लागू होती है।

प्रभाव क्षेत्र — इनलाइन CSS केवल उस तत्व तक सीमित रहती है, जिस पर वह लागू की गई हो।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inline CSS Example</title>
</head>
<body>
    <h1 style="color: green; font-family: Arial, Helvetica, sans-serif;">This is a styled heading.</h1>
    <p style="color: blue; font-size: 18px;">This is a styled paragraph.</p>
</body>
</html>
```

चित्र 11.3— CSS के प्रकार

इनलाइन CSS के लाभ

- व्यक्तिगत तत्वों पर शैलियों का सूक्ष्म नियंत्रण प्रदान करता है।
- किसी विशिष्ट तत्व के लिए त्वरित और एकबारगी शैली निर्धारण हेतु उपयोगी होता है।
- आवश्यकता पड़ने पर बाहरी और आंतरिक CSS शैलियों को अधिरोहित (override) कर सकता है।

इनलाइन CSS की सीमाएँ

- यदि एक ही शैली कई तत्वों पर लागू हो तो कोड की पुनरावृत्ति हो सकती है।
- किसी संपूर्ण वेबसाइट पर एकरूप शैली बनाए रखने के लिए उपयुक्त नहीं है।
- HTML मार्कअप अव्यवस्थित और पढ़ने में कठिन हो सकता है।

2. आंतरिक CSS (Internal CSS)

आंतरिक CSS, जिसे एम्बेडेड CSS या दस्तावेज़-स्तरीय CSS भी कहा जाता है, HTML दस्तावेज़ के अंदर सीधे CSS शैलियों को निर्दिष्ट करने की विधि है। इस विधि में सामान्यतः CSS परिभाषाओं को <style> तत्व का प्रयोग करके HTML फ़ाइल के <head> अनुभाग में रखा जाता है। इसकी कार्यप्रणाली को और स्पष्ट करने हेतु चित्र 11.4 देखें।

क्या आप जानते हैं?
आंतरिक CSS का उपयोग एक ही HTML दस्तावेज़ के कई तत्वों के लिए किया जाता है। आंतरिक CSS को एम्बेडेड कैस्केडिंग स्टाइल शीट भी कहा जाता है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Internal CSS Example</title>
    <style>
        /* CSS rules for styling paragraphs */
        p {
            color: blue;
            font-size: 18px;
        }
        /* Additional CSS rules for other elements */
        h1 {
            color: green;
            font-family: Arial, Helvetica, sans-serif;
        }
    </style>
</head>
<body>
    <h1>This is a styled heading.</h1>
    <p>This is a styled paragraph.</p>
</body>
</html>
```

चित्र 11.4— आंतरिक CSS

आंतरिक CSS के लाभ
• CSS शैलियों को उसी HTML दस्तावेज़ में रखने से देखरेख में आसानी होती है।
• दस्तावेज़ के कई तत्वों पर दोबारा उपयोग किए जाने योग्य शैलियाँ बनाई जा सकती हैं।
• छोटे से मध्यम आकार की परियोजनाओं में, जहाँ सभी शैलियाँ एक ही दस्तावेज़ में होती हैं, यह विधि उपयोगी है।

आंतरिक CSS की सीमाएँ
• बड़ी परियोजनाओं के लिए यह उतनी संगठित नहीं होती जितनी कि बाहरी CSS।
• शैलियाँ कई HTML दस्तावेज़ों में पुनः उपयोग नहीं की जा सकतीं।
• जटिल परियोजनाओं में एक ही दस्तावेज़ में सामग्री और शैलियों को मिलाना दस्तावेज़ को अव्यवस्थित कर सकता है।

2. बाहरी CSS (External CSS)

बाहरी CSS में एक पृथक CSS फ़ाइल का निर्माण किया जाता है और उसे HTML दस्तावेज़ से जोड़ा जाता है, जैसा कि चित्र 11.5 में दिखाया गया है।

क्या आप जानते हैं?
बाहरी CSS में केवल शैलियों को परिभाषित करने वाली अलग CSS फ़ाइलें होती हैं।

बाहरी CSS को लागू करने के लिए निर्देशों का विस्तृत विवरण निम्नानुसार है—

चरण 1— अपनी CSS फ़ाइल बनाएँ

एक नई CSS फ़ाइल .css एक्सटेंशन के साथ बनाएँ। आप इसे किसी भी टेक्स्ट एडिटर (text editor) या कोड एडिटर जैसे Visual Studio Code, Notepad या Sublime Text में बना सकते हैं। फ़ाइल का नाम ऐसा रखें जिससे पहचान आसान हो, जैसे— styles.css।

चरण 2— CSS शैलियों को बाहरी फ़ाइल में परिभाषित करें

अपनी CSS फ़ाइल (styles.css) में, CSS नियम उसी प्रकार लिखें जैसे आंतरिक <style> तत्व या इनलाइन style विशेषता में लिखे जाते हैं। उदाहरण—

```
css
/* styles.css */
/* अनुच्छेदों के लिए CSS नियम */
p {
   color— blue;
   font-size— 18px;
}
/* शीर्षकों के लिए CSS नियम */
h1, h2 {
   color: green;
   font-family: Arial, sans-serif;
}
```

इस उदाहरण में, हमने अनुच्छेदों (<p>) और शीर्षकों (<h1>, <h2>) के लिए CSS नियम परिभाषित किए हैं। p चयनक (selector) सभी <p> तत्वों पर प्रभाव डालेगा और उनके रंग व फ़ॉन्ट आकार को नियंत्रित करेगा। h1, h2 चयनक दोनों शीर्षक तत्वों की रंग व फ़ॉन्ट शैली को प्रभावित करेगा।

चरण 3— अपनी CSS फ़ाइल को सहेजें

CSS शैलियाँ परिभाषित करने के बाद, CSS फ़ाइल को .css एक्सटेंशन सहित सहेजें।

चरण 4— अपनी HTML फ़ाइल बनाएँ

अब एक नई HTML फ़ाइल बनाएँ या किसी मौजूदा HTML फ़ाइल का उपयोग करें जिसमें आप बाहरी CSS लागू करना चाहते हैं। इसे किसी साधारण टेक्स्ट एडिटर जैसे Notepad, Visual Studio Code या अन्य किसी एडिटर में बनाया जा सकता है।

चरण 5— अपनी HTML फ़ाइल को सहेजें

HTML फ़ाइल को .html एक्सटेंशन के साथ सहेजें। जैसे index.html, ताकि इसे पहचानना सरल हो।

चरण 6— CSS फ़ाइल को HTML फ़ाइल के समान फोल्डर में रखें

सुनिश्चित करें कि आपकी CSS फ़ाइल (styles.css) HTML फ़ाइल के साथ एक ही फोल्डर (directory) में हो। इससे संबंधित फ़ाइलों को व्यवस्थित रखना आसान होता है।

चरण 7— CSS फ़ाइल को HTML दस्तावेज़ से जोड़ें

अपनी HTML फ़ाइल के <head> भाग में <link> टैग का उपयोग करके CSS फ़ाइल को जोड़ें। कोड इस प्रकार होगा—

html

<head>

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Web Page</title>
    <!-- Link your external CSS file -->
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <!-- Your HTML content goes here -->
    <h1>This is a heading.</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

चित्र 11.5— बाहरी CSS

व्याख्या— <link> तत्व के गुण (attributes) का विवरण

- rel: HTML दस्तावेज़ और लिंक की गई संसाधन के बीच संबंध को दर्शाता है। यहाँ stylesheet इंगित करता है कि लिंक की गई फ़ाइल एक स्टाइल शीट है।
- type— लिंक की गई संसाधन के MIME प्रकार को परिभाषित करता है। CSS के लिए "text/css" प्रयोग होता है।
- href: CSS फ़ाइल के पथ (path) को दर्शाता है। यदि CSS फ़ाइल और HTML फ़ाइल एक ही फ़ोल्डर में हों तो "styles.css" जैसा सापेक्ष पथ (relative path) दिया जाता है।

चरण 7— अपनी HTML फ़ाइल सहेजें

CSS फ़ाइल को अपने HTML दस्तावेज़ में लिंक करने के बाद, HTML फ़ाइल को पुनः सहेजें।

चरण 8— HTML फ़ाइल को वेब ब्राउज़र में खोलें

अब आप अपनी HTML फ़ाइल को किसी वेब ब्राउज़र में खोल सकते हैं, ताकि यह देखा जा सके कि बाहरी CSS शैलियाँ आपकी HTML सामग्री पर कैसे लागू हुई हैं।

इन चरणों का पालन करके, आपने सफलतापूर्वक एक बाहरी CSS फ़ाइल को अपने HTML दस्तावेज़ से जोड़ दिया है, जिससे आप सामग्री (Content) और प्रस्तुति (Presentation) को अलग रखते हुए कई पृष्ठों पर एक समान शैली लागू कर सकते हैं।

कब किस प्रकार की CSS का प्रयोग करें—

जब आप एक ही HTML दस्तावेज़ में कई तत्वों पर शैलियाँ लागू करना चाहते हैं और संगठन बनाए रखना चाहते हैं, तो आंतरिक CSS (Internal CSS) का उपयोग करें। यह छोटे से मध्यम आकार की परियोजनाओं के लिए उपयुक्त होता है।

जब आपको किसी विशिष्ट तत्व पर त्वरित, एक बार की स्टाइलिंग करनी हो और बाहरी या आंतरिक स्टाइलशीट्स आवश्यक न हों, या आपको मौजूदा शैलियों को अस्थायी रूप से ओवरराइड करना हो, तो इनलाइन CSS (Inline CSS) का उपयोग करें।

व्यवहार में, अधिकांश वेब विकास परियोजनाओं में CSS के तीनों प्रकारों—बाहरी, आंतरिक और इनलाइन—का संयोजन उपयोग किया जाता है, यह परियोजना के आकार, जटिलता और विशिष्ट स्टाइलिंग आवश्यकताओं पर निर्भर करता है।

11.5 बहु-शैली पत्रक (Multiple Style Sheets)

अपने HTML दस्तावेज़ में एक से अधिक स्टाइलशीट्स का उपयोग करने से आपको विभिन्न तत्वों या वेबपृष्ठ के भागों पर अलग-अलग शैलियाँ संगठित और एप्लाइ करने की सुविधा मिलती है। नीचे दिया गया है कि HTML दस्तावेज़ में बहुशैली पत्रक का उपयोग कैसे किया जाए —

चरण 1— अपनी CSS फ़ाइलें बनाएँ

अलग-अलग शैलियों के लिए अलग-अलग CSS फ़ाइलें बनाना आरंभ करें। प्रत्येक CSS फ़ाइल का एक विशिष्ट उद्देश्य होना चाहिए या वेबपृष्ठ के किसी विशेष तत्व को लक्षित करना चाहिए। प्रत्येक फ़ाइल का नाम ऐसा रखें जिससे उसके कार्य का स्पष्ट संकेत मिले। उदाहरणार्थ—

- styles.css— पूरे पृष्ठ के लिए सामान्य शैलियाँ।
- header.css— हेडर अनुभाग के लिए शैलियाँ।
- sidebar.css— साइडबार के लिए शैलियाँ।
- footer.css— फुटर के लिए शैलियाँ।

चरण 2— HTML दस्तावेज़ में अपनी CSS फ़ाइलों को लिंक करें

अपने HTML दस्तावेज़ के <head> अनुभाग में, कई <link> एलिमेंट्स का उपयोग करके प्रत्येक CSS फ़ाइल को लिंक करें। प्रत्येक <link> टैग href एट्रिब्यूट के माध्यम से अलग-अलग CSS फ़ाइल को संदर्भित करेगा, जैसा कि **चित्र 11.6** में दर्शाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Multiple External Stylesheets Example</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="header.css">
    <link rel="stylesheet" href="sidebar.css">
    <link rel="stylesheet" href="footer.css">
</head>
<body>
    <!-- Your HTML content goes here -->
    <header>
        <h1>Welcome to My Website</h1>
    </header>

    <nav>
        <!-- Sidebar Content -->
    </nav>

    <main>
        <!-- Main Content -->
    </main>

    <footer>
        <!-- Footer Content -->
    </footer>
</body>
</html>
```

चित्र 11.6— HTML दस्तावेज़ में अपनी CSS फ़ाइलों को लिंक करें

चरण 3— CSS फ़ाइलों में शैलियों को व्यवस्थित और परिभाषित करें

अपनी प्रत्येक CSS फ़ाइल (styles.css, header.css, sidebar.css, footer.css) में उस फ़ाइल से संबंधित अनुभाग या तत्वों के लिए शैलियाँ परिभाषित कर सकते हैं।

उदाहरण—

css

```
/* styles.css */
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}
/* header.css */
header {
  background-color: #333;
  color: #fff;
}
/* sidebar.css */
```

```
nav {
  background-color: #555;
  color: #fff;
}
/* footer.css */
footer {
  background-color: #333;
  color: #fff;
  text-align: center;
}
```

प्रत्येक CSS फ़ाइल में वे शैलियाँ होती हैं जो वेबपृष्ठ के किसी विशिष्ट भाग को लक्षित करती हैं। इस उदाहरण में, हम body, header, navigation (sidebar), और footer को अलग-अलग स्टाइल कर रहे हैं।

चरण 4— अपनी वेबपृष्ठ फ़ाइल को सहेजें और पूर्वावलोकन करें

अपने HTML दस्तावेज़ को सभी CSS फ़ाइलों के साथ एक ही निर्देशिका में सहेजें। जब आप HTML फ़ाइल को वेब ब्राउज़र में खोलेंगे, तो यह सभी लिंक की गई CSS फ़ाइलों से संबंधित शैलियों को वेबपृष्ठ के संबद्ध भागों पर लागू करेगा।

इस प्रकार बहु-शैली पत्रकों के उपयोग से कोड स्वच्छ और व्यवस्थित बना रहता है, जिससे वेबसाइट के विभिन्न अनुभागों के लिए शैलियों का प्रबंधन और अपडेशन सरल हो जाता है। इससे पुन— प्रयोगशीलता और डिजाइन में एकरूपता को भी बढ़ावा मिलता है।

11.6 मान लंबाई और प्रतिशत (Value Lengths and Percentage)

CSS में, आप विभिन्न गुणों (properties) के लिए मानों को विभिन्न इकाइयों—जैसे लंबाई और प्रतिशत—में निर्दिष्ट कर सकते हैं। ये इकाइयाँ वेबपृष्ठ पर तत्वों के आकार, आयाम और स्थिति को परिभाषित करने के लिए अत्यंत महत्वपूर्ण होती हैं। आइए CSS में मान लंबाई और प्रतिशत की अवधारणाओं को समझें—

1. लंबाई मान (Length Values)— लंबाई मान किसी वस्तु के आकार या दूरी का मापन दर्शाते हैं। CSS कई प्रकार की लंबाई इकाइयों का समर्थन करता है, जिनमें से प्रत्येक का विशिष्ट प्रयोजन होता है —

- **पिक्सेल (px)—** स्क्रीन-आधारित डिज़ाइन में सबसे आम मॉड्यूल। यह प्रयोक्ता के स्क्रीन रिज़ॉल्यूशन से स्वतंत्र, एक स्थायी और सुसंगत आकार प्रदान करता है। उदाहरण— width: 200px; तत्व की चौड़ाई 200 पिक्सेल सेट करता है।
- **em—** यह मॉड्यूल पैरेंट एलिमेंट के फ़ॉन्ट आकार के सापेक्ष होती है। उदाहरण— यदि पैरेंट एलिमेंट का फ़ॉन्ट आकार 16px है, तो font-size: 1.5em; लागू करने पर चाइल्ड एलिमेंट का फ़ॉन्ट आकार 24px होगा।
- **rem—** यह "em" के समान है, लेकिन यह मूल (root) <html> एलिमेंट के फ़ॉन्ट आकार के सापेक्ष होता है। यह स्केलिंग में सुसंगतता सुनिश्चित करता है।
- **सेंटीमीटर (cm), मिलीमीटर (mm), इंच (in)—** ये इकाइयाँ प्रिंट शैलियों या भौतिक मीडिया के लिए प्रयुक्त होती हैं। उदाहरण— width: 2cm; से चौड़ाई 2 सेंटीमीटर निर्धारित होती है।
- **पॉइंट (pt)—** आमतौर पर प्रिंट मीडिया में प्रयुक्त होता है। एक पॉइंट लगभग 1/72 इंच होता है। उदाहरण— font-size: 12pt; फ़ॉन्ट आकार को 12 पॉइंट सेट करता है।
- **पाइका (pc)—** यह टाइपोग्राफी में प्रयुक्त होती है और 12 पॉइंट्स के बराबर होती है। उदाहरण— line-height: 1.5pc; पंक्ति ऊँचाई को 18 पॉइंट सेट करता है।

2. प्रतिशत मान (Percentage Values)—

प्रतिशत मान पैरेंट एलिमेंट के आकार या किसी अन्य सापेक्ष संदर्भ बिंदु का एक भाग दर्शाते हैं। इन्हें उत्तरदायी (responsive) डिज़ाइन तथा तत्वों को उनके कंटेनर के सापेक्ष स्केल करने के लिए अक्सर उपयोग किया जाता है—

● **चौड़ाई और ऊँचाई (Width and Height)—** आप किसी तत्व की चौड़ाई या ऊँचाई को उसके पैरेंट के आयामों के प्रतिशत रूप में सेट कर सकते हैं। उदाहरण— width— 50%; तत्व को उसके पैरेंट की चौड़ाई का आधा बनाता है।

● **मार्जिन और पैडिंग—** इनको प्रतिशत में निर्दिष्ट करके उत्तरदायी रिक्तियाँ बनाई जा सकती हैं। उदाहरण— margin— 5%; तत्व के पैरेंट की चौड़ाई का 5% मार्जिन बनाता है।

● **फ़ॉन्ट आकार—** फ़ॉन्ट आकारों को प्रतिशत में सेट करने से टेक्स्ट आनुपातिक रूप से स्केल होता है। उदाहरण— font-size— 120%; पैरेंट के फ़ॉन्ट आकार की तुलना में 20% अधिक आकार बनाता है।

उदाहरण—

नीचे एक उदाहरण है जिसमें लंबाई और प्रतिशत दोनों मानों का उपयोग कर एक उत्तरदायी लेआउट बनाया गया है—

```
css
.container {
  width: 80%; /* पैरेंट की चौड़ाई का 80% */
  margin: 0 auto; /* क्षैतिज रूप से केंद्रित करें */
  padding: 10px; /* कंटेनर के अंदर 10 पिक्सेल का पैडिंग */
  font-size: 16px; /* फ़ॉन्ट आकार 16 पिक्सेल */
}
.box {
  width: 200px; /* 200 पिक्सेल चौड़ाई */
  margin: 2em; /* पैरेंट के फ़ॉन्ट आकार का 2 गुना मार्जिन */
}
```

इस उदाहरण में, .container की चौड़ाई उसके पैरेंट की चौड़ाई का 80% है, और .box की चौड़ाई 200 पिक्सेल तय है। .box के लिए मार्जिन "em" मॉड्यूल में दिया गया है, जो पैरेंट के फ़ॉन्ट आकार के सापेक्ष होता है।

लंबाई और प्रतिशत मानों का कार्यनीतिक उपयोग करके आप ऐसे लचीले और उत्तरदायी वेब डिज़ाइन बना सकते हैं जो विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार स्वयं को अनुकूलित कर सकें।

चरण 7— अपनी HTML फ़ाइल सहेजें

CSS फ़ाइल को अपने HTML दस्तावेज़ में लिंक करने के बाद, HTML फ़ाइल को पुनः सहेजें।

चरण 8— HTML फ़ाइल को वेब ब्राउज़र में खोलें

अब आप अपनी HTML फ़ाइल को किसी वेब ब्राउज़र में खोल सकते हैं, ताकि यह देखा जा सके कि बाहरी CSS शैलियाँ आपकी HTML सामग्री पर कैसे लागू हुई हैं।

इन चरणों का पालन करके, आपने सफलतापूर्वक एक बाहरी CSS फ़ाइल को अपने HTML दस्तावेज़ से जोड़ दिया है, जिससे आप सामग्री (Content) और प्रस्तुति (Presentation) को अलग रखते हुए कई पृष्ठों पर एक समान शैली लागू कर सकते हैं।

अभ्यास 11.2

- इनलाइन सी.एस.एस. के लाभ और हानियाँ सूचीबद्ध कीजिए।
- एक्सटर्नल सी.एस.एस. के चरण सूचीबद्ध कीजिए।
- इंटरनल सी.एस.एस. के लाभ और हानियाँ सूचीबद्ध कीजिए।
- इंटरनल सी.एस.एस. का कोड लिखिए।

कब किस प्रकार की CSS का प्रयोग करें—

जब आप एक ही HTML दस्तावेज़ में कई तत्वों पर शैलियाँ लागू करना चाहते हैं और संगठन बनाए रखना चाहते हैं, तो आंतरिक CSS (Internal CSS) का उपयोग करें। यह छोटे से मध्यम आकार की परियोजनाओं के लिए उपयुक्त होता है।
जब आपको किसी विशिष्ट तत्व पर त्वरित, एक बार की स्टाइलिंग करनी हो और बाहरी या आंतरिक स्टाइलशीट्स आवश्यक न हों, या आपको मौजूदा शैलियों को अस्थायी रूप से ओवरराइड करना हो, तो इनलाइन CSS (Inline CSS) का उपयोग करें।
व्यवहार में, अधिकांश वेब विकास परियोजनाओं में CSS के तीनों प्रकारों—बाहरी, आंतरिक और इनलाइन—का संयोजन उपयोग किया जाता है, यह परियोजना के आकार, जटिलता और विशिष्ट स्टाइलिंग आवश्यकताओं पर निर्भर करता है।

11.5 बहु-शैली पत्रक (Multiple Style Sheets)

अपने HTML दस्तावेज़ में एक से अधिक स्टाइलशीट्स का उपयोग करना आपको विभिन्न तत्वों या वेबपृष्ठ के भागों पर अलग-अलग शैलियाँ संगठित और लागू करने की सुविधा देता है। नीचे बताया गया है कि HTML दस्तावेज़ में बहु-शैली पत्रक का उपयोग कैसे किया जाए—

चरण 1— अपनी CSS फ़ाइलें बनाएँ

अलग-अलग शैलियों के लिए अलग-अलग CSS फ़ाइलें बनाना आरंभ करें। प्रत्येक CSS फ़ाइल का एक विशिष्ट उद्देश्य होना चाहिए या वेबपृष्ठ के किसी विशेष तत्व को लक्षित करना चाहिए। प्रत्येक फ़ाइल का नाम ऐसा रखें जिससे उसके कार्य का स्पष्ट संकेत मिले। उदाहरणार्थ—

- styles.css— पूरे पृष्ठ के लिए सामान्य शैलियाँ।
- header.css— हेडर अनुभाग के लिए शैलियाँ।
- sidebar.css— साइडबार के लिए शैलियाँ।
- footer.css— फुटर के लिए शैलियाँ।

चरण 2— HTML दस्तावेज़ में अपनी CSS फ़ाइलों को लिंक करें

अपने HTML दस्तावेज़ के <head> अनुभाग में, कई <link> एलिमेंट्स का उपयोग करके प्रत्येक CSS फ़ाइल को लिंक करें। प्रत्येक <link> टैग href एट्रिब्यूट के माध्यम से अलग-अलग CSS फ़ाइल को संदर्भित करेगा, जैसा कि **चित्र 11.6** में दर्शाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Multiple External Stylesheets Example</title>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="header.css">
    <link rel="stylesheet" href="sidebar.css">
    <link rel="stylesheet" href="footer.css">
</head>
<body>
    <!-- Your HTML content goes here -->
    <header>
        <h1>Welcome to My Website</h1>
    </header>

    <nav>
        <!-- Sidebar Content -->
    </nav>

    <main>
        <!-- Main Content -->
    </main>

    <footer>
        <!-- Footer Content -->
    </footer>
</body>
</html>
```

चित्र 11.6— HTML दस्तावेज़ में अपनी CSS फ़ाइलों को लिंक करें

चरण 3— CSS फ़ाइलों में शैलियों को व्यवस्थित और परिभाषित करें

अपनी प्रत्येक CSS फ़ाइल (styles.css, header.css, sidebar.css, footer.css) में उस फ़ाइल से संबंधित अनुभाग या तत्वों के लिए शैलियाँ परिभाषित कर सकते हैं।

उदाहरण—

```
css
/* styles.css */
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}
/* header.css */
header {
```

```
  background-color: #333;
  color: #fff;
}
/* sidebar.css */
nav {
  background-color: #555;
  color: #fff;
}
/* footer.css */
footer {
  background-color: #333;
  color: #fff;
  text-align: center;
}
```

प्रत्येक CSS फ़ाइल में वे शैलियाँ होती हैं जो वेबपृष्ठ के किसी विशिष्ट भाग को लक्षित करती हैं। इस उदाहरण में, हम body, header, navigation (sidebar), और footer को अलग-अलग स्टाइल कर रहे हैं।

चरण 4— अपनी वेबपृष्ठ फ़ाइल को सहेजें और पूर्वावलोकन करें

अपने HTML दस्तावेज़ को सभी CSS फ़ाइलों के साथ एक ही निर्देशिका में सहेजें। जब आप HTML फ़ाइल को वेब ब्राउज़र में खोलेंगे, तो यह सभी लिंक की गई CSS फ़ाइलों से संबंधित शैलियों को वेबपृष्ठ के संबद्ध भागों पर लागू करेगा।

इस प्रकार बहु-शैली पत्रकों के उपयोग से कोड स्वच्छ और व्यवस्थित बना रहता है, जिससे वेबसाइट के विभिन्न अनुभागों के लिए शैलियों का प्रबंधन और अपडेशन सरल हो जाता है। इससे पुन: प्रयोगशीलता और डिजाइन में एकरूपता को भी बढ़ावा मिलता है।

11.6 मान लंबाई और प्रतिशत (Value Lengths and Percentage)

आप CSS में विभिन्न गुणों (properties) के लिए मानों को विभिन्न इकाइयों—जैसे लंबाई और प्रतिशत—में निर्दिष्ट कर सकते हैं। ये इकाइयाँ वेबपृष्ठ पर तत्वों के आकार, आयाम और स्थिति को परिभाषित करने के लिए अत्यंत महत्वपूर्ण होती हैं। आइए CSS में मान लंबाई और प्रतिशत की अवधारणाओं को समझें —

1. लंबाई मान (Length Values)—

लंबाई मान किसी वस्तु के आकार या दूरी का मापन दर्शाते हैं। CSS कई प्रकार की लंबाई इकाइयों का समर्थन करता है, जिनमें से प्रत्येक का विशिष्ट प्रयोजन होता है—

- **पिक्सेल (px)**— स्क्रीन-आधारित डिज़ाइन में सबसे आम मॉड्यूल। यह प्रयोक्ता के स्क्रीन रिज़ॉल्यूशन से स्वतंत्र, एक स्थायी और सुसंगत आकार प्रदान करता है। उदाहरण— width: 200px; तत्व की चौड़ाई 200 पिक्सेल सेट करता है।
- **em**— यह मॉड्यूल पैरेंट एलिमेंट के फ़ॉन्ट आकार के सापेक्ष होती है। उदाहरण— यदि पैरेंट एलिमेंट का फ़ॉन्ट आकार 16px है, तो font-size: 1.5em; लागू करने पर चाइल्ड एलिमेंट का फ़ॉन्ट आकार 24px होगा।
- **rem**— यह "em" के समान है, लेकिन यह मूल (root) <html> एलिमेंट के फ़ॉन्ट आकार के सापेक्ष होता है। यह स्केलिंग में सुसंगतता सुनिश्चित करता है।
- **सेंटीमीटर (cm), मिलीमीटर (mm), इंच (in)**— ये इकाइयाँ प्रिंट शैलियों या भौतिक मीडिया के लिए प्रयुक्त होती हैं। उदाहरण— width: 2cm; से चौड़ाई 2 सेंटीमीटर निर्धारित होती है।

● **पॉइंट (pt)**— आम तौर पर प्रिंट मीडिया में प्रयुक्त होता है। एक पॉइंट लगभग 1/72 इंच होता है। उदाहरण— font-size: 12pt; फ़ॉन्ट आकार को 12 पॉइंट सेट करता है।

● **पाइका (pc)**— यह टाइपोग्राफी में प्रयुक्त होती है और 12 पॉइंट्स के बराबर होती है। उदाहरण— line-height: 1.5pc; पंक्ति ऊँचाई को 18 पॉइंट सेट करता है।

2. प्रतिशत मान (Percentage Values)—

प्रतिशत मान पैरेंट एलिमेंट के आकार या किसी अन्य सापेक्ष संदर्भ बिंदु का एक भाग दर्शाते हैं। इन्हें उत्तरदायी (responsive) डिज़ाइन तथा तत्वों को उनके कंटेनर के सापेक्ष स्केल करने के लिए अक्सर उपयोग किया जाता है—

● **चौड़ाई और ऊँचाई (Width and Height)**— आप किसी तत्व की चौड़ाई या ऊँचाई को उसके पैरेंट के आयामों के प्रतिशत रूप में सेट कर सकते हैं। उदाहरण— width: 50%; तत्व को उसके पैरेंट की चौड़ाई का आधा बनाता है।

● **मार्जिन और पैडिंग**— इनको प्रतिशत में निर्दिष्ट करके उत्तरदायी रिक्तियाँ बनाई जा सकती हैं। उदाहरण— margin: 5%; तत्व के पैरेंट की चौड़ाई का 5% मार्जिन बनाता है।

● **फ़ॉन्ट आकार**— फ़ॉन्ट आकारों को प्रतिशत में सेट करने से टेक्स्ट आनुपातिक रूप से स्केल होता है। उदाहरण— font-size: 120%; पैरेंट के फ़ॉन्ट आकार की तुलना में 20% अधिक आकार बनाता है।

उदाहरण— नीचे एक उदाहरण है जिसमें लंबाई और प्रतिशत दोनों मानों का उपयोग कर एक उत्तरदायी लेआउट बनाया गया है—

```
css
.container {
  width: 80%; /* पैरेंट की चौड़ाई का 80% */
  margin: 0 auto; /* क्षैतिज रूप से केंद्रित करें */
  padding: 10px; /* कंटेनर के अंदर 10 पिक्सेल का पैडिंग */
  font-size: 16px; /* फ़ॉन्ट आकार 16 पिक्सेल */
}
.box {
  width: 200px; /* 200 पिक्सेल चौड़ाई */
  margin: 2em; /* पैरेंट के फ़ॉन्ट आकार का 2 गुना मार्जिन */
}
```

इस उदाहरण में, .container की चौड़ाई उसके पैरेंट की चौड़ाई का 80% है, और .box की चौड़ाई 200 पिक्सेल तय है। .box के लिए मार्जिन "em" मॉड्यूल में दिया गया है, जो पैरेंट के फ़ॉन्ट आकार के सापेक्ष होता है।

लंबाई और प्रतिशत मानों का कार्यनीतिक उपयोग करके आप ऐसे लचीले और उत्तरदायी वेब डिज़ाइन बना सकते हैं जो विभिन्न स्क्रीन आकारों और उपकरणों के अनुसार स्वयं को अनुकूलित कर सकें।

11.7 शैली तत्वों की पृष्ठभूमि और किनारे

11.7.1 पृष्ठभूमि शैलियाँ (Background Styles)—

पृष्ठभूमि रंग (background-color) — आप background-color प्रॉपर्टी का उपयोग करके किसी तत्व की पृष्ठभूमि का रंग निर्धारित कर सकते हैं। यह प्रॉपर्टी रंग मान (color value) को तर्क (argument) के रूप में लेती है। उदाहरणस्वरूप—

```
css
.box {
  background-color: #f0f0f0; /* पृष्ठभूमि का रंग हल्का स्लेटी निर्धारित करता है */
}
```

पृष्ठभूमि चित्र (background-image) — आप ठोस रंग के स्थान पर पृष्ठभूमि चित्र का उपयोग कर सकते हैं। background-image प्रॉपर्टी आपको एक चित्र का URL निर्दिष्ट करने की सुविधा देती है। उदाहरण—

```
css
.header {
  background-image: url('header-bg.jpg'); /* पृष्ठभूमि चित्र निर्धारित करता है */
}
```

पृष्ठभूमि पुनरावृत्ति (background-repeat) — आप background-repeat प्रॉपर्टी का उपयोग करके पृष्ठभूमि चित्र की पुनरावृत्ति को नियंत्रित कर सकते हैं। सामान्य मान हैं— repeat (डिफ़ॉल्ट), no-repeat, repeat-x और repeat-y।

```
css
.background {
  background-image: url('pattern.png');
  background-repeat: repeat-x; /* चित्र को क्षैतिज रूप से दोहराता है */
}
```

पृष्ठभूमि आकार (background-size) — यह प्रॉपर्टी यह निर्धारित करती है कि पृष्ठभूमि चित्र को उसके कंटेनर में कैसे आकार दिया जाए। आप cover, contain या विशिष्ट माप जैसे मानों का उपयोग कर सकते हैं।

```
css
.cover-image {
  background-image: url('cover-image.jpg');
  background-size: cover; /* चित्र को पूरे तत्व को ढकने हेतु स्केल करता है */
}
```

11.7.2 किनारे की शैलियाँ (Border Styles)—

किनारे की चौड़ाई (border-width) — आप border-width प्रॉपर्टी का उपयोग करके किसी तत्व की सीमा (border) की चौड़ाई निर्धारित कर सकते हैं। यह thin, medium (डिफ़ॉल्ट), thick या जैसे 2px जैसी विशिष्ट लंबाई स्वीकार करती है, जैसा कि चित्र 11.7 में दर्शाया गया है।

```
css
.bordered-element {
  border-width: 2px; /* 2-पिक्सेल चौड़ाई की सीमा निर्धारित करता है */
}
```

किनारे का रंग (border-color) — border-color प्रॉपर्टी सीमा का रंग निर्धारित करती है। आप कोई भी रंग मान निर्दिष्ट कर सकते हैं।

```
css
.bordered-element {
  border-color: #333; /* गहरा स्लेटी रंग की सीमा निर्धारित करता है */
}
```

किनारे की शैली (border-style) — आप border-style प्रॉपर्टी का उपयोग करके सीमा की शैली निर्धारित कर सकते हैं, जैसे— solid, dotted, dashed, double आदि।

```
css
.bordered-element {
  border-style: dashed; /* बिंदु-धारी शैली की सीमा निर्धारित करता है */
}
```

किनारों का वक्रण (border-radius) — यह प्रॉपर्टी तत्व के कोनों को गोल बनाती है, जिससे वक्रित किनारों वाली सीमा बनती है।

css

```
.rounded-element {
  border-radius: 10px; /* सभी चार कोनों को 10-पिक्सेल की त्रिज्या से वक्रित करता है */
}
```

संक्षिप्त सीमा प्रॉपर्टी (border) — आप border शॉर्टहैंड प्रॉपर्टी का उपयोग एक ही घोषणा में चौड़ाई, शैली और रंग निर्धारित करने के लिए कर सकते हैं।

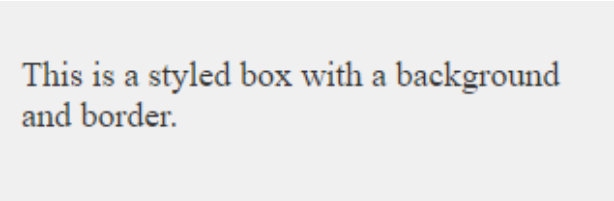
css

```
.bordered-element {
  border: 2px dashed #333; /* चौड़ाई, शैली और रंग एक पंक्ति में निर्धारित करता है */
}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Background and Border Example</title>
    <style>
        /* Styling for a container with background and border */
        .styled-box {
            width: 300px;
            height: 150px;
            background-color: #f0f0f0; /* Light gray background color */
            border: 2px solid #333; /* Dark gray solid border */
            border-radius: 10px; /* Rounded corners with a 10-pixels radius */
            padding: 20px; /* Padding inside the container */
        }
        /* Text styles within the container */
        .styled-box p {
            color: #333; /* Text color */
            font-size: 18px; /* Font size */
        }
    </style>
</head>
<body>
    <div class="styled-box">
        <p>This is a styled box with a background and border.</p>
    </div>
</body>
</html>
```

चित्र 11.7 — सीमा की शैलियाँ (Border styles)

आउटपुट —



चित्र 11.8 — सीमा शैलियों के लिए आउटपुट

अभ्यास 11.3

❖ निम्नलिखित पृष्ठभूमि तत्वों की सिंटैक्स लिखिए—

- Background Repeat
- Background Color

❖ निम्नलिखित सीमा शैलियों की सिंटैक्स लिखिए—

- Border Color
- Border Style

11.8 फॉन्ट संबंधी प्रॉपर्टीज़

CSS में, आप फॉन्ट और टेक्स्ट की शैली से संबंधित विभिन्न पहलुओं को फॉन्ट प्रॉपर्टीज़ द्वारा नियंत्रित कर सकते हैं। ये प्रॉपर्टीज़ आपको फॉन्ट परिवार, आकार, मोटाई, शैली आदि को निर्धारित करने की अनुमति देती हैं। यहाँ कुछ सामान्यतः प्रयुक्त फॉन्ट प्रॉपर्टीज़ दी जा रही हैं—

फॉन्ट परिवार (font-family) — यह प्रॉपर्टी टेक्स्ट के लिए टाइपफेस या फॉन्ट परिवार निर्धारित करती है। आप एकाधिक फॉन्ट्स को इस रूप में निर्दिष्ट कर सकते हैं कि यदि प्राथमिक फॉन्ट उपलब्ध न हो तो अन्य फॉन्ट का उपयोग हो सके।

```
css
body {
  font-family: Arial, Helvetica, sans-serif;
}
```

फॉन्ट आकार (font-size) — font-size प्रॉपर्टी फॉन्ट का आकार निर्धारित करती है। आप पिक्सेल (px), ईएम (em), प्रतिशत (%) या small, medium, large जैसे की-वर्ड्स का प्रयोग कर सकते हैं।

```
css
h1 {
  font-size: 24px;
}
```

फॉन्ट मोटाई (font-weight) — यह प्रॉपर्टी फॉन्ट की मोटाई या भार को नियंत्रित करती है। सामान्य मानों में normal, bold, bolder और 400, 700 जैसे संख्यात्मक मान होते हैं।

```
css
strong {
  font-weight: bold;
}
```

फॉन्ट शैली (font-style) — font-style प्रॉपर्टी द्वारा निर्धारित किया जाता है कि फॉन्ट को सामान्य, italic या oblique शैली में प्रदर्शित किया जाए।

```
css
em {
  font-style: italic;
}
```

पंक्ति ऊँचाई (line-height) — line-height प्रॉपर्टी से यह निर्धारित किया जाता है कि टेक्स्ट लाइनों के बीच कितनी ऊँचाई होगी। इसे यूनिट मान, प्रतिशत या की-वर्ड के रूप में सेट किया जा सकता है।

```
css
p {
  line-height: 1.5; /* पंक्ति ऊँचाई को फॉन्ट आकार का 1.5 गुना निर्धारित करता है */
}
```

टेक्स्ट सजावट (text-decoration) — text-decoration प्रॉपर्टी के माध्यम से आप टेक्स्ट पर रेखाएँ (जैसे अंडरलाइन, ओवरलाइन, स्ट्राइकथ्रू) नियंत्रित कर सकते हैं।

```
css
a {
  text-decoration: none; /* लिंक से टेक्स्ट सजावट हटाता है */
}
```

टेक्स्ट परिवर्तन (text-transform) — यह प्रॉपर्टी टेक्स्ट के केस को नियंत्रित करती है, जैसे uppercase (सभी बड़े अक्षर), lowercase (सभी छोटे अक्षर), capitalize (प्रत्येक शब्द का पहला अक्षर बड़ा)।

```
css
.uppercase {
  text-transform: uppercase; /* टेक्स्ट को बड़े अक्षरों में बदलता है */
}
```

टेक्स्ट रंग (color) — color प्रॉपर्टी टेक्स्ट का रंग निर्धारित करती है। इसमें आप hexadecimal, RGB, या नामांकित रंग मानों का प्रयोग कर सकते हैं।

```
css
h2 {
  color: #336699; /* टेक्स्ट रंग को नीले रंग की एक छाया में सेट करता है */
}
```

अक्षर रिक्ति (letter-spacing) — letter-spacing प्रॉपर्टी टेक्स्ट में अक्षरों के बीच की दूरी को नियंत्रित करती है। इसमें आप धनात्मक या ऋणात्मक मान दे सकते हैं।

```
css
.spaced-text {
  letter-spacing: 2px; /* अक्षरों के बीच की दूरी को 2 पिक्सेल से बढ़ाता है */
}
```

अभ्यास 11.4

❖ निम्नलिखित *फ़ॉन्ट-संबंधी गुणों* (font-related properties) के सिंटैक्स सूचीबद्ध कीजिए—

- फ़ॉन्ट आकार
- फ़ॉन्ट शैली

- टेक्स्ट रूपांतरण
- टेक्स्ट सज्जा

11.9 CSS सूची तालिका (List Table) का उपयोग करते हुए तालिका डेटा का लेआउट

आइए एक साधारण HTML तालिका को देखें जिसमें केवल कुछ डेटा हो, और उस पर कोई CSS स्टाइलिंग लागू न हो, जैसा कि चित्र 11.9 में दिखाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <table>
        <caption>Table of Indian States and its Capitals</caption>
        <thead>
            <tr>
                <th>SN</th>
                <th>State</th>
                <th>Capital</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>Uttar Pradesh</td>
                <td>Lucknow</td>
            </tr>
            <tr>
                <td>2</td>
                <td>Madhya Pradesh</td>
                <td>Bhopal</td>
            </tr>
            <tr>
                <td>3</td>
                <td>Maharashtra</td>
                <td>Mumbai</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

चित्र 11.9 — CSS सूची तालिका का उपयोग करते हुए तालिका डेटा का लेआउट

आउटपुट —

Table of Indian States and its Capitals

SN	State	Capital
1	Uttar Pradesh	Lucknow
2	Madhya Pradesh	Bhopal
3	Maharashtra	Mumbai

चित्र 11.10 — CSS सूची तालिका के लिए आउटपुट

तालिकाओं को CSS द्वारा स्टाइल करने के लिए हम सामान्यतः कोई अतिरिक्त class या id नहीं जोड़ते। इसके स्थान पर, हम सीधे CSS प्रॉपर्टीज़ को HTML तत्वों पर लागू करते हैं। यदि हमें किसी विशेष सेल को रंग देना हो या उस पर विशेष प्रभाव देना हो, तब हम class या id का उपयोग कर सकते हैं।

तालिका शैली जोड़ने के लिए CSS कोड —

CSS विभिन्न प्रॉपर्टीज़ प्रदान करता है जो आपको तालिकाओं की दृश्य प्रस्तुति और संरचना को नियंत्रित करने में सक्षम बनाती हैं। ये प्रॉपर्टीज़ आमतौर पर HTML के table, table row (tr), table header (th), और table cell (td) तत्वों पर लागू की जाती हैं। चित्र 11.11 में दिखाए अनुसार, तालिकाओं को अनुकूलित करने के लिए कुछ सामान्यतः प्रयुक्त CSS प्रॉपर्टीज़ इस प्रकार हैं—

बॉर्डर कोलेप्स (border-collapse) — यह प्रॉपर्टी यह निर्धारित करती है कि तालिका की सीमाएँ कैसे दिखाई दें। दो सामान्य मान हैं— collapse (सीमाएँ एक में विलीन हों) और separate (प्रत्येक सेल की अलग सीमा हो)।

```
css
table {
  border-collapse: collapse; /* सभी बॉर्डर्स को एक में विलीन करता है */
}
```

बॉर्डर रिक्ति (border-spacing) — जब border-collapse: separate होता है, तब यह प्रॉपर्टी आसन्न सेल बॉर्डर्स के बीच की दूरी निर्धारित करती है। यह दो मान लेती है— क्षैतिज और ऊर्ध्वाधर रिक्ति।

```
css
table {
  border-collapse: separate;
  border-spacing: 5px 10px; /* क्षैतिज 5px और ऊर्ध्वाधर 10px रिक्ति निर्धारित करता है */
}
```

चौड़ाई (width) और ऊँचाई (height) — ये प्रॉपर्टीज़ तालिका की चौड़ाई और ऊँचाई नियंत्रित करती हैं।

```
css
table {
  width: 100%; /* तालिका की चौड़ाई को कंटेनर के 100% तक फैलाता है */
  height: 200px; /* तालिका की ऊँचाई को 200 पिक्सेल तक सेट करता है */
}
```

टेक्स्ट संरेखण (text-align) — यह प्रॉपर्टी तालिका कोशिकाओं के अंदर टेक्स्ट का क्षैतिज संरेखण निर्धारित करती है। सामान्य मान हैं— left, center, right, justify।

```
css
th, td {
```

```
  text-align: center; /* तालिका कोशिकाओं में टेक्स्ट को केंद्र में संरेखित करता है */
}
```

ऊर्ध्वाधर संरेखण (vertical-align) — यह प्रॉपर्टी तालिका कोशिकाओं में सामग्री का ऊर्ध्वाधर संरेखण नियंत्रित करती है। सामान्य मान हैं— top, middle, bottom, baseline।

```
css
th, td {
  vertical-align: middle; /* तालिका कोशिकाओं में सामग्री को ऊर्ध्वाधर रूप से केंद्र में रखता है */
}
```

पैडिंग (padding) — padding प्रॉपर्टी सामग्री और सीमा के बीच स्थान जोड़ती है।

```
css
th, td {
  padding: 10px; /* तालिका कोशिकाओं के अंदर 10px का पैडिंग जोड़ता है */
}
```

फॉन्ट आकार (font-size) — यह प्रॉपर्टी तालिका कोशिकाओं के अंदर टेक्स्ट का आकार निर्धारित करती है।

```
css
th, td {
  font-size: 16px; /* टेक्स्ट का आकार 16 पिक्सेल सेट करता है */
}
```

पृष्ठभूमि रंग (background-color) — आप तालिका तत्वों, पंक्तियों, शीर्षकों और कोशिकाओं की पृष्ठभूमि का रंग निर्धारित कर सकते हैं।

```
css
th {
  background-color: #f0f0f0; /* तालिका शीर्षकों के लिए पृष्ठभूमि रंग निर्धारित करता है */
}
td {
  background-color: #ffffff; /* तालिका कोशिकाओं के लिए सफेद पृष्ठभूमि रंग निर्धारित करता है */
}
```

ये CSS प्रॉपर्टीज़ आपको अपनी वेब पृष्ठों में तालिकाओं के लेआउट, रूप और शैली को नियंत्रित करने की सुविधा देती हैं, जिससे आप सुंदर और सुव्यवस्थित तालिकाएँ बना सकते हैं।

```html
<!DOCTYPE html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Table Example</title>
    <style>
        table {
            border-collapse: collapse;
            width: 100%;
            border: 2px solid #333;
        }
        th, td {
            border: 1px solid #333;
        }
        th {
            background-color: #f0f0f0;
            font-weight: bold;
        }
        td {
            background-color: #ffffff;
            font-size: 16px;
        }
        /* Alternating row colors for better readability */
        tr:nth-child(even) {
            background-color: #f9f9f9;
        }
    </style>
</head>
<body>
    <table>
        <caption>Table of Indian States and its Capitals</caption>
        <thead>
            <tr>
                <th>SN</th>
                <th>State</th>
                <th>Capital</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>Uttar Pradesh</td>
                <td>Lucknow</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

चित्र 11.11 — तालिका शैलियाँ जोड़ने के लिए CSS कोड

आउटपुट

Table of Indian States and its Capitals

SN	State	Capital
1	Uttar Pradesh	Lucknow

चित्र 11.12 — तालिका शैलियों को जोड़ने के लिए CSS कोड का आउटपुट

सारांश

- CSS वेब विकास के लिए अत्यंत महत्वपूर्ण है, क्योंकि यह HTML दस्तावेज़ों की प्रस्तुति और शैली को नियंत्रित करता है।
- प्रमुख अवधारणाओं में कार्य विभाजन, चयनकर्ता, गुण, मान, कैस्केडिंग और इनहेरिटेंस शामिल हैं।
- CSS विशिष्ट तत्वों को लक्षित करने के लिए विभिन्न चयनकर्ताओं और संयोजकों की सुविधा प्रदान करता है।
- CSS को बाहरी स्टाइलशीट के रूप में जोड़ा जा सकता है, आंतरिक रूप से लागू किया जा सकता है या इनलाइन रूप में प्रयोग किया जा सकता है।
- लंबाई और प्रतिशत मानों का उपयोग तत्वों के आकार और स्थिति को परिभाषित करने के लिए किया जाता है।
- CSS आपको बैकग्राउंड और बॉर्डर्स को स्टाइल करने, फॉन्ट गुण निर्दिष्ट करने और तालिका लेआउट प्रबंधित करने की अनुमति देता है।

अपनी प्रगति जाँचें

क. बहुविकल्पी प्रश्न (MCQs)—

1. CSS का पूर्ण रूप क्या है?
   (क) कंप्यूटर स्टाइल शीट्स
   (ख) कैस्केडिंग स्टाइल शीट्स
   (ग) क्रिएटिव स्टाइल शीट्स
   (घ) सेंट्रलाइज़्ड स्टाइल शीट्स

2. CSS मुख्य रूप से HTML दस्तावेज़ों की _____ को परिभाषित करता है।
   (क) संरचना
   (ख) प्रस्तुति
   (ग) व्यवहार
   (घ) इंटरैक्शन

3. किसी तत्व की पृष्ठभूमि का रंग सेट करने के लिए कौन-सा CSS गुण प्रयोग होता है?
   (क) color
   (ख) font-size
   (ग) background-color
   (घ) margin

4. वह CSS गुण जो यह निर्धारित करता है कि किसी तत्व के अंदर टेक्स्ट कैसे संरेखित हो —
   (क) text-align
   (ख) line-height

(ग) font-style
(घ) background-color

5. CSS में कौन-सी मॉड्यूल पेरेंट तत्व (parent element) के फॉन्ट आकार के सापेक्ष होती है?
(क) px
(ख) em
(ग) cm
(घ) in

6. CSS का वह गुण जो किसी तत्व के किनारों को गोलाकार बनाता है —
(क) border-width
(ख) border-color
(ग) border-style
(घ) border-radius

7. CSS का कौन-सा गुण तालिका कोशिकाओं के अंदर सामग्री की ऊर्ध्व दिशा में संरेखण को नियंत्रित करता है?
(क) text-align
(ख) vertical-align
(ग) letter-spacing
(घ) padding

8. HTML दस्तावेज़ से बाहरी CSS फ़ाइल को जोड़ने के लिए कौन-सा HTML टैग प्रयोग किया जाता है?
(क) <style>
(ख) <link>
(ग) <script>
(घ) <head>

9. किसी तत्व के अंदर पृष्ठभूमि छवियों के आकार को निर्धारित करने वाला CSS गुण कौन-सा है?
(क) background-repeat
(ख) background-image
(ग) background-size
(घ) background-color

10. CSS में "कैस्केडिंग" का क्या तात्पर्य है?
(क) तत्वों को मोड़ने की प्रक्रिया
(ख) शैलियों को लागू करने की प्राथमिकता का क्रम
(ग) फॉन्ट का बड़े से छोटे में प्रवाह
(घ) टेक्स्ट का एनिमेशन

ख. रिक्त स्थान भरें

1. CSS मुख्यतः HTML दस्तावेज़ों की _____ और शैलीगत विशेषताओं पर केंद्रित होता है।
2. CSS में सार्वत्रिक चयनकर्ता को _____ द्वारा दर्शाया जाता है।
3. CSS में चयनकर्ता यह परिभाषित करते हैं कि कौन से _____ को स्टाइल किया जाना है।
4. CSS का वह गुण जो यह निर्धारित करता है कि किसी तत्व के अंदर टेक्स्ट कैसे संरेखित हो — _____।
5. CSS में "em" मॉड्यूल एक माप है जो _____ तत्व के फॉन्ट आकार के सापेक्ष होती है।
6. CSS का वह गुण जो टेक्स्ट के अक्षरों के बीच की दूरी को समायोजित करता है — _____।
7. CSS विशिष्ट तत्वों या तत्वों के समूह को लक्षित करने के लिए विभिन्न चयनकर्ता और संयोजक प्रदान करता है, जिनमें _____ चयनकर्ता भी शामिल हैं।
8. CSS _____ वेब डिज़ाइन बनाने में सहायक है जो विभिन्न स्क्रीन आकारों के अनुरूप ढलते हैं।
9. <link> टैग का उपयोग HTML दस्तावेज़ में बाहरी CSS फ़ाइल को जोड़ने हेतु _____ अनुभाग में किया जाता है।
10. _____ CSS वह विधि है जिसमें CSS शैलियाँ सीधे HTML दस्तावेज़ के अंदर निर्दिष्ट की जाती हैं।

ग. सत्य या असत्य

1. CSS मुख्य रूप से HTML दस्तावेज़ों की संरचना पर केंद्रित होता है।
2. CSS में "कैस्केडिंग" का अर्थ दस्तावेज़ में तत्वों को मोड़ने की प्रक्रिया है।
3. CSS में "em" मॉड्यूल <html> तत्व के फॉन्ट आकार के सापेक्ष होती है।
4. वेबसाइट भर में शैलियों को बनाए रखने के लिए इनलाइन CSS सर्वोत्तम अभ्यास है।
5. CSS गुण यह निर्धारित करते हैं कि HTML तत्व कैसे संरचित और व्यवस्थित होंगे।
6. बाहरी CSS का उपयोग HTML दस्तावेज़ में शैलियों को सीधे लागू करने के लिए किया जाता है।
7. CSS विभिन्न स्क्रीन आकारों और उपकरणों के अनुरूप ढलने वाले प्रतिक्रियाशील (responsive) वेब डिज़ाइन बनाने की अनुमति देता है।
8. <link> टैग का उपयोग <body> अनुभाग में बाहरी CSS फ़ाइल को जोड़ने के लिए किया जाता है।
9. CSS में "rem" मॉड्यूल एक अनौपचारिक माप है जो संदर्भ के अनुसार बदलती रहती है।
10. एक HTML दस्तावेज़ में एक से अधिक स्टाइलशीट का उपयोग नहीं किया जा सकता।

घ. लघु उत्तर प्रश्न

1. वेब विकास में CSS का प्रमुख उद्देश्य क्या है?
2. CSS वेब विकास में कार्य विभाजन कैसे करता है?
3. CSS में "चयनकर्ताओं" की अवधारणा को समझाइए और उदाहरण दीजिए।
4. तत्वों की शैली निर्धारण हेतु प्रयुक्त कुछ सामान्य CSS गुण कौन-से हैं?
5. CSS के संदर्भ में "कैस्केडिंग" का क्या तात्पर्य है?
6. CSS गुणों के लिए इनहेरिटेंस (विरासत) कैसे कार्य करता है?
7. मीडिया क्वेरी (media queries) क्या हैं, और CSS में इनका महत्व क्या है?
8. आप किसी HTML दस्तावेज़ में बाहरी CSS फ़ाइल को कैसे जोड़ सकते हैं?
9. आंतरिक CSS के लाभ और सीमाएँ क्या हैं?
10. वेब विकास में आप इनलाइन CSS का उपयोग कब करेंगे?

सत्र 12
CSS बॉक्स मॉडल

अतुल, एक जिज्ञासु बालक, "CSS बॉक्स मॉडल" के बारे में जानता है, जो वेब पृष्ठ पर वस्तुओं को एक पहेली की तरह व्यवस्थित करने में सहायता करता है। इसमें चार भाग होते हैं— **सामग्री (Content)** (जहाँ टेक्स्ट और चित्र होते हैं), **पैडिंग (Padding)** (सामग्री के चारों ओर की रिक्त जगह), **बॉर्डर (Border)** (सामग्री के चारों ओर का फ्रेम), और **मार्जिन (Margin)** (पूरे बॉक्स के चारों ओर की खाली जगह)। इस मॉडल के माध्यम से अतुल ने वेब पृष्ठों को अपने मनमुताबिक व्यवस्थित व आकर्षक तरीके से डिज़ाइन करना सीखा — वह अब एक पहेली हल करने वाला और एक वेब पृष्ठ डिज़ाइनर, दोनों बन गया। इससे इंटरनेट और भी अधिक रोचक और देखने योग्य बन गया, जैसा कि चित्र 12.1 में दर्शाया गया है।

चित्र 12.1— वेबसाइट के लिए CSS टूल्स का उपयोग करता हुआ अतुल

इस अध्याय में आप CSS बॉक्स मॉडल, CSS डाइमेंशन (आयाम), CSS पोज़िशनिंग, CSS विज़िबिलिटी, CSS डिस्प्ले और CSS स्क्रॉलबार्स के बारे में जानेंगे।

12.1 CSS बॉक्स मॉडल

CSS बॉक्स मॉडल वेब डिज़ाइन और लेआउट की एक मौलिक अवधारणा है, जो यह निर्धारित करता है कि वेब पृष्ठ पर तत्वों की संरचना कैसी होगी और उनके आयाम कैसे गणना किए जाएँगे। इस मॉडल में चार मुख्य घटक होते हैं—

1. **सामग्री (Content)**— बॉक्स का सबसे भीतरी भाग सामग्री क्षेत्र कहलाता है, जिसमें तत्व की वास्तविक सामग्री होती है, जैसे कि टेक्स्ट, चित्र या अन्य HTML तत्व। सामग्री क्षेत्र के आयाम width और height गुणों द्वारा निर्धारित होते हैं।
2. **पैडिंग (Padding)**— पैडिंग एक अदृश्य क्षेत्र है जो सामग्री के चारों ओर होता है और इसका आकार padding गुण द्वारा निर्धारित किया जाता है। यह सामग्री और बॉर्डर के बीच स्थान बनाता है। आप शीर्ष (top), दायाँ (right), नीचे (bottom), और बायाँ (left) — सभी पक्षों के लिए अलग-अलग पैडिंग मान निर्धारित कर सकते हैं या एक ही मान सभी पक्षों पर समान रूप से लागू कर सकते हैं।
3. **बॉर्डर (Border)**— बॉर्डर तत्व की सामग्री और पैडिंग को घेरे रहता है और इसे border गुण द्वारा परिभाषित किया जाता है। यह तत्व के चारों ओर एक दृश्यमान सीमा बनाता है जिसमें आप चौड़ाई, शैली और रंग जैसे गुण निर्दिष्ट कर सकते हैं।
4. **मार्जिन (Margin)**— मार्जिन बॉर्डर के चारों ओर एक बाहरी, पारदर्शी क्षेत्र होता है और इसका आकार margin गुण द्वारा निर्धारित होता है। यह वेब पृष्ठ पर तत्व और अन्य आस-पास के तत्वों के बीच रिक्त स्थान बनाता है। पैडिंग की तरह, मार्जिन के मान भी प्रत्येक पक्ष के लिए अलग-अलग या एक समान रूप में दिए जा सकते हैं।

क्या आप जानते हैं?
CSS बॉक्स मॉडल का उपयोग वेब पृष्ठों की डिज़ाइन और लेआउट तैयार करने में किया जाता है।

नीचे दिए गए कोड में CSS बॉक्स मॉडल के उपयोग का एक उदाहरण दिखाया गया है—

css

/* सामग्री की चौड़ाई और ऊँचाई निर्धारित करें */

.box {

/* आयाम निर्धारित करें */

width: 200px;

height: 100px;

/* पैडिंग जोड़ें */

padding: 10px;

/* बॉर्डर जोड़ें */

border: 2px solid #333;

/* मार्जिन जोड़ें */

margin: 20px;

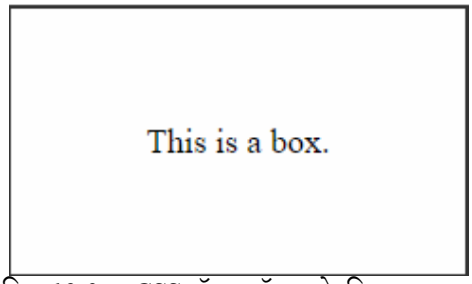
}

इस उदाहरण में, हमारे पास एक .box नामक तत्व है जिसमें विशिष्ट आयाम, पैडिंग, बॉर्डर और मार्जिन हैं। बॉक्स तत्व के कुल आयाम इन सभी गुणों के आधार पर गणना किए जाते हैं, जैसा कि चित्र 12.2 में दिखाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Box Model Example</title>
    <style>
        /* Define Styling for the box */
        .box {
            width: 200px;
            height: 100px;
            padding: 20px;
            border: 2px solid #333;
            margin: 30px;
            background-color: #fefefe;
            text-align: center;
            line-height: 100px;
            font-size: 18px;
        }
        /* Defining Styling for the container */
        .container {
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">
            This is a box.
        </div>
    </div>
</body>
</html>
```

चित्र 12.2— CSS बॉक्स मॉडल

आउटपुट



चित्र 12.3— CSS बॉक्स मॉडल के लिए आउटपुट

इस उदाहरण में—

- हमने एक div तत्व बनाया है जिसकी वर्ग (class) .box है, जो बॉक्स के रूप में प्रस्तुत होता है।
- इस बॉक्स पर CSS स्टाइल लागू किए गए हैं जो CSS बॉक्स मॉडल के गुणों को दर्शाते हैं—
- width और height सामग्री क्षेत्र के आयाम निर्धारित करते हैं।
- padding सामग्री के अंदर रिक्त स्थान जोड़ता है।
- border पैडिंग के चारों ओर बॉर्डर जोड़ता है।
- margin पूरे बॉक्स के चारों ओर रिक्त स्थान बनाता है।
- background-color, text-align, line-height, और font-size जैसे गुण अतिरिक्त सज्जा के लिए उपयोग किए गए हैं।
- container वर्ग का उपयोग पृष्ठ पर बॉक्स को केंद्र में संरेखित (center-align) करने हेतु किया गया है।
- "This is a box." यह टेक्स्ट बॉक्स के अंदर रखा गया है।

आप इस कोड को किसी HTML फ़ाइल में कॉपी-पेस्ट करके वेब ब्राउज़र में खोल सकते हैं और CSS बॉक्स मॉडल को वास्तविक रूप में देख सकते हैं। आप विभिन्न मानों को परिवर्तित कर सकते हैं और CSS बॉक्स मॉडल के प्रभाव को समझने हेतु प्रयोग कर सकते हैं।

12.2 CSS डाइमेंशन (आयाम)

CSS तत्वों के आयामों को नियंत्रित करने के लिए कई गुण प्रदान करता है। मुख्य डाइमेंशन संबंधित गुणों में शामिल हैं— width, height, min-width, max-width, min-height, और max-height। प्रत्येक गुण का विवरण नीचे दिया गया है—

width और height— ये गुण तत्व के सामग्री बॉक्स (content box) की चौड़ाई और ऊँचाई निर्धारित करते हैं, जिसमें केवल सामग्री होती है — पैडिंग, बॉर्डर और मार्जिन शामिल नहीं होते। आप इन आयामों को पिक्सेल (px), प्रतिशत (%), ईएम (em) आदि जैसे विभिन्न इकाइयों में निर्दिष्ट कर सकते हैं।

```
css
.element {
  width: 300px;
  height: 150px;
}
```

min-width और min-height— ये गुण किसी तत्व की न्यूनतम चौड़ाई और ऊँचाई निर्धारित करते हैं। भले ही तत्व के अंदर की सामग्री कम जगह लेती हो, फिर भी यह तत्व इन न्यूनतम मानों से छोटा नहीं होगा।

```
css
.element {
  min-width: 100px;
```

```
  min-height: 50px;
}
```

max-width और max-height— ये गुण किसी तत्व की अधिकतम चौड़ाई और ऊँचाई निर्धारित करते हैं। यदि सामग्री तत्व को स्वाभाविक रूप से बड़ा बनाती है, तो भी यह इन अधिकतम मानों से अधिक नहीं होगा।

```
css
.element {
  max-width: 500px;
  max-height: 300px;
}
```

इन डाइमेंशन संबंधी गुणों के साथ, आप वेब पृष्ठ पर तत्वों के आकार और बॉर्डर्स को सटीक रूप से नियंत्रित कर सकते हैं। इस स्तर का नियंत्रण आपको उत्तरदायी डिज़ाइन (responsive design) तैयार करने में सक्षम बनाता है, जो विभिन्न स्क्रीन आकारों के अनुसार समायोजित होते हैं; यह सुनिश्चित करता है कि सामग्री निर्दिष्ट बॉर्डर्स के अंदर बनी रहे, और आपके वेब पृष्ठों का समग्र लेआउट और स्वरूप संरक्षित रहे — जैसा कि चित्र 12.4 में दिखाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Dimensions Example</title>
    <style>
        /* Define a container with fixed width and height */
        .container {
            width: 300px;
            height: 200px;
            background-color: #f0f0f0;
            text-align: center;
        }
        /* define a box with minimum and maximum dimensions */
        .box {
            min-width: 100px;
            max-width: 250px;
            min-height: 50px;
            max-height: 150px;
            background-color: #3498db;
            color: #ffffff;
            padding: 10px;
            margin: 20px auto;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">
            This box has dimension constraints.
        </div>
    </div>
</body>
</html>
```

चित्र 12.4— CSS डाइमेंशन

आउटपुट

चित्र 12.5— CSS डायमेंशन (आयाम) का आउटपुट

अभ्यास 12.1

- CSS बॉक्स मॉडल के प्रमुख घटकों के नाम सूचीबद्ध कीजिए।
- CSS डायमेंशन (Dimensions) की प्रॉपर्टीज़ के नाम सूचीबद्ध कीजिए।

12.3 CSS द्वारा प्रदर्शित स्थिति (पोजिशनिंग)

CSS विभिन्न प्रॉपर्टीज़ प्रदान करता है जो आपको वेब पृष्ठ पर तत्वों की स्थिति और रूप को नियंत्रित करने में सक्षम बनाते हैं। नीचे पोजिशनिंग और डिस्प्ले से संबंधित कुछ सामान्यतः उपयोग की जाने वाली CSS प्रॉपर्टीज़ दी गई हैं—

position — यह प्रॉपर्टी किसी तत्व के लिए प्रयुक्त पोजिशनिंग विधि को निर्दिष्ट करती है। इसमें static (डिफ़ॉल्ट), relative, absolute, fixed या sticky जैसे मान हो सकते हैं। ये मान यह निर्धारित करते हैं कि कोई तत्व अपने परिवेष्ठित तत्व (containing element) या दृश्य क्षेत्र (viewport) में कैसे स्थित होगा।

```
css
.element {
  position: relative; /* सापेक्ष पोजिशनिंग का उपयोग करें */
}
```

top, right, bottom, left — इन प्रॉपर्टीज़ का उपयोग position प्रॉपर्टी के साथ मिलकर किसी तत्व को सटीक रूप से स्थित करने के लिए किया जाता है। ये उस तत्व के परिवेष्ठित भाग से ऊपर, दाएँ, नीचे या बाएँ किनारे की दूरी को दर्शाती हैं।

```
css
.element {
  position: absolute; /* पूर्ण पोजिशनिंग का उपयोग करें */
  top: 20px;
  left: 30px;
}
```

display — यह प्रॉपर्टी दस्तावेज़ प्रवाह (document flow) में किसी तत्व को कैसे प्रदर्शित किया जाए, इसे नियंत्रित करती है। सामान्य मानों में block, inline, inline-block, none आदि शामिल हैं। यह अन्य लेआउट प्रॉपर्टीज़ के व्यवहार को प्रभावित करती है।

```
css
.element {
  display: inline-block; /* इनलाइन-ब्लॉक तत्व के रूप में प्रदर्शित करें */
}
```

float — इस प्रॉपर्टी का उपयोग किसी तत्व को उसके परिवेष्ठित तत्व के अंदर बाएँ या दाएँ तरफ तैरता हुआ (float) बनाने के लिए किया जाता है। इसका उपयोग सामान्यतः बहु-स्तंभीय लेआउट बनाने या चित्रों के चारों ओर टेक्स्ट लपेटने के लिए किया जाता है।

```
css
.element {
  float: left; /* तत्व को बाएँ तैराएँ */
}
```

clear — यह प्रॉपर्टी किसी तत्व को फ्लोटिंग तत्व के साथ समान पंक्ति में आने से रोकने के लिए उपयोग की जाती है। इसका प्रयोग फ्लोटेड तत्वों के बाद आने वाले तत्वों पर किया जाता है ताकि वे नीचे से प्रारंभ हों।

```
css
```

```
.clear-element {
  clear: both; /* दोनों ओर के फ्लोटेड तत्वों को साफ़ करें */
}
```

12.4 CSS दृश्यता (visibility)

visibility प्रॉपर्टी किसी तत्व की दृश्यता को नियंत्रित करती है। इसके निम्नलिखित मान हो सकते हैं—

- **visible (डिफ़ॉल्ट)** — तत्व प्रदर्शित होता है।
- **hidden** — तत्व छिपा होता है लेकिन पृष्ठ पर उसकी जगह घेरता है।
- **collapse** — तालिका पंक्तियों (<tr>) और पंक्ति समूहों (<tbody>, <thead>, <tfoot>) के साथ उपयोग किया जाता है। यह पंक्ति या पंक्ति समूह को हटा देता है परन्तु तालिका का लेआउट बनाए रखता है।

उदाहरण—

```
css
.hidden-element {
  visibility: hidden; /* तत्व को छिपाएँ, लेकिन वह स्थान घेरता रहेगा */
}
```

12.5 CSS डिस्प्ले (display)

display प्रॉपर्टी यह नियंत्रित करती है कि कोई तत्व पृष्ठ पर कैसे रेंडर (प्रदर्शित) किया जाए। इसमें निम्न मान हो सकते हैं—

- **block** — तत्व को ब्लॉक-स्तरीय (block-level) तत्व के रूप में प्रदर्शित करता है, जो अपने कंटेनर की पूरी चौड़ाई लेता है।
- **inline** — तत्व को इनलाइन-स्तरीय (inline-level) तत्व के रूप में प्रदर्शित करता है, जो कंटेंट के अंदर प्रवाहित होता है।
- **inline-block** — ब्लॉक और इनलाइन दोनों के गुणों का संयोजन करता है।
- **none** — तत्व को पूरी तरह से छिपा देता है और वह कोई स्थान भी नहीं घेरता।
- **table, table-row, table-cell** — तालिका-जैसे लेआउट बनाने के लिए उपयोग किए जाते हैं।

उदाहरण—

```
css
.block-element {
  display: block; /* ब्लॉक-स्तरीय तत्व के रूप में रेंडर करें */
}
```

{visibility: hidden;} और {display: none;} में अंतर

visibility: hidden केवल तत्व को पृष्ठ पर छिपाता है, परन्तु वह पृष्ठ पर स्थान घेरता रहता है।
वहीं **display: none** न केवल तत्व को छिपा देता है, बल्कि वह किसी प्रकार का स्थान भी नहीं घेरता।

12.6 CSS स्क्रॉलबार (overflow, overflow-x, overflow-y)

overflow प्रॉपर्टी यह नियंत्रित करती है कि जब किसी तत्व की सामग्री उसके आबंटित स्थान से बाहर निकलती है तो क्या किया जाए। इसके मानों में visible, hidden, scroll, या auto हो सकते हैं, जैसा कि **चित्र 12.6** में दर्शाया गया है।

- **visible (डिफ़ॉल्ट)** — कोई स्क्रॉलबार प्रदर्शित नहीं होता, और अतिरिक्त सामग्री तत्व की सीमा से बाहर दिखाई देती है।
- **hidden** — जो सामग्री सीमा से बाहर होती है, वह काट दी जाती है और दिखाई नहीं देती।

- **scroll** — हमेशा स्क्रॉलबार प्रदर्शित करता है, भले ही सामग्री सीमा से बाहर न जाए (स्क्रॉलबार के लिए स्थान सुरक्षित करता है)।
- **auto** — केवल तभी स्क्रॉलबार प्रदर्शित करता है जब सामग्री सीमा से बाहर जाती है।

उदाहरण—

```
css
.scrollable-element {
   overflow: scroll; /* क्षैतिज और ऊर्ध्वाधर दोनों स्क्रॉलबार हमेशा दिखाएँ */
}
```

अभ्यास 12.2

• पोजिशनिंग और डिस्प्ले से संबंधित निम्नलिखित CSS प्रॉपर्टीज़ का सिंटैक्स (वाक्य विन्यास) लिखिए—

○ display

○ clear

○ position

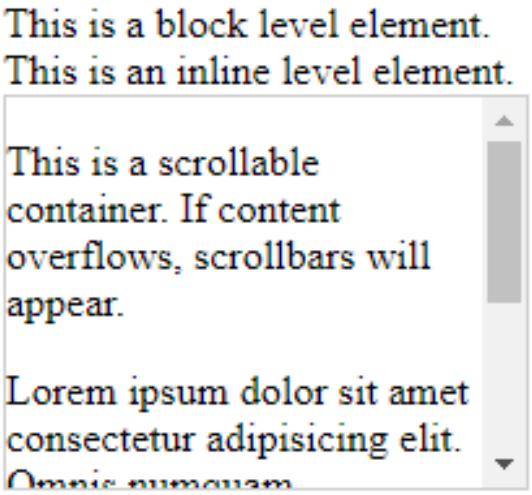
• CSS डिस्प्ले की प्रॉपर्टीज़ सूचीबद्ध कीजिए।

• CSS स्क्रॉलबार की प्रॉपर्टीज़ सूचीबद्ध कीजिए।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Visibility, Display, Scrollbars Example</title>
    <style>
        /* CSS Visibility */
        .hidden-element {
            visibility: hidden;
        }
        /* CSS Display */
        .block-element {
            display: block;
        }
        .inline-element {
            display: inline;
        }
        .none-element {
            display: none;
        }
        /* CSS Scrollbars */
        .scroll-container {
            width: 200px;
            height: 150px;
            overflow-x: hidden;
            overflow-y: scroll;
            border: 1px solid #ccc;
        }
    </style>
</head>
<body>
    <div class="hidden-element">This is a hidden element.</div>
    <div class="block-element">This is a block level element.</div>
    <div class="inline-element">This is an inline level element.</div>
    <div class="none-element">This is a hidden element.</div>
    <div class="scroll-container">
        <p>This is a scrollable container. If content overflows, scrollbars will appear.</p>
        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Omnis numquam consequatur excepturi.
            Saepe a sunt vero voluptas, aliquid iure fugiat.
            Ex dolore ab exercitationem culpa dicta distinctio molestias fuga aliquam!
        </p>
    </div>
</body>
</html>
```

चित्र 12.6— CSS स्क्रॉलबार

आउटपुट



चित्र 12.7— CSS स्क्रॉलबार का आउटपुट

12.7 CSS फ्लोट (float) प्रॉपर्टी

CSS में "float" प्रॉपर्टी का उपयोग किसी तत्व के उसके कंटेनिंग एलिमेंट (अंतर्विष्ट तत्व) के अंदर क्षैतिज (horizontal) संरेखण को नियंत्रित करने के लिए किया जाता है। जब किसी तत्व को फ्लोट किया जाता है, तो वह सामान्य दस्तावेज़ प्रवाह (document flow) से हटा दिया जाता है और अपने कंटेनर के बाएँ या दाएँ स्थित किया जाता है। इस व्यवस्था के कारण अन्य तत्व उसके चारों ओर प्रवाहित (flow) हो सकते हैं।

float प्रॉपर्टी के निम्नलिखित मान (values) हो सकते हैं —

1. **left** — तत्व को बाईं ओर फ्लोट किया जाता है और अन्य सामग्री उसकी दाहिनी ओर प्रवाहित होती है।
2. **right** — तत्व को दाहिनी ओर फ्लोट किया जाता है और अन्य सामग्री उसकी बाईं ओर प्रवाहित होती है।
3. **none (डिफ़ॉल्ट)** — तत्व फ्लोट नहीं किया जाता है और सामग्री सामान्य प्रवाह में रहती है।
4. **inherit** — तत्व अपने पैरेंट (मूल) एलिमेंट से float प्रॉपर्टी को विरासत में प्राप्त करता है।

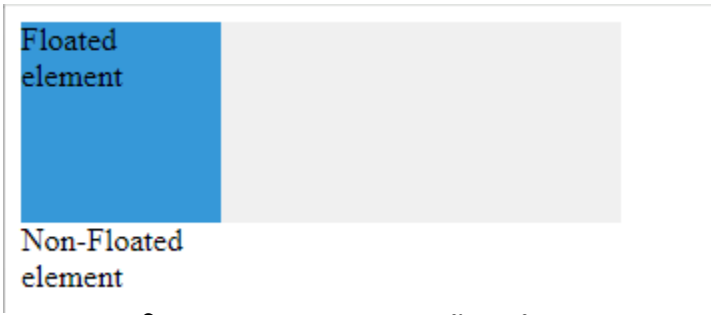
क्या आप जानते हैं?
float प्रॉपर्टी का सामान्य उपयोग छवियों के चारों ओर टेक्स्ट लपेटने (wrap) के लिए किया जाता है।

नीचे दिए गए उदाहरण में float प्रॉपर्टी के उपयोग को दर्शाया गया है, जैसा कि **चित्र 12.8** में दिखाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Float Property Example</title>
    <style>
        /* Define a container */
        .container {
            width: 300px;
            background-color: #f0f0f0;
        }
        /* Define a floated element */
        .floated {
            width: 100px;
            height: 100px;
            background-color: #3498db;
            float: left;
        }
        /* Define a non-floated element */
        .non-floated {
            width: 100px;
            height: 100px;
            background-color: #e74c3c;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="floated">
            Floated element
        </div>
        <div class="non-floated">
            Non-Floated element
        </div>
    </div>
</body>
</html>
```

चित्र 12.8 — CSS Float प्रॉपर्टी

आउटपुट —



चित्र 12.9 — CSS Float प्रॉपर्टी का परिणाम

इस उदाहरण में—

• हमारे पास एक निश्चित चौड़ाई (fixed width) वाला कंटेनर है।

• कंटेनर के अंदर दो तत्व हैं — .floated और .non-floated।

• .floated तत्व को float: left; के माध्यम से बाईं ओर फ्लोट किया गया है। परिणामस्वरूप, .non-floated तत्व .floated तत्व के दाईं ओर प्रवाहित होता है।

• .non-floated तत्व फ्लोट नहीं किया गया है और सामान्य दस्तावेज़ प्रवाह में ही रहता है।

आप इस कोड को किसी HTML फ़ाइल में कॉपी-पेस्ट करके दोहरा सकते हैं। इसके बाद, फ़ाइल को किसी वेब ब्राउज़र में खोलें और देखें कि "float" प्रॉपर्टी तत्वों की स्थिति को कैसे प्रभावित करती है।

क्या आप जानते हैं?

CSS में float प्रॉपर्टी का उपयोग किसी तत्व को उसके कंटेनर के बाएँ या दाएँ स्थित करने के लिए किया जाता है।

सारांश

- CSS बॉक्स मॉडल में चार घटक होते हैं — **content, padding, border और margin**, जो किसी तत्व की संरचना को परिभाषित करते हैं।
- **Content क्षेत्र** में वास्तविक सामग्री होती है और इसका आकार width और height प्रॉपर्टी से निर्धारित होता है।
- **Padding** सामग्री और बॉर्डर के बीच अंतराल बनाता है, जबकि **border** उसकी शैली, चौड़ाई और रंग को परिभाषित करता है।
- **Margins** बॉर्डर के बाहर का स्थान प्रदान करते हैं और पृष्ठ पर अन्य तत्वों से दूरी बनाए रखते हैं।
- CSS में width, height, min-width, max-width, min-height, max-height जैसे गुणों से माप नियंत्रित किए जा सकते हैं।
- Position प्रॉपर्टी किसी तत्व की स्थिति को निर्धारित करती है, जिसके मान हो सकते हैं— static, relative, absolute, fixed, और sticky।
- Top, right, bottom, left जैसे गुणों का उपयोग सटीक स्थिति निर्धारण के लिए किया जाता है।
- Display प्रॉपर्टी यह नियंत्रित करती है कि तत्व कैसे प्रस्तुत किए जाएँ — जैसे block, inline, inline-block, और none।
- Float प्रॉपर्टी का उपयोग किसी तत्व को बाएँ या दाएँ फ्लोट कराने के लिए किया जाता है।
- Visibility प्रॉपर्टी तत्व की दृश्यता को नियंत्रित करती है — जैसे visible, hidden, और collapse।
- Overflow प्रॉपर्टी सामग्री के अतिप्रवाह को नियंत्रित करती है — जैसे visible, hidden, scroll, और auto।
- **Float प्रॉपर्टी** कंटेनिंग एलिमेंट के अंदर तत्वों के क्षैतिज संरेखण को प्रभावित करती है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न (MCQ)

1. CSS बॉक्स मॉडल का मुख्य कार्य क्या है?
   (क) वेब तत्वों के लिए रंग परिभाषित करना
   (ख) वेब पृष्ठ पर तत्वों का क्रम निर्धारित करना
   (ग) यह तय करना कि तत्वों की संरचना और आकार कैसा हो
   (घ) तत्वों के साथ प्रयोक्ता की सहभागिता को प्रबंधित करना

2. बॉक्स मॉडल का वह घटक कौन-सा है जिसमें तत्व की वास्तविक सामग्री (जैसे टेक्स्ट या चित्र) होती है?
   (क) पैडिंग
   (ख) बॉर्डर
   (ग) कंटेंट
   (घ) मार्जिन

3. CSS में कंटेंट बॉक्स के आकार को नियंत्रित करने के लिए कौन-सी प्रॉपर्टी उपयोग में ली जाती है?
   (क) width
   (ख) padding
   (ग) border
   (घ) margin

4. कौन-सी CSS प्रॉपर्टी किसी तत्व की न्यूनतम चौड़ाई निर्धारित करती है?
   (क) min-width
   (ख) max-width
   (ग) width
   (घ) margin

5. CSS में कौन-सी प्रॉपर्टी किसी तत्व की स्थिति को उसके कंटेनर या व्यूपोर्ट में नियंत्रित करती है?
   (क) width
   (ख) height
   (ग) position
   (घ) margin

6. CSS में float प्रॉपर्टी का उद्देश्य क्या है?
   (क) किसी तत्व का फ़ॉन्ट आकार बढ़ाना
   (ख) किसी तत्व को अदृश्य करना
   (ग) किसी तत्व को सामान्य दस्तावेज़ प्रवाह से हटाना
   (घ) किसी तत्व की दृश्यता को नियंत्रित करना

7. CSS में visibility प्रॉपर्टी का वह कौन-सा मान है जो किसी तत्व को अदृश्य तो करता है, परंतु वह पृष्ठ पर स्थान घेरता है?
   (क) visible
   (ख) hidden
   (ग) collapse
   (घ) block

8. CSS में display प्रॉपर्टी का "inline-block" मान क्या करता है?
   (क) तत्व को ब्लॉक-स्तरीय तत्व की तरह प्रस्तुत करता है
   (ख) ब्लॉक और इनलाइन तत्वों दोनों के गुणों को संयोजित करता है
   (ग) तत्व को पूरी तरह से छिपा देता है
   (घ) सामग्री के अतिप्रवाह को नियंत्रित करता है

9. CSS की overflow प्रॉपर्टी में "scroll" मान क्या करता है?
   (क) स्क्रॉलबार को नहीं दिखाता
   (ख) अतिप्रवाहित सामग्री को काट देता है
   (ग) हमेशा स्क्रॉलबार दिखाता है
   (घ) स्क्रॉलबार तब ही दिखाता है जब सामग्री अतिप्रवाहित हो

10. CSS में "float: left" क्या करता है?
   (क) तत्व को बाईं ओर फ्लोट करता है, जिससे सामग्री उसकी दाईं ओर प्रवाहित होती है
   (ख) तत्व को दाईं ओर फ्लोट करता है
   (ग) तत्व को अदृश्य करता है
   (घ) तत्व की न्यूनतम चौड़ाई निर्धारित करता है

ख. रिक्त स्थान भरें

1. _____ बॉक्स मॉडल में चार प्रमुख घटक होते हैं— content, padding, border और margin।
2. Padding तत्व की सामग्री और बॉर्डर के बीच _____ उत्पन्न करता है।
3. कंटेंट क्षेत्र का आकार _____ प्रॉपर्टीज़ से निर्धारित होता है।
4. Min-width और _____ प्रॉपर्टीज़ तत्व के आकार पर प्रतिबंध लगाती हैं।
5. CSS में _____ प्रॉपर्टी का उपयोग किसी तत्व की स्थिति निर्धारित करने के लिए किया जाता है।
6. _____ प्रॉपर्टी का उपयोग अक्सर बहु-स्तंभित लेआउट (multi-column layout) बनाने के लिए किया जाता है।
7. visibility प्रॉपर्टी के मानों में शामिल हैं — visible, _____, और collapse।
8. CSS प्रॉपर्टी "display: inline-block" ब्लॉक और _____ तत्वों के गुणों को जोड़ती है।
9. "overflow" प्रॉपर्टी नियंत्रण करती है कि कोई तत्व अपनी निर्दिष्ट जगह से _____ होने पर कैसा व्यवहार करे।
10. "float: left" तत्व को बाईं ओर फ्लोट करता है, जिससे सामग्री उसकी _____ ओर प्रवाहित होती है।

ग. सही या गलत

1. किसी तत्व के content क्षेत्र का निर्धारण padding प्रॉपर्टी द्वारा किया जाता है।
2. "min-width" प्रॉपर्टी सुनिश्चित करती है कि कोई तत्व निर्दिष्ट मान से छोटा नहीं होगा।
3. "float" प्रॉपर्टी का उपयोग किसी तत्व को सामान्य दस्तावेज़ प्रवाह से हटाने के लिए किया जाता है।
4. "visibility: hidden" किसी तत्व को अदृश्य बनाता है और वह पृष्ठ पर कोई स्थान नहीं घेरता।
5. "display: inline-block" तत्व को ब्लॉक-स्तरीय तत्व की तरह प्रस्तुत करता है।
6. "overflow: scroll" तब भी स्क्रॉलबार नहीं दिखाता जब सामग्री अतिप्रवाहित हो।
7. "float: left" किसी तत्व को दाईं ओर फ्लोट करता है और सामग्री बाईं ओर प्रवाहित होती है।
8. "clear" प्रॉपर्टी का उपयोग किसी तत्व को पारदर्शी बनाने के लिए किया जाता है।
9. डिफ़ॉल्ट margin मान 0 होता है।
10. "border" प्रॉपर्टी तत्व के content बॉक्स के आकार को नियंत्रित करती है।

घ. लघु उत्तर प्रश्न

1. CSS बॉक्स मॉडल के चार प्रमुख घटक कौन-कौन से हैं?
2. CSS में किसी तत्व के content क्षेत्र का आकार कैसे निर्धारित किया जाता है?
3. CSS में "padding" प्रॉपर्टी का उद्देश्य स्पष्ट कीजिए।
4. किसी तत्व की न्यूनतम चौड़ाई और ऊँचाई पर प्रतिबंध लगाने के लिए कौन-सी CSS प्रॉपर्टी उपयोग होती है?
5. CSS में "position: relative" कैसे कार्य करता है?

6. CSS में "float" प्रॉपर्टी का उपयोग कैसे किया जाता है, और इसके सामान्य मान क्या हैं?
7. CSS में "visibility: hidden" मान क्या करता है?
8. CSS में "display: inline-block" मान का वर्णन कीजिए।
9. CSS की "overflow: scroll" प्रॉपर्टी का कार्य क्या है?
10. CSS में margin प्रॉपर्टी का डिफ़ॉल्ट मान क्या होता है?

मॉड्यूल 3
JavaScript के माध्यम से वेब विकास

परिचय

इस मॉड्यूल में, आप सर्वप्रथम "JavaScript" के बारे में सीखेंगे, जिसमें JavaScript का अवलोकन, इसकी विशेषताएँ, JavaScript और Java में अंतर, "Hello, world" संदेश को प्रदर्शित करना और ब्राउज़र ऑब्जेक्ट मॉडल सम्मिलित हैं। इसके पश्चात हम "शर्त आधारित तर्क और प्रवाह नियंत्रण" विषय पर बढ़ेंगे, जिसमें आप अभिव्यक्ति में शर्त आधारित तर्क (condition based logic), JavaScript में प्रवाह नियंत्रण और लूप्स के बारे में जानेंगे।

फिर हम "ऐरे और फंक्शन" पर चर्चा करेंगे, जिसमें ऐरे, ऐरे विधियाँ, ऐरे और लूप्स, फंक्शन, फंक्शन रिटर्न, फंक्शन और फंक्शन पैरामीटर एवं आर्ग्युमेंट के मध्य अंतर शामिल हैं।

इसके बाद हम "स्ट्रिंग प्रबंधन" पर चर्चा करेंगे, जिसमें स्ट्रिंग मैनीपुलेशन, दिनांक और समय तथा फॉर्म्स की अवधारणाएँ समझाई जाएँगी। इसके अतिरिक्त, "JavaScript में चित्र और भू-स्थान (Geolocation) का प्रबंधन" पर भी प्रकाश डाला जाएगा, जिसमें छवियों में परिवर्तन, JavaScript से नई छवियाँ बनाना, इमेज रोलओवर प्रभाव और टाइमर एवं चित्रों का उपयोग बताया जाएगा।

अंत में, हम "HTML5 कैनवस" पर चर्चा करेंगे, जिसमें JavaScript में HTML5 कैनवस टैग, आकृतियाँ बनाना, ग्रेडिएंट्स और कैनवस टैग के साथ छवियों का उपयोग सम्मिलित है। यह मॉड्यूल आपको JavaScript में आवश्यक ज्ञान और कौशल प्रदान करती है।

सत्र 13

JavaScript

अंकित एक जिज्ञासु बच्चा है, जब उसे "JavaScript" के बारे में पता चला, तो उसके सामने अनेक संभावनाएं हैं, जो वेब पृष्ठों के लिए एक जादू की छड़ी जैसी थी, जिससे वेब पेजों को रोचक बनाया जा सकता था, जैसे खेल और संवादात्मक प्रश्नोत्तरी। JavaScript की सहायता से वह पात्रों को एनिमेट कर सकता था, वेब पृष्ठों को क्लिक पर प्रतिक्रिया देने योग्य बना सकता था, स्कोर गिन सकता था और एक क्लिक में चीजें बदल सकता था। अंकित ने JavaScript का उपयोग करके वेब गेम और संवादात्मक कहानियाँ बनाई और वह स्वयं को एक डिजिटल जादूगर जैसा अनुभव करने लगा। इससे इंटरनेट एक मनोरंजक स्थान बन गया, जहाँ वह अपने मित्रों के साथ आनंद ले सकता था और अपनी बनाई हुई रोचक सामग्री साझा कर सकता था, जैसा कि चित्र 13.1 में दर्शाया गया है।

चित्र 13.1— JavaScript का उपयोग करता हुआ अंकित

इस अध्याय में, आप JavaScript का अवलोकन, इसकी विशेषताएँ, JavaScript और Java में अंतर, “Hello, world” संदेश प्रदर्शित करना और ब्राउज़र ऑब्जेक्ट मॉडल के बारे में समझेंगे।

13.1 JavaScript का अवलोकन

JavaScript एक अत्यंत बहुपरकीय और गतिशील प्रोग्रामिंग भाषा है जो वेब विकास में महत्त्वपूर्ण भूमिका निभाती है। इसका प्रमुख कार्य वेब पृष्ठों में संवादात्मकता (interactivity) को सक्षम बनाना, वेब की सामग्री में परिवर्तन करना और वेब अनुप्रयोगों (applications) में उत्तरदायी प्रयोक्ता इंटरफ़ेस (responsive UI) तैयार करना है।

> **क्या आप जानते हैं?**
> JavaScript को 1995 में Brendan Eich ने Netscape Communications Corporation में बनाया था। इसका प्रारंभिक नाम “LiveScript” था।

क्लाइंट-साइड भाषा

JavaScript एक प्रमुख क्लाइंट-साइड स्क्रिप्टिंग भाषा के रूप में कार्य करती है, जो प्रयोक्ता के वेब ब्राउज़र में निष्पादित होती है। यह एक गतिशील भाषा है जिसे HTML दस्तावेज़ों के अंदर प्रत्यक्ष रूप से शामिल किया जा सकता है या बाह्य रूप से लिंक किया जा सकता है। यह डेवलपर्स को वेब अनुप्रयोगों में संवादात्मकता जोड़ने, पृष्ठ की सामग्री को गतिशील रूप से परिवर्तित करने और उत्तरदायी UI विकसित करने की सुविधा देती है।

गतिशील और कमजोर टाइपिंग वाली

JavaScript में डेटा प्रकार को स्पष्ट रूप से घोषित करने की आवश्यकता नहीं होती, अर्थात यह **डायनामिकली टाइप्ड** भाषा है। यह **वीकली टाइप्ड** भी होती है, जिससे यह लचीलापन प्रदान करती है कि विभिन्न प्रकारों के बीच परिवर्तन बिना त्रुटि के हो सके।

संवादात्मकता

JavaScript वेब पर संवादात्मकता का मूल आधार है। यह फ़ॉर्म सत्यापन, एनीमेशन और प्रयोक्ता इंटरैक्शन की प्रतिक्रिया जैसी सुविधाएँ प्रदान करती है, जिससे वेबसाइट अधिक प्रयोक्ता मित्रवत और रोचक बनती है।

इवेंट-चालित

JavaScript एक इवेंट-ड्रिवन भाषा है, जिसका अर्थ है कि यह प्रयोक्ता द्वारा क्लिक, कीबोर्ड इनपुट या पृष्ठ लोड जैसी घटनाओं पर प्रतिक्रिया दे सकती है। इन घटनाओं को नियंत्रित करने के लिए *इवेंट लिसनर* का उपयोग किया जाता है।

DOM (Document Object Model) प्रबंधन

DOM वेब दस्तावेज़ों के प्रोग्रामन के लिए एक इंटरफ़ेस है। JavaScript DOM को संशोधित कर सकती है, जिससे वेब पृष्ठों की सामग्री और संरचना में गतिशील परिवर्तन संभव हो पाते हैं। यह संवादात्मक और अनुकूलनीय वेब अनुप्रयोगों के विकास में एक मौलिक घटक है।

वेरिएबल और डेटा प्रकार

JavaScript में अनेक डेटा प्रकारों का समर्थन होता है—जैसे संख्या (numbers), स्ट्रिंग (strings), बूलियन (booleans), ऑब्जेक्ट (objects), ऐरे (arrays) और फंक्शन (functions)। वेरिएबल्स को var, let या const की सहायता से परिभाषित किया जा सकता है।

फंक्शन

JavaScript में फंक्शन एक मूलभूत घटक होते हैं। आप फंक्शन को परिभाषित करके कोड को संक्षेप में संगठित कर सकते हैं और उसे बार-बार उपयोग कर सकते हैं। *एनॉनिमस फंक्शन्स* (anonymous functions) या *लैम्ब्डा फंक्शन्स* का भी उपयोग प्रायः किया जाता है।

नियंत्रण प्रवाह (Control Flow)

JavaScript विभिन्न नियंत्रण प्रवाह संरचनाएँ प्रदान करता है, जैसे—if...else, switch, for, while लूप आदि। ये संरचनाएँ प्रोग्राम की कार्यप्रणाली और निष्पादन को नियंत्रित करती हैं, जिससे निर्णय लेना, कार्यों को दोहराना और तर्क संचालन करना संभव होता है।

असमकालिक प्रोग्रामन (Asynchronous Programming)

JavaScript में असमकालिक प्रोग्रामिंग एक ऐसी विधि है जिसमें कोड इस प्रकार लिखा जाता है कि किसी सर्वर या अन्य संसाधन से उत्तर की प्रतीक्षा करते हुए अन्य कार्य भी किए जा सकें। यह वेब पृष्ठों को अधिक उत्तरदायी बनाने या दीर्घकालिक कार्यों को दक्षतापूर्वक निष्पादित करने के लिए उपयोगी होता है।

JavaScript में असमकालिक प्रोग्रामिंग को लागू करने के लिए कई विधियाँ हैं, लेकिन सबसे सामान्य तरीका है *कॉलबैक (callback)* का उपयोग।

क्या आप जानते हैं?

कॉलबैक एक ऐसी फंक्शन होती है जो किसी असमकालिक कार्य के पूर्ण होने पर क्रियान्वित होती है।

लाइब्रेरीज़

JavaScript में लाइब्रेरी ऐसे पूर्व-लिखित कोड या फंक्शन का संग्रह होती हैं जिन्हें आप आसानी से अपने कोड में पुनः उपयोग कर सकते हैं। ये लाइब्रेरी DOM के साथ कार्य करना, एनीमेशन संभालना, HTTP अनुरोध करना आदि जैसे सामान्य कार्यों को सरल बनाती हैं। कुछ प्रमुख JavaScript लाइब्रेरी हैं—**jQuery**, **React**, **Angular** आदि, जो भिन्न-भिन्न उद्देश्यों के लिए प्रयुक्त होती हैं।

क्या आप जानते हैं?

लाइब्रेरीज़ वर्गों (classes) और विधियों (methods) का संग्रह होती हैं जो पुनः उपयोग के लिए स्टोर की जाती हैं।

फ्रेमवर्क

JavaScript फ्रेमवर्क पूर्वनिर्धारित उपकरण और दिशा-निर्देश होते हैं जो वेब विकास को सुगम बनाते हैं। ये जटिल कार्यों को प्रबंधित करने, कोड पुनः उपयोग को बढ़ावा देने और विकास को संरचित बनाने में सहायता करते हैं। जैसे—**React**, **Angular**, **Vue.js** आदि। ये विकास को सरल बनाते हैं और अनुकूलन, संगतता तथा समुदाय समर्थन भी प्रदान करते हैं। फ्रेमवर्क का चयन परियोजना की आवश्यकताओं और टीम की विशेषज्ञता के आधार पर किया जाना चाहिए।

क्या आप जानते हैं?

फ्रेमवर्क एक संपूर्ण अनुप्रयोग विकसित करने हेतु आवश्यक सभी घटकों को प्रदान करने का प्रयास करता है।

सर्वर-साइड उपयोग

JavaScript का सर्वर-साइड उपयोग तब होता है जब कोड वेब ब्राउज़र के बजाय सर्वर पर चलाया जाता है। इसके लिए **Node.js** एक प्रमुख प्लेटफॉर्म है, जिसका उपयोग API निर्माण, डेटाबेस इंटरैक्शन, फ़ाइल प्रबंधन, प्रमाणीकरण आदि में किया जाता है। सर्वर-साइड JavaScript पूर्ण स्टैक एप्लीकेशन निर्माण के लिए आवश्यक और बहुपरकीय है।

क्रॉस-ब्राउज़र संगतता

JavaScript की लाइब्रेरीज़ और फ्रेमवर्क यह सुनिश्चित करने में सहायता करते हैं कि कोड विभिन्न ब्राउज़रों में एक समान रूप से कार्य करे और ब्राउज़र-विशिष्ट समस्याओं को दूर किया जा सके।

13.1 JavaScript की विशेषताएँ

- अधिकांश वेब ब्राउज़र JavaScript को चला सकते हैं क्योंकि उनमें अंतर्निहित टूल होते हैं।
- JavaScript का सिंटैक्स और संरचना C प्रोग्रामिंग भाषा के समान है।
- यह एक वस्तु-उन्मुख (object-oriented) भाषा है जो क्लास के बजाय प्रोटोटाइप आधारित इनहेरिटेंस पर कार्य करती है।
- यह एक हल्की भाषा है जिसे पठन और प्रोसेसिंग के दौरान निष्पादित किया जाता है।
- JavaScript एक हल्की और केस-सेंसिटिव भाषा है।
- यह Windows, macOS और अन्य कई ऑपरेटिंग सिस्टम पर कार्य करती है।
- यह प्रयोक्ताओं को वेब ब्राउज़र पर उल्लेखनीय नियंत्रण प्रदान करती है।

क्या आप जानते हैं?

JavaScript एक हल्की, वस्तुनिष्ठ उन्मुख प्रोग्रामिंग भाषा है जिसका उपयोग विभिन्न वेबसाइट्स अपनी स्क्रिप्टिंग के लिए करती हैं।

13.2 जावास्क्रिप्ट और जावा के बीच अंतर

जावास्क्रिप्ट (JavaScript)	**जावा (Java)**
वेब विकास के लिए प्रयुक्त एक स्क्रिप्टिंग भाषा।	उच्च स्तरीय, सामान्य प्रयोजन की प्रोग्रामिंग भाषा।
हल्की और शिथिल रूप से टाइप की जाती है।	कठोर रूप से टाइप की जाती है और इसका वाक्य विन्यास अधिक जटिल होता है।
मुख्यतः वेब ब्राउज़र में चलती है।	जावा वर्चुअल मशीन (JVM) पर चलती है और विविध अनुप्रयोगों में प्रयुक्त होती है।
प्रोटोटाइप-आधारित वस्तु-उन्मुख (object-oriented) होती है।	क्लास-आधारित वस्तु-उन्मुख (object-oriented) होती है।
रनटाइम पर इंटरप्रेट की जाती है।	निष्पादन से पहले बाइटकोड में संकलित की जाती है।

तालिका 13.1— जावास्क्रिप्ट और जावा के बीच अंतर

अभ्यास 13.1

• जावास्क्रिप्ट की विशेषताओं को सूचीबद्ध करें।

• जावास्क्रिप्ट और जावा के बीच अंतर सूचीबद्ध करें।

13.3 जावास्क्रिप्ट टेम्पलेट से एक वेब पृष्ठ बनाना

13.3.1 HTML संरचना

• <!DOCTYPE html> घोषणा HTML दस्तावेज़ के प्रकार और संस्करण (HTML5) को निर्दिष्ट करती है।

• HTML दस्तावेज़ का मूल तत्व <html> है, जो दस्तावेज़ की भाषा को परिभाषित करता है (यहाँ अंग्रेज़ी)।

• <head> अनुभाग में दस्तावेज़ से संबंधित महत्त्वपूर्ण मेटाडेटा होता है, जैसे कैरेक्टर एन्कोडिंग और व्यूपोर्ट विन्यास।

• <head> अनुभाग के अंदर <style> टैग होता है, जिसमें इनलाइन CSS शैलियाँ (styles) होती हैं, जो वेब पृष्ठ पर विभिन्न तत्वों की डिज़ाइनिंग के लिए प्रयुक्त होती हैं।

13.3.2 CSS शैलियाँ

- <style> अनुभाग में परिभाषित CSS शैलियाँ वेब पृष्ठ के विभिन्न तत्वों पर लागू होती हैं।
- body शैली डिफ़ॉल्ट मार्जिन और पैडिंग को हटाती है, फ़ॉन्ट को Arial सेट करती है, और एक पृष्ठभूमि छवि (background.jpg) को पूरी स्क्रीन पर कवर करने के लिए background-size: cover लागू करती है।
- .container शैली एक फिक्स्ड स्थिति वाली नेविगेशन पट्टी तैयार करती है जिसमें अर्ध-पारदर्शी सफेद पृष्ठभूमि और पैडिंग होती है।
- #icon शैली आइकन के लिए कर्सर को पॉइंटर पर सेट करती है, जिससे वह क्लिक योग्य प्रतीत होता है।

13.3.3 पृष्ठ की सामग्री

<body> टैग के अंदर एक टैग होता है जिसकी कक्षा container है। यह पृष्ठ के शीर्ष पर स्थित नेविगेशन पट्टी का कार्य करता है। इसके अंदर <h1> टैग में "MY SOUNDS LIST" लिखा होता है और एक <img> टैग होता है जिसकी ID "icon" है। इस छवि (image) का नाम "pause.jpg" होता है, जो मूलतः एक **रोकने (pause)** वाले बटन को दर्शाती है।

13.3.4 ऑडियो प्लेयर

- एक <audio> टैग होता है जिसकी ID "mysound" है। इसके अंदर दो <source> टैग होते हैं जो विभिन्न स्वरूपों (formats) में ऑडियो फ़ाइलें प्रदान करते हैं ताकि संगतता (compatibility) बनी रहे। src गुण "XYZ.mp3" नामक एक MP3 फ़ाइल की ओर इशारा करता है और दूसरा "audio.ogg" नामक Ogg Vorbis फ़ाइल का स्रोत होता है (यदि आपके पास वास्तविक पथ हो तो "audio.ogg" को उससे बदलें)।
- यदि ब्राउज़र HTML5 ऑडियो का समर्थन नहीं करता है, तो "Your browser does not support the audio element" संदेश प्रदर्शित होगा।

13.3.5 जावास्क्रिप्ट

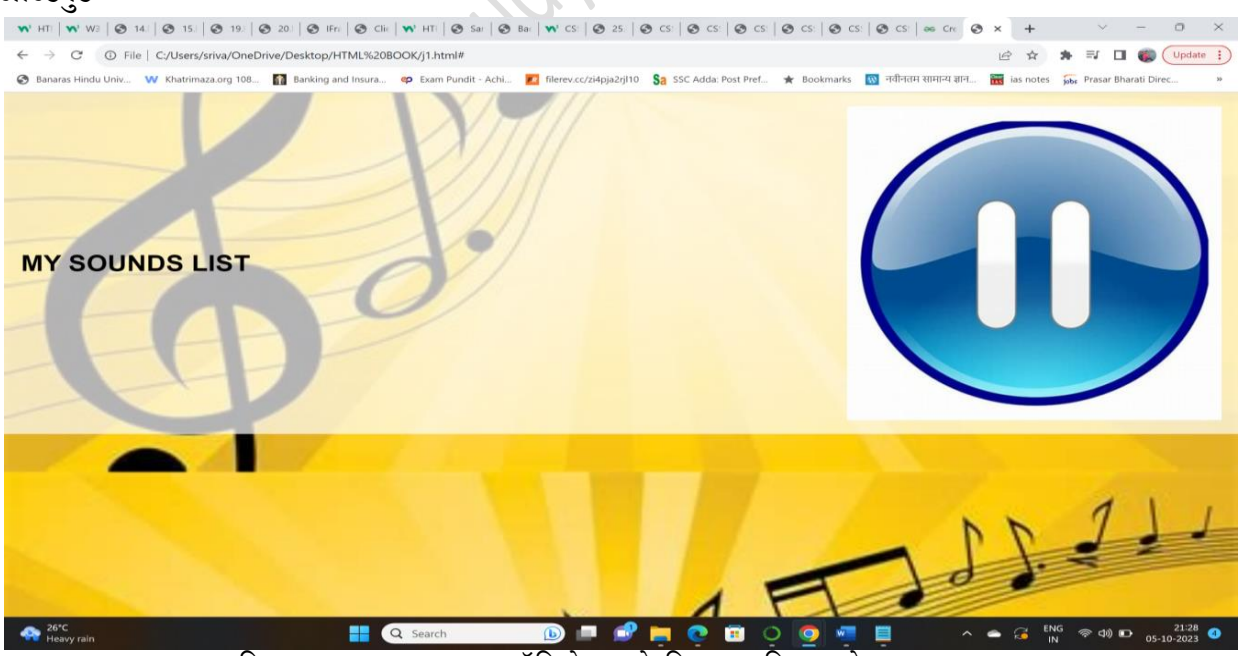
- दस्तावेज़ के अंत में दिया गया जावास्क्रिप्ट कोड ऑडियो प्लेबैक को नियंत्रित करता है और **प्ले/पॉज़ आइकन** को बदलता है।
- यह document.getElementById विधि का उपयोग करके "mysound" ऑडियो तत्व और "icon" इमेज तत्व को संदर्भित करता है।
- जब "icon" पर क्लिक किया जाता है, तो onclick इवेंट हैंडलर यह जाँचता है कि ऑडियो रुका हुआ है या नहीं। यदि रुका हुआ है, तो ऑडियो चलना शुरू हो जाता है और आइकन "play.jpg" में बदल जाता है। यदि ऑडियो चल रहा हो, तो वह रुक जाता है और आइकन "pause.jpg" में बदल जाता है।

कोड —

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
      body {
         margin: 0;
         padding: 0;
         font-family: Arial, sans-serif;
         background-image: url('background.jpg'); /* Replace 'background.jpg' with your image file's name
*/
         background-size: cover;
         background-position: center;
      }
      .container {
         display: flex;
         justify-content: space-between;
         align-items: center;
         background-color: rgba(255, 255, 255, 0.7);
         padding: 20px;
         position: fixed;
         top: 0;
         left: 0;
         right: 0;
      }
      #icon {
         cursor: pointer;
      }
</style>
</head>
<body>
<div class="container">
<h1>MY SOUNDS LIST</h1>
<img src="pause.jpg" id="icon">
</div>

<audio id="mysound">
<source src="XYZ.mp3" type="audio/mpeg">
```

```
<!-- Provide alternate sources for compatibility -->
<source src="audio.ogg" type="audio/ogg">
      Your browser does not support the audio element.
</audio>

<script>
      const mysound = document.getElementById("mysound");
      const icon = document.getElementById("icon");

icon.onclick = function () {
         if (mysound.paused) {
mysoundplay();
            icon.src = "play.jpg";
         } else {
mysoundpause();
            icon.src = "pause.jpg";
         }
      };
</script>
</body>
</html>
```

आउटपुट —



चित्र 13.2— "mysound" ऑडियो तत्व के लिए जावास्क्रिप्ट कोड का आउटपुट

13.4 संदेश "Hello, world" को प्रदर्शित करना

हम अपना प्रारंभिक क्लाइंट-साइड JavaScript स्क्रिप्ट बना सकते हैं ताकि "Hello, world" संदेश को प्रदर्शित किया जा सके। प्रारंभ करने के लिए, एक नई फ़ाइल बनाएँ और उसमें निम्नलिखित कोड दर्ज करें। कृपया ध्यान दें कि दिए गए लाइन नंबर केवल समझाने के लिए हैं, इन्हें शामिल न करें। निम्नलिखित बातों को ध्यान में रखें—

JavaScript में **अक्षर केस (case)** का महत्व होता है। "A rose" और "A ROSE" तथा "A Rose" तीनों अलग-अलग माने जाते हैं।

"अतिरिक्त" श्वेत स्थान जैसे स्पेस, टैब और नई पंक्ति को JavaScript अनदेखा करता है। अर्थात्, कई रिक्त स्थानों को एक रिक्त स्थान (space) के रूप में ही समझा जाता है। आप इन्हें कोड की पठनीयता बढ़ाने के लिए प्रयोग कर सकते हैं, जैसा कि **चित्र 13.3** में दिखाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript Example: Functions alert() and document.write()</title>
    <script>
        alert("Hello World!")
    </script>
</head>
<body>
    <h1>My First JavaScript Says:</h1>
    <script>
        document.write("<h2><em>Hello World!, again</em></h2>")
        document.write("<p>This document was last modified on " + document.lastModified + "</p>")
    </script>
</body>
</html>
```

चित्र 13.3 — "Hello, world" संदेश को प्रदर्शित करने के लिए JavaScript स्क्रिप्ट

नीचे कुछ महत्त्वपूर्ण बिंदुओं का सारांश और स्पष्टीकरण दिया गया है—

13.4.1 HTML में JavaScript को जोड़ना

आम तौर पर JavaScript कोड HTML दस्तावेज़ में <script> टैग के माध्यम से जोड़ा जाता है। JavaScript कोड <script> टैग के ओपनिंग और समापन टैग के बीच लिखा जाता है।

HTML4/XHTML में <script> टैग में type="text/javascript" गुण देना वैलिड है, हालाँकि वर्तमान HTML5 में यह अनिवार्य नहीं है।

13.4.2 JavaScript का स्थान

JavaScript कोड HTML दस्तावेज़ के <head> अनुभाग में (जिसे "हेडर स्क्रिप्ट" कहा जाता है) या <body> अनुभाग में (जिसे "बॉडी स्क्रिप्ट" कहा जाता है) लिखा जा सकता है।

यदि एक ही HTML दस्तावेज़ में कई JavaScript स्क्रिप्ट्स शामिल करनी हों, तो आप कई <script> टैग का प्रयोग कर सकते हैं।

13.4.3 कथनों का समापन

JavaScript में प्रत्येक कथन (statement) का समापन **सेमिकोलन (;)** से किया जाता है, जैसे कि Java, C, C++ और C# जैसी भाषाओं में होता है।

13.4.4 alert() फंक्शन

alert(str) फ़ंक्शन एक पॉप-अप संवाद बॉक्स (dialog box) प्रदर्शित करता है, जिसमें str नामक संदेश और "OK" बटन होता है।

JavaScript में स्ट्रिंग्स को डबल कोट्स ("hello") या सिंगल कोट्स ('hello') में लिखा जा सकता है।

13.4.5 document ऑब्जेक्ट

JavaScript document ऑब्जेक्ट का उपयोग वर्तमान वेब पृष्ठ को संदर्भित करने के लिए करता है।
वर्तमान दस्तावेज़ की अंतिम बार संशोधित तिथि document.lastModified गुण (property) में संग्रहित होती है।
document.write(str) फ़ंक्शन का प्रयोग किसी निर्दिष्ट स्ट्रिंग (str) को वर्तमान HTML दस्तावेज़ में डायनामिक रूप से शामिल करने हेतु किया जाता है।

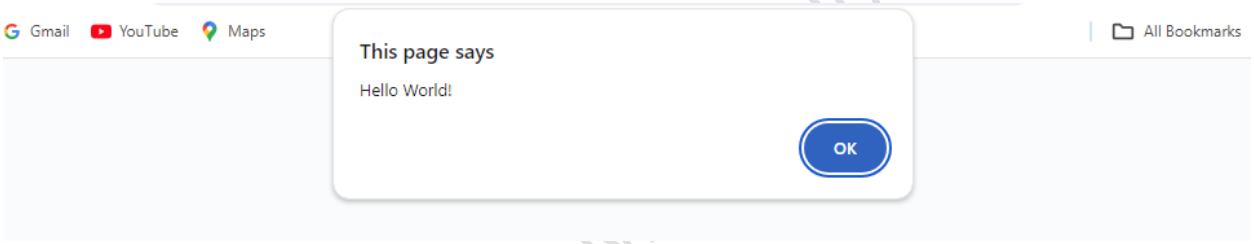
13.4.6 स्ट्रिंग संयोजन (Concatenation)

JavaScript में + ऑपरेटर का उपयोग दो स्ट्रिंग्स को जोड़ने (concatenate) के लिए किया जाता है, ठीक वैसे ही जैसे Java जैसी भाषाओं में किया जाता है।

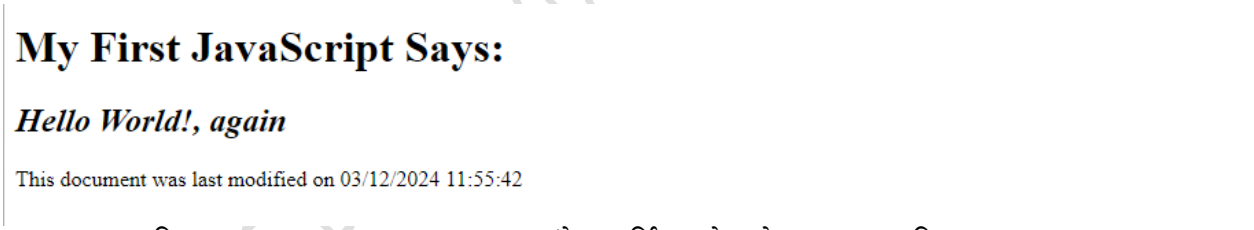
13.4.7 सामान्यत: प्रयुक्त फ़ंक्शन

alert() और document.write() JavaScript द्वारा प्रदत्त सामान्य अंतर्निर्मित (built-in) फ़ंक्शन हैं। alert() का उपयोग पॉप-अप संदेश प्रदर्शित करने हेतु होता है, जबकि document.write() का उपयोग HTML दस्तावेज़ में सामग्री जोड़ने के लिए किया जाता है।

आउटपुट —



चित्र 13.4 — "Hello, world" संदेश प्रदर्शित करने वाले JavaScript स्क्रिप्ट का आउटपुट



चित्र 13.5 — "Hello, world again" संदेश प्रदर्शित करने वाले JavaScript स्क्रिप्ट का आउटपुट

13.5 JavaScript में त्रुटि सुधार (Debugging)

वेब पृष्ठ में JavaScript कोड को डीबग (debug) करने का अर्थ है — कोड में समस्याओं की पहचान कर उन्हें ठीक करना, ताकि अपेक्षित आउटपुट प्राप्त किया जा सके। नीचे एक चरण-दर-चरण प्रदर्शनी दी गई है, जिसमें JavaScript टेम्पलेट से बनाए गए एक वेब पृष्ठ को डीबग करके आउटपुट प्राप्त किया गया है।
मान लीजिए हमारे पास एक सरल HTML संरचना वाला वेब पृष्ठ है, जिसमें JavaScript को इस प्रकार जोड़ा गया है कि बटन क्लिक करने पर एक संदेश प्रदर्शित हो। हालाँकि, कोड में एक त्रुटि (bug) है, जिसे हमें ठीक करना है।

चरण 1— HTML संरचना तैयार करें — जैसा कि **चित्र 13.6** में दिखाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Debugging Example</title>
</head>
<body>
    <h1>Debugging JavaScript</h1>
    <button id="showMessageButton">Show Message</button>
    <div id="messageDisplay"></div>

    <script src="script.js"></script>
</body>
</html>
```

चित्र 13.6 — HTML संरचना कोड बनाना

इस HTML कोड में हमारे पास एक बटन है जिसका ID है showMessageButton और एक रिक्त div है जिसका ID है messageDisplay। JavaScript कोड एक पृथक फ़ाइल script.js में सम्मिलित किया गया है, जिसे हम अगले चरण में बनाएँगे।

चरण 2— JavaScript फ़ाइल बनाएँ

एक नई फ़ाइल script.js नाम से बनाएँ और निम्न JavaScript कोड दर्ज करें—

javascript

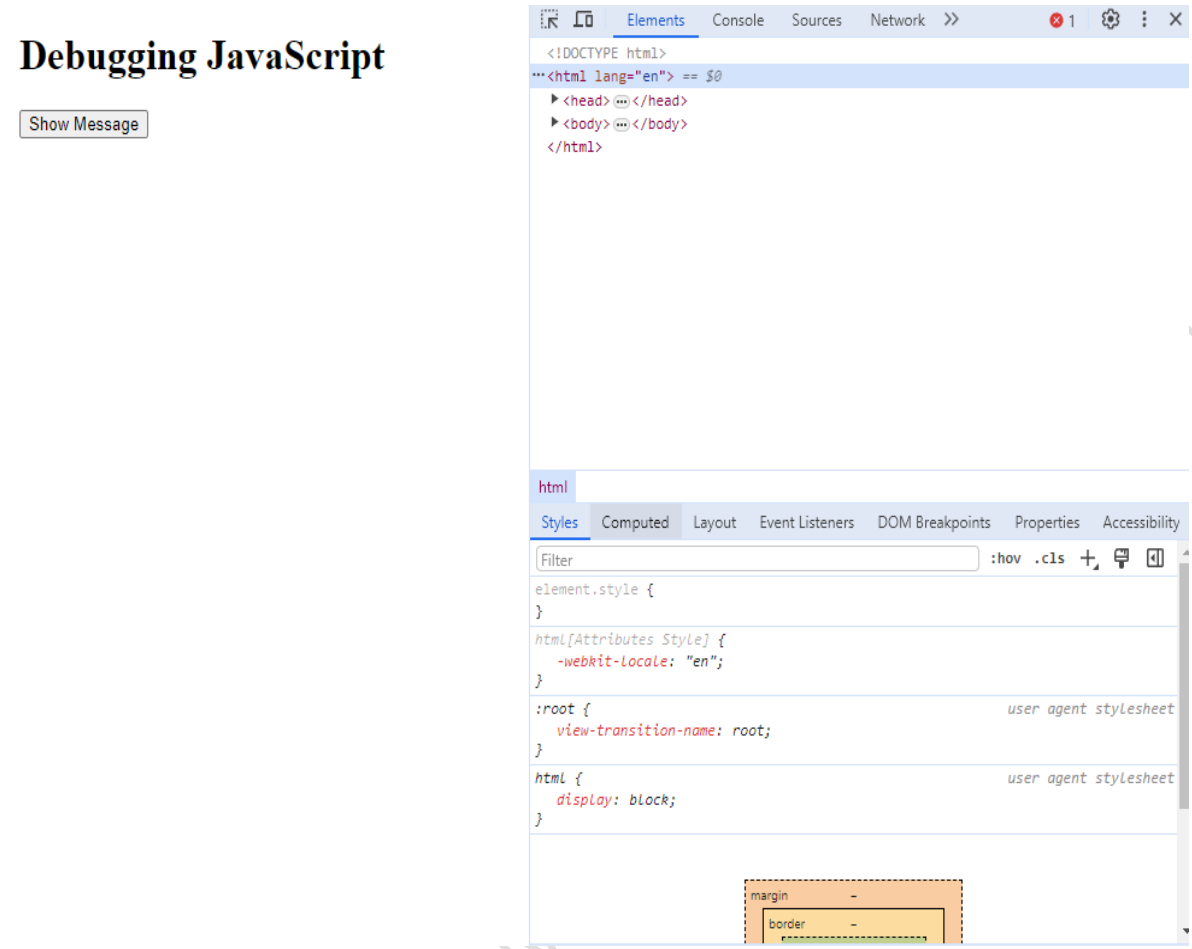
```
// बटन क्लिक करने पर संदेश प्रदर्शित करने का प्रयास
document.getElementById("showMessageButton").addEventListener("click", function() {
    document.getElementById("messageDisplay").textContent = "Hello, Debugging!";
});
```

चरण 3— त्रुटि सुधार प्रक्रिया (Debugging Process)

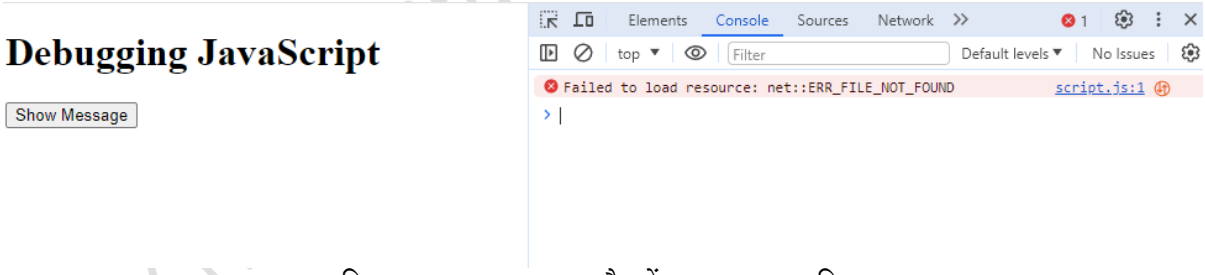
अब, मान लीजिए JavaScript कोड में कोई समस्या है, और बटन क्लिक करने पर कुछ नहीं होता। इस समस्या को डीबग करने के लिए निम्न चरणों का पालन करें—

i. ब्राउज़र कंसोल जाँचें—

- अपने वेब पृष्ठ को किसी वेब ब्राउज़र (जैसे— Google Chrome, Firefox, या Microsoft Edge) में खोलें।
- पृष्ठ पर राइट-क्लिक करें और “Inspect” विकल्प चुनें या F12 दबाएँ — इससे **डिवेलपर टूल्स** खुलेंगे, जैसा कि **चित्र 13.7** में दिखाया गया है।
- "Console" टैब पर जाएँ — यहाँ JavaScript त्रुटियाँ और संदेश प्रदर्शित होते हैं, जैसा कि **चित्र 13.8** में दर्शाया गया है।



चित्र 13.7 — "Elements" टैब में Debugging प्रक्रिया



चित्र 13.8 — "Console" टैब में Debugging प्रक्रिया

ii. त्रुटियाँ खोजें—

- कंसोल में किसी भी त्रुटि संदेश की जाँच करें। यहाँ JavaScript कोड में समस्या को बताने वाली कोई त्रुटि हो सकती है।

iii. समस्या ठीक करें—

- इस उदाहरण में समस्या यह है कि script.js फ़ाइल में एक वर्तनी त्रुटि (typo) है। हमने बटन पर इवेंट लिसनर जोड़ने के लिए getElementById का प्रयोग किया, लेकिन गलती से "B" को बड़े अक्षर में लिख दिया — जबकि सही वाक्यांश getElementById (छोटा 'b') होना चाहिए।

सुधारा गया कोड—

javascript

```
document.getElementById("showMessageButton").addEventListener("click", function() {
```

```
    document.getElementById("messageDisplay").textContent = "Hello, Debugging!";
});
```

iv. पुनः परीक्षण करें—

- सुधारा गया script.js फ़ाइल सहेजें।
- वेब पृष्ठ को पुनः लोड करें।
- "Show Message" बटन पर क्लिक करें।

चरण 4— आउटपुट उत्पन्न करना

त्रुटि सुधार और कोड ठीक करने के बाद, अब वेब पृष्ठ अपेक्षित रूप से कार्य करेगा। जब आप "Show Message" बटन पर क्लिक करेंगे, तो messageDisplay नामक div में "Hello, Debugging!" संदेश प्रदर्शित होगा, जैसा कि **चित्र 13.9** में दिखाया गया है।

चित्र 13.9 — "Hello, Debugging!" आउटपुट प्रदर्शित करने वाली Debugging प्रक्रिया

यह प्रस्तुति वेब पृष्ठ में JavaScript कोड को डीबग कर वांछित आउटपुट प्राप्त करने की पूरी प्रक्रिया को दर्शाती है। डीबगिंग में समस्या की पहचान करना, ब्राउज़र कंसोल में त्रुटियों को देखना, आवश्यक सुधार करना और कोड को पुनः परीक्षण करना सम्मिलित होता है।

13.6 ब्राउज़र ऑब्जेक्ट मॉडल

ब्राउज़र ऑब्जेक्ट मॉडल (Browser Object Model - BOM) उन ऑब्जेक्ट्स (वस्तुओं) का एक समूह है, जो वेब ब्राउज़र द्वारा प्रदान किए जाते हैं ताकि ब्राउज़र स्वयं, विंडो तथा अन्य ब्राउज़र-विशिष्ट सुविधाओं के साथ संवाद किया जा सके। यह डाक्युमेंट ऑब्जेक्ट मॉडल (Document Object Model - DOM) से अलग घटक है, जो वेब दस्तावेज़ों (HTML, XML आदि) की संरचना और सामग्री से संबंधित होता है। BOM जावास्क्रिप्ट (JavaScript) को वेब पृष्ठ की सामग्री में परिवर्तन करने के अतिरिक्त, ब्राउज़र के साथ संवाद स्थापित कर अन्य कार्य करने की अनुमति देता है।
नीचे कुछ प्रमुख घटकों और ऑब्जेक्ट्स का विवरण दिया गया है।

13.6.1 विंडो ऑब्जेक्ट (Window Object)—

- ब्राउज़र ऑब्जेक्ट मॉडल (BOM) के अंतर्गत, सबसे व्यापक ऑब्जेक्ट "window" होता है, जो ब्राउज़र विंडो या टैब का प्रतिनिधित्व करता है। यह ब्राउज़र के साथ संवाद स्थापित करने का मुख्य इंटरफेस होता है।
- जावास्क्रिप्ट में, आप window ऑब्जेक्ट से संबंधित विशेषताओं और कार्यों (functions) को प्राप्त कर सकते हैं और उन्हें परिवर्तित भी कर सकते हैं।
- जावास्क्रिप्ट में window ऑब्जेक्ट विभिन्न प्रकार की विधियाँ (methods) प्रदान करता है, जो ब्राउज़र विंडो या टैब के साथ संवाद और नियंत्रण की अनुमति देती हैं। इन विधियों का उपयोग नई विंडो खोलने, अलर्ट और प्रॉम्प्ट प्रदर्शित करने, दस्तावेज़ के व्यवहार को नियंत्रित करने आदि के लिए किया जा सकता है। नीचे कुछ सामान्य window विधियाँ दी गई हैं—

```
javascript
// window ऑब्जेक्ट तक पहुँच
window.location.href = "https://www.example.com"; // URL को बदलना
window.alert("Hello, BOM!"); // अलर्ट प्रदर्शित करना
```

alert(message)—

यह विधि एक पॉप-अप संवाद बॉक्स (डायलॉग बॉक्स) प्रदर्शित करती है, जिसमें दिया गया संदेश और एक "OK" बटन होता है।

javascript

```
window.alert("This is an alert!");
```

confirm(message)—

यह एक पॉप-अप डायलॉग बॉक्स प्रदर्शित करता है, जिसमें दिया गया संदेश, "OK" और "Cancel" बटन होते हैं। यदि प्रयोक्ता "OK" क्लिक करता है, तो true लौटता है; यदि "Cancel" क्लिक करता है, तो false लौटता है।

javascript

```
var result = window.confirm("Do you want to continue?");
if (result) {
   // प्रयोक्ता ने OK क्लिक किया
} else {
   // प्रयोक्ता ने Cancel क्लिक किया
}
```

prompt(message, defaultText)—

यह एक पॉप-अप डायलॉग बॉक्स प्रदर्शित करता है, जिसमें दिया गया संदेश, टेक्स्ट दर्ज करने के लिए एक इनपुट फ़ील्ड और "OK" तथा "Cancel" बटन होते हैं। यदि "OK" पर क्लिक किया जाता है, तो दर्ज किया गया टेक्स्ट string के रूप में लौटता है; यदि "Cancel" पर क्लिक किया जाता है, तो null लौटता है।

javascript

```
var name = window.prompt("Enter your name:", "John Doe");
if (name !== null) {
   // प्रयोक्ता ने नाम दर्ज किया
} else {
   // प्रयोक्ता ने Cancel क्लिक किया
}
```

open(url, target, features)—

यह एक नई ब्राउज़र विंडो या टैब खोलता है, जिसमें दिए गए URL को लोड किया जाता है। इसमें target (जैसे "_blank" नई टैब के लिए) और अतिरिक्त गुण (जैसे विंडो का आकार और व्यवहार) निर्दिष्ट किए जा सकते हैं।

javascript

```
window.open("https://www.example.com","_blank", "width=500,height=300");
```

window.open विधि तीन पैरामीटर लेती है—

- **url (string)—** वह पृष्ठ जिसे आप नई विंडो में खोलना चाहते हैं।
- **target (string)—** यह निर्दिष्ट करता है कि नई विंडो कहाँ खोली जाए। सामान्य मान निम्नलिखित हैं—
    - "_blank": नई टैब या विंडो में खोलता है (डिफ़ॉल्ट)
    - "_self": वर्तमान टैब या विंडो में खोलता है
    - "_parent": पेरेंट विंडो या फ्रेम में खोलता है
    - "_top": टॉप-लेवल ब्राउज़िंग संदर्भ में खोलता है
    - कस्टम विंडो नाम— निर्दिष्ट नाम वाली विंडो में खोलता है

- **features (string)**— विंडो के गुणों (जैसे आकार, स्थिति) की कॉमा-सेparated सूची

close()—
यह वर्तमान ब्राउज़र विंडो या टैब को बंद करता है, यदि उसे जावास्क्रिप्ट द्वारा खोला गया था। अधिकांश आधुनिक ब्राउज़र सुरक्षा कारणों से उन विंडो को बंद करने की अनुमति नहीं देते जिन्हें स्क्रिप्ट द्वारा नहीं खोला गया हो।
javascript
window.close();

setTimeout(function, delay)—
यह एक निर्दिष्ट देरी (मिलीसेकंड में) के बाद किसी फ़ंक्शन को निष्पादित करने के लिए अनुसूचित करता है। यह एक टाइमर ID लौटाता है जिसे clearTimeout() द्वारा रद्द किया जा सकता है।
javascript
var timerId = setTimeout(function() {
  console.log("Delayed function executed!");
}, 2000); // 2 सेकंड बाद निष्पादन

clearTimeout(timerId)—
यह setTimeout() द्वारा सेट किए गए टाइमआउट को रद्द करता है।
javascript
clearTimeout(timerId);

setInterval(function, delay)—
यह निर्दिष्ट देरी (मिलीसेकंड में) के अंतराल पर किसी फ़ंक्शन को बार-बार निष्पादित करता है। यह एक इंटरवल ID लौटाता है जिसे clearInterval() द्वारा रद्द किया जा सकता है।
javascript
var intervalId = setInterval(function() {
  console.log("Interval function executed!");
}, 1000); // हर 1 सेकंड में निष्पादन

clearInterval(intervalId)—
यह setInterval() द्वारा सेट किए गए इंटरवल को रद्द करता है।
javascript
clearInterval(intervalId); उपरोक्त विधियाँ जावास्क्रिप्ट में window ऑब्जेक्ट द्वारा प्रदान की गई कुछ सामान्य विधियाँ हैं, जो ब्राउज़र विंडो के व्यवहार को नियंत्रित करने और प्रयोक्ता से पॉप-अप डायलॉग के माध्यम से संवाद स्थापित करने में सहायता करती हैं।

नेविगेटर ऑब्जेक्ट (Navigator Object)—
navigator ऑब्जेक्ट प्रयोक्ता के ब्राउज़र और सिस्टम के बारे में जानकारी प्रदान करता है।
javascript
// navigator गुणों तक पहुँच
var browserName = navigator.userAgent;
var isMobile = navigator.userAgent.match(/Mobile/);

स्क्रीन ऑब्जेक्ट (Screen Object)—
screen ऑब्जेक्ट प्रयोक्ता की स्क्रीन की जानकारी प्रदान करता है, जैसे चौड़ाई, ऊँचाई और रंग गहराई।

javascript
// screen गुणों तक पहुँच
var screenWidth = screen.width;
var screenHeight = screen.height;

इतिहास ऑब्जेक्ट (History Object)—
history ऑब्जेक्ट ब्राउज़र के इतिहास में नेविगेशन करने की सुविधा देता है, जैसे आगे और पीछे जाना।
javascript
// history विधियों का उपयोग
window.history.back(); // पिछला पृष्ठ खोलें
window.history.forward(); // अगला पृष्ठ खोलें

लोकेशन ऑब्जेक्ट (Location Object)—
location ऑब्जेक्ट वर्तमान URL की जानकारी देता है और अन्य URL पर नेविगेट करने की अनुमति देता है।
javascript
// location गुणों तक पहुँच
var currentURL = location.href;
// URL बदलना
location.href = "https://www.example.com";

कुकीज़ (Cookies)—
ब्राउज़र छोटे डेटा टुकड़ों (cookies) को स्टोर करने की सुविधा प्रदान करता है, जो प्रयोक्ता के कंप्यूटर पर स्टोर होते हैं।
javascript
// कुकी बनाना
document.cookie = "username=John; expires=Thu, 18 Dec 2023 12:00:00 UTC; path=/";

टाइमर्स (Timers)—
BOM कोड निष्पादन को अनुसूचित करने की विधियाँ प्रदान करता है, जैसे setTimeout() और setInterval()।
javascript
// setTimeout का उपयोग
setTimeout(function() {
  alert("Delayed alert!");
}, 2000); // 2 सेकंड बाद अलर्ट दिखाएँ

निष्कर्ष—
ब्राउज़र ऑब्जेक्ट मॉडल जावास्क्रिप्ट को ब्राउज़र के परिवेश के साथ संवाद करने की क्षमता प्रदान करता है—जैसे इतिहास प्रबंधन, कुकीज़ को संभालना, प्रयोक्ता एजेंट की जानकारी प्राप्त करना आदि। यह गतिशील और इंटरैक्टिव वेब एप्लीकेशन बनाने में एक अत्यंत महत्त्वपूर्ण भूमिका निभाता है।

13.6.2 विंडो ईवेंट्स (Window Events)
JavaScript में विंडो ईवेंट्स (window events) आपको ब्राउज़र विंडो के अंदर होने वाली विभिन्न इंटरैक्शन और परिवर्तनों पर प्रतिक्रिया देने की अनुमति देते हैं। नीचे कुछ सामान्यतः प्रयुक्त विंडो ईवेंट्स दिए गए हैं —
load ईवेंट— load ईवेंट तब सक्रिय होता है जब कोई वेब पेज या उसके सभी बाहरी संसाधन (जैसे चित्र, स्टाइलशीट्स, स्क्रिप्ट्स आदि) लोड हो जाते हैं।

javascript
```
window.addEventListener("load", function() {
  // पेज लोड होने के बाद चलने वाला कोड
});
```

unload ईवेंट— unload ईवेंट ठीक उस समय सक्रिय होता है जब प्रयोक्ता पृष्ठ छोड़ने वाला होता है (उदाहरण के लिए, ब्राउज़र विंडो बंद करते समय या किसी अन्य पृष्ठ पर जाते समय)।

javascript
```
window.addEventListener("unload", function() {
  // पृष्ठ छोड़ने से पहले की क्लियरिंग का कोड
});
```

resize ईवेंट— resize ईवेंट तब सक्रिय होता है जब ब्राउज़र विंडो का आकार बदला जाता है।

javascript
```
window.addEventListener("resize", function() {
  // विंडो आकार परिवर्तन की स्थिति में कार्य
});
```

scroll ईवेंट— scroll ईवेंट तब सक्रिय होता है जब प्रयोक्ता पृष्ठ को स्क्रॉल करता है।

javascript
```
window.addEventListener("scroll", function() {
  // स्क्रॉलिंग की स्थिति में प्रतिक्रिया देने वाला कोड
});
```

beforeunload ईवेंट— beforeunload ईवेंट ठीक उस समय सक्रिय होता है जब प्रयोक्ता पृष्ठ छोड़ने वाला होता है। इसका उपयोग पुष्टिकरण संवाद (confirmation dialog) दिखाने हेतु किया जा सकता है।

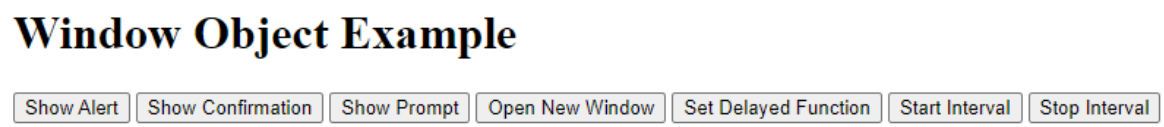
javascript
```
window.addEventListener("beforeunload", function(event) {
  event.returnValue = "क्या आप वाकई इस पृष्ठ को छोड़ना चाहते हैं?";
});
```

ये JavaScript में विंडो ईवेंट्स के कुछ उदाहरण हैं जैसा कि चित्र 13.10 में दर्शाया गया है। आप इन ईवेंट्स पर प्रतिक्रिया देने के लिए इवेंट लिस्नर्स (event listeners) का उपयोग कर सकते हैं और ब्राउज़र विंडो में प्रयोक्ता की क्रियाओं अथवा परिवर्तनों के अनुसार कार्य कर सकते हैं। इवेंट हैंडलिंग, इंटरैक्टिव और गत्यात्मक वेब अनुप्रयोग (web applications) बनाने की मूलभूत प्रक्रिया का हिस्सा है, जैसा कि कोड में प्रदर्शित किया गया है।

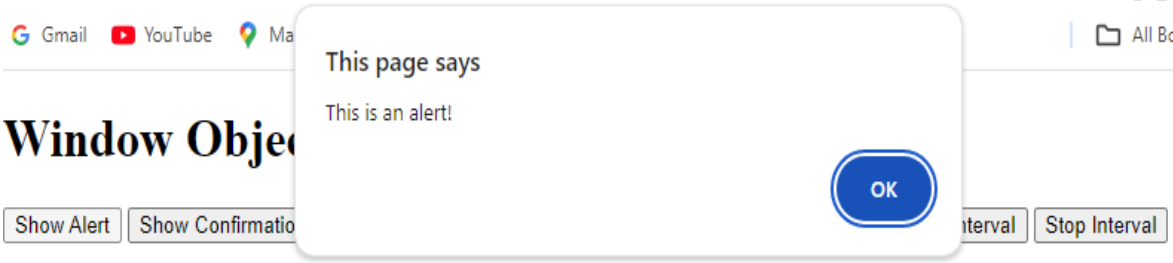
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Window Object Example</title>
    <script>
        function showAlert() {
            window.alert("This is an alert!")
        }
        function showConfirmation() {
            var result = window.confirm("Do you want to continue?")
            if (result) {
                console.log("User clicked OK.")
            }else{
                console.log("User clicked cancel.")
            }
        }
        function showPrompt() {
            var name = window.prompt("Enter your name:", "Vijay Goswami")
            if (name !== null) {
                console.log("User entered: " + name)
            }else{
                console.log("User clicked cancel.")
            }
        }
        function openNewWindow() {
            window.open("https://www.example.com", "_blank", "width=500, height=300")
        }
        function setDelayedFunction() {
            window.setTimeout(function() {
                console.log("Delayed function executed!")
            }, 2000) //Execute after 2 seconds
        }
        var intervalId
        function startInterval() {
            intervalId = window.setInterval(function() {
                console.log("Interval function executed!")
            }, 1000) //Execute every 1 second
        }
        function stopInterval() {
            clearInterval(intervalId)
            console.log("Interval stopped.")
        }
    </script>
</head>
<body>
    <h1>Window Object Example</h1>
    <button onclick="showAlert()">Show Alert</button>
    <button onclick="showConfirmation()">Show Confirmation</button>
    <button onclick="showPrompt()">Show Prompt</button>
    <button onclick="openNewWindow()">Open New Window</button>
    <button onclick="setDelayedFunction()">Set Delayed Function</button>
    <button onclick="startInterval()">Start Interval</button>
    <button onclick="stopInterval()">Stop Interval</button>
</body>
</html>
```

चित्र 13.10— JavaScript में विंडो ईवेंट्स

आउटपुट



चित्र 13.11— JavaScript में विंडो ईवेंट्स का आउटपुट



चित्र 13.12— Show Alert बटन पर क्लिक करने के बाद का आउटपुट (JavaScript में विंडो ईवेंट्स)

इसी प्रकार, यह प्रत्येक बटन पर कार्य करता है।

13.7 दस्तावेज़ वस्तु मॉडल (Document Object Model)

Document Object Model (DOM) वेब दस्तावेजों के लिए एक महत्वपूर्ण प्रोग्रामिंग इंटरफेस के रूप में कार्य करता है, जो HTML या XML दस्तावेज़ का एक संरचित निरूपण पेड़ जैसी संरचना (tree-like structure) में प्रदान करता है। यह मॉडल मूलतः एक पेड़ संरचना का निर्माण करता है जिसमें प्रत्येक नोड (node) दस्तावेज़ के किसी विशेष भाग या तत्व जैसे कि शीर्षक, अनुच्छेद, चित्र आदि का प्रतिनिधित्व करता है। DOM का प्रमुख उद्देश्य वेब पृष्ठों के साथ गत्यात्मक (dynamic) तरीके से इंटरैक्शन को संभव बनाना है, जिससे प्रोग्राम और स्क्रिप्ट्स वेब पृष्ठ की सामग्री, संरचना और प्रस्तुति में संशोधन कर सकें।

व्यावहारिक रूप से, DOM डेवलपर्स को JavaScript जैसी स्क्रिप्टिंग भाषाओं का उपयोग करके वेब पृष्ठ के विभिन्न घटकों तक पहुँचने और उनमें परिवर्तन करने की अनुमति देता है, चाहे वह किसी टेक्स्ट को बदलना हो, किसी तत्व का स्वरूप परिवर्तित करना हो, या प्रयोक्ता की क्रियाओं पर प्रतिक्रिया देना हो। यह गत्यात्मक क्षमता आधुनिक वेब एप्लीकेशन विकास की मूल आत्मा है, जहाँ बिना पूरे पृष्ठ को पुनः लोड किए सामग्री को तात्कालिक रूप से बदला जा सकता है। संक्षेप में, DOM आधुनिक वेब विकास का एक आधारभूत घटक है, जो स्थिर वेब पृष्ठों को गत्यात्मक और प्रयोक्ता के लिए सहज अनुभवों में बदलने के लिए माध्यम प्रदान करता है।

DOM के प्रमुख तत्व और अवधारणाएँ इस प्रकार हैं—

पेड़ संरचना (Tree Structure)—

- DOM किसी HTML या XML दस्तावेज़ को पेड़ संरचना के रूप में निरूपित करता है, जिसमें नोड्स होते हैं।
- सबसे उच्च स्तर का नोड document object होता है, जो पूरे वेब पृष्ठ का प्रतिनिधित्व करता है।
- DOM में नोड्स तत्व (elements), गुण (attributes), टेक्स्ट, टिप्पणियाँ आदि को निरूपित कर सकते हैं।

तत्वों तक पहुँच (Accessing Elements) —

- आप DOM में विभिन्न तरीकों और गुणों का उपयोग करके वेब पृष्ठ पर तत्वों को चुनकर उन तक पहुँच सकते हैं और उनमें परिवर्तन कर सकते हैं।

- सामान्यतः प्रयुक्त विधियाँ हैं — getElementById, getElementsByClassName, getElementsByTagName, और querySelector।
- document.write() तथा अन्य चयन विधियाँ जैसे getElementById, getElementsByClassName, getElementsByTagName, और querySelector वेब पृष्ठ के DOM में तत्वों तक पहुँचने और उन्हें परिवर्तित करने के लिए सामान्य तकनीकें हैं। आइए प्रत्येक विधि को विस्तार से समझें

document.write(content)—

- document.write() विधि का उपयोग HTML दस्तावेज़ में वर्तमान स्थान पर सीधे सामग्री लिखने के लिए किया जाता है, जो सामान्यतः <body> टैग के अंदर होता है।
- इसका उपयोग प्रायः वेब पृष्ठ पर गत्यात्मक रूप से सामग्री जोड़ने के लिए किया जाता है।

javascript
```
document.write("<h1>Hello, World!</h1>");
```

नोट— document.write() का उपयोग करते समय सावधानी रखें, क्योंकि यदि यह विधि पृष्ठ के पूर्णतः लोड हो जाने के बाद कॉल की जाती है तो यह पूरे दस्तावेज़ को अधिलेखित कर सकती है। बेहतर नियंत्रण के लिए सामान्यतः अन्य DOM हेरफेर विधियों का प्रयोग अनुशंसित किया जाता है।

getElementById(id)— getElementById() विधि किसी तत्व को उसके अद्वितीय id गुण द्वारा DOM से प्राप्त करती है।

javascript
```
var element = document.getElementById("myElement");
```

getElementsByClassName(className)— getElementsByClassName() विधि किसी निर्दिष्ट क्लास नाम वाले तत्वों का संग्रह (array-like object) लौटाती है।

javascript
```
var elements = document.getElementsByClassName("myClass");
```

getElementsByTagName(tagName)— getElementsByTagName() विधि किसी विशेष HTML टैग नाम वाले तत्वों का संग्रह लौटाती है।

javascript
```
var paragraphs = document.getElementsByTagName("p");
```

querySelector(selector)— querySelector() विधि DOM से एक CSS चयनकर्ता के आधार पर एकल तत्व को चयनित करती है। यह पहला मेल खाता तत्व लौटाती है।

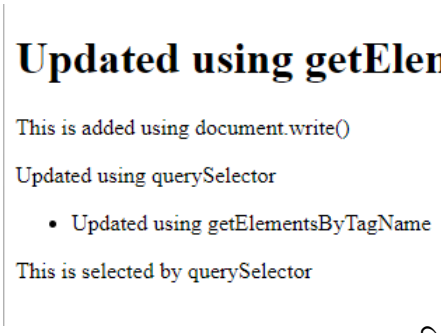
javascript
```
var element = document.querySelector(".myClass");
```

ये DOM चयन विधियाँ वेब पृष्ठ पर तत्वों के साथ इंटरैक्ट करने और उन्हें हेरफेर करने हेतु आवश्यक हैं, जैसा कि चित्र 13.13 में दिखाया गया है। ये विधियाँ आपको विशिष्ट तत्वों तक पहुँचने, उनकी सामग्री व गुणों को संशोधित करने तथा प्रयोक्ता की क्रियाओं के अनुसार प्रतिक्रिया देने की सुविधा देती हैं। उपयुक्त विधि का चयन करते समय अपने कार्य की आवश्यकताओं और लचीलापन स्तर पर विचार करें।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Selection and document.write Example</title>
</head>
<body>
    <h1 id="header">DOM Selection Example</h1>
    <!-- Using document.write() to add content -->
    <script>
        document.write("<p>This is added using document.write()</p>")
    </script>
    <!-- Using getElementById to modify content -->
    <script>
        var header = document.getElementById("header")
        header.innerHTML = "Updated using getElementById"
    </script>
    <!-- Using getElementsByClassName to modufy content -->
    <div>
        <p class="myClass">This is selected by getElementsByClassName</p>
    </div>
    <script>
        var elementByClass = document.getElementsByClassName("myClass")[0]
        elementByClass.textContent = "Updated using getElementsByClassName"
    </script>
    <!-- Using getElementsByTagName to modufy content -->
    <ul>
        <li>This is selected by getElementsByTagName</li>
    </ul>
    <script>
        var elementsByTagName = document.getElementsByTagName("li")[0]
        elementsByTagName.textContent = "Updated using getElementsByTagName"
    </script>
    <!-- Using querySelector to modify content -->
    <div>
        <p>This is selected by querySelector</p>
    </div>
    <script>
        var elementByQuery = document.querySelector(".myClass")
        elementByQuery.textContent = "Updated using querySelector"
    </script>
</body>
</html>
```

चित्र 13.13— Document Object Model (DOM) विधियाँ

आउटपुट



चित्र 13.14— DOM विधियों का आउटपुट

व्याख्या

- <!DOCTYPE html>: दस्तावेज़ प्रकार और संस्करण को HTML5 के रूप में घोषित करता है। <html lang="en">: दस्तावेज़ की मूल तत्व को निरूपित करता है, और lang="en" इंगित करता है कि भाषा अंग्रेज़ी है।
- <head>: दस्तावेज़ के बारे में मेटाडेटा (metadata) को सम्मिलित करता है, जैसे कि कैरेक्टर सेट और शीर्षक।
- <meta charset="UTF-8">: उचित टेक्स्ट प्रदर्शन हेतु कैरेक्टर एन्कोडिंग को UTF-8 पर सेट करता है।
- <title>: वेब पृष्ठ का शीर्षक सेट करता है, जो ब्राउज़र की शीर्ष पट्टी में प्रदर्शित होता है।
- <body>: वेब पृष्ठ की दृश्य सामग्री को संलग्न करता है।
- <h1 id="header">DOM Selection Example</h1>: एक <h1> शीर्षक तत्व को परिभाषित करता है जिसका id="header" है और उसमें टेक्स्ट सामग्री सेट की गई है।
- <!-- Using document.write() to add content -->: स्क्रिप्ट के उद्देश्य को स्पष्ट करने वाली टिप्पणी।
- <script>: JavaScript स्क्रिप्ट की शुरुआत को इंगित करता है।
- document.write(" This is added using document.write() ");: यह विधि गत्यात्मक रूप से HTML सामग्री जोड़ती है, उदाहरणतः टैग सहित।
- <!-- Using getElementById to modify content -->: एक और टिप्पणी जो प्रयोजन स्पष्ट करती है।
- var header = document.getElementById("header");: यह पंक्ति id="header" वाले तत्व को प्राप्त करती है और header वेरिएबल में स्टोर करती है।
- header.innerHTML = "Updated using getElementById";: innerHTML गुण के द्वारा header तत्व की सामग्री को अपडेट करती है।
- <!-- Using getElementsByClassName to modify content -->: एक टिप्पणी।
- और : "myClass" क्लास वाले HTML तत्व।
- var elementByClass = document.getElementsByClassName("myClass")[0];: यह पंक्ति "myClass" क्लास वाले सभी तत्वों को प्राप्त करती है और पहले को elementByClass में स्टोर करती है।
- elementByClass.textContent = "Updated using getElementsByClassName";: चयनित तत्व की टेक्स्ट सामग्री को अपडेट करती है।
- <!-- Using getElementsByTagName to modify content -->: एक टिप्पणी।
- और <li>: अनुक्रमहीन सूची और सूची आइटम तत्व।

- var elementsByTagName = document.getElementsByTagName("li")[0];: यह सभी तत्वों को प्राप्त करती है और पहले को elementsByTagName में स्टोर करती है।
- elementsByTagName.textContent = "Updated using getElementsByTagName";: चयनित <li> की टेक्स्ट सामग्री को अपडेट करती है।
- <!-- Using querySelector to modify content -->: एक टिप्पणी।
- var elementByQuery = document.querySelector(".myClass");: यह CSS चयनकर्ता के आधार पर पहले "myClass" क्लास वाले तत्व का चयन करता है।
- elementByQuery.textContent = "Updated using querySelector";: चयनित तत्व की टेक्स्ट सामग्री को अपडेट करता है।

ये उदाहरण दर्शाते हैं कि कैसे JavaScript के माध्यम से document.write() और विभिन्न DOM चयन विधियों का उपयोग करके HTML दस्तावेज़ में तत्वों तक पहुँचा और उनमें परिवर्तन किया जा सकता है।

13.8 जावास्क्रिप्ट में वेरिएबल्स (Variables)

JavaScript में वेरिएबल्स डेटा को संग्रहित करने और उसका संचालन करने में एक महत्त्वपूर्ण भूमिका निभाते हैं। यह बहुपरकारी भाषा डायनामिकली टाइप्ड होती है, अर्थात जब आप किसी वेरिएबल को घोषित करते हैं, तो आपको उसके डेटा प्रकार को स्पष्ट रूप से बताने की आवश्यकता नहीं होती। इसके स्थान पर JavaScript स्वतः ही उसमें असाइन किए गए मान के आधार पर डेटा प्रकार निर्धारित कर लेता है। नीचे JavaScript में वेरिएबल्स को घोषित करने और उपयोग करने के लिए एक मूल मार्गदर्शिका प्रस्तुत है।

13.8.1 घोषणा और मान असाइन करना

आप वेरिएबल को var, let, या const कीवर्ड द्वारा घोषित कर सकते हैं—

var (JavaScript के पुराने संस्करणों में प्रयुक्त) —

```
var myVariable;
```

let (ECMAScript 6 में प्रस्तुत, उन वेरिएबल्स के लिए प्रयुक्त जो परिवर्तित हो सकते हैं) —

```
let myVariable;
```

const (ECMAScript 6 में प्रस्तुत, उन स्थिरांकों के लिए जो पुनः असाइन नहीं किए जा सकते) —

```
const myConstant = 10;
```

वेरिएबल घोषित करने के बाद, आप उसमें कोई मान असाइन कर सकते हैं—

```
let myVariable = "Hello, world!";
```

13.8.2 वेरिएबल नामकरण नियम

JavaScript में वेरिएबल नामों के लिए निम्नलिखित नियम लागू होते हैं—

• अक्षर के केस (बड़े/छोटे) का महत्त्व होता है; myVariable और myvariable अलग-अलग वेरिएबल होते हैं।
• वेरिएबल नाम किसी अक्षर, अंडरस्कोर (_) या डॉलर चिह्न ($) से प्रारंभ होना चाहिए।
• इसके बाद अंकों (0-9) का भी प्रयोग किया जा सकता है।
• वेरिएबल नामों में रिक्त स्थान या विशेष वर्ण (अंडरस्कोर और डॉलर चिह्न को छोड़कर) नहीं होने चाहिए।
• कुछ शब्द जैसे let, const, if, for आदि आरक्षित शब्द होते हैं और वेरिएबल नाम के रूप में प्रयुक्त नहीं किए जा सकते।

13.8.3 डेटा प्रकार

JavaScript में कई मूलभूत डेटा प्रकार होते हैं जैसा कि चित्र 13.14 में दिखाया गया है—

प्राथमिक (Primitive) डेटा प्रकार — ये अपरिवर्तनीय होते हैं और इनमें शामिल हैं—

• **Number** — फ्लोटिंग पॉइंट संख्याएँ (जैसे— 3.14, -42)

• **String** — टेक्स्ट डेटा (जैसे— "Hello")

• **Boolean** — true या false

• **Undefined** — जिन वेरिएबल्स को कोई मान असाइन नहीं किया गया है

• **Null** — किसी वस्तु के अभाव को दर्शाता है

जटिल (Complex) डेटा प्रकार —

• **Object** — कुंजी-मूल्य (key-value) जोड़ों का संग्रह (जैसे— { name: "Vijay", age: 38 })

• **Array** — मानों की क्रमबद्ध सूची (जैसे— [1, 2, 3])

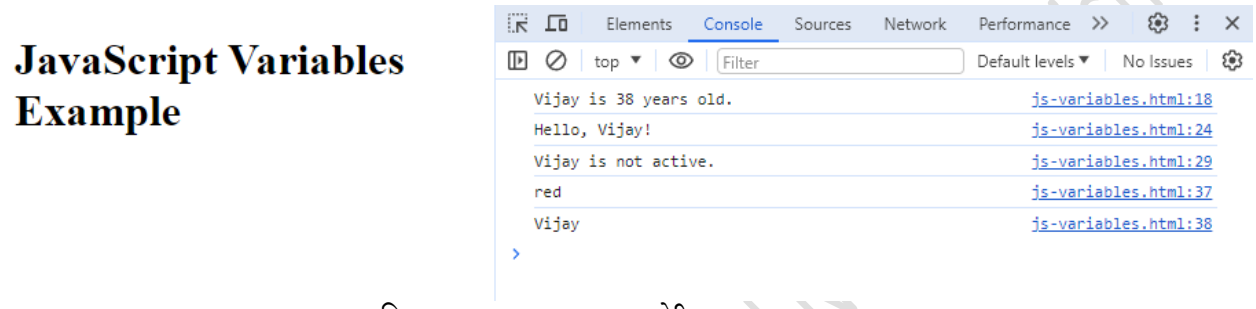
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript Variables Example</title>
</head>
<body>
<h1>JavaScript Variables Example</h1>

<script>
        // Declaring and initializing variables
        let name = "Vijay";
        const age = 38;
        var isActive = true;

        // Using variables in calculations
        let birthYear = 2022 - age;
       console.log(name + " is " + age + " years old.");

        // Changing variable values
        isActive = false;
        // Concatenating strings using variables
        let greeting = "Hello, " + name + "!";
        console.log(greeting);
        // Using variables in conditions
        if (isActive) {
        console.log(name + " is active.");
        } else {
        console.log(name + " is not active.");
        }
        // Arrays and objects using variables
        let colors = ["red", "green", "blue"];
        let person = {
            firstName: "Vijay",
            lastName: "Goswami"
        };
        console.log(colors[0]); // Outputs "red"
        console.log(person.firstName); // Outputs "Vijay"
</script>
</body>
</html>
```

चित्र 13.14— JavaScript में कई मूल डेटा प्रकार होते हैं

इस कोड को चलाने के लिए निम्न चरणों का पालन करें—

1. एक टेक्स्ट एडिटर (जैसे— Windows में Notepad या macOS में TextEdit) खोलें।
2. ऊपर दिए गए HTML कोड को उसमें चिपकाएँ।
3. फ़ाइल को .html एक्सटेंशन (जैसे— index.html) के साथ सहेजें।
4. उस फ़ाइल को डबल-क्लिक करके या राइट-क्लिक करके "Open with" से कोई ब्राउज़र चुनकर खोलें।
5. ब्राउज़र में HTML फ़ाइल खुलने के बाद, JavaScript आउटपुट देखने के लिए डिवेलपर कंसोल खोलें। ऐसा आप आमतौर पर पेज पर राइट-क्लिक करके "Inspect" या "Inspect Element" चुनकर और "Console" टैब में जाकर कर सकते हैं।

आप HTML फ़ाइल में JavaScript कोड का आउटपुट देख पाएँगे, जिसमें वेरिएबल अभ्यास, गणनाएँ और शर्त युक्त कथन (conditional statements) शामिल होंगे, जैसा कि चित्र 13.15 में दर्शाया गया है।



चित्र 13.15— JavaScript वेरिएबल्स का आउटपुट

13.9 वेरिएबल्स पर प्रयुक्त गणितीय ऑपरेटर

JavaScript में, आप वेरिएबल्स पर विभिन्न गणितीय ऑपरेटरों का प्रयोग कर गणनाएँ और संख्यात्मक डेटा का संचालन कर सकते हैं। नीचे कुछ सामान्य ऑपरेटरों का विवरण दिया गया है—

Addition (+)— दो या अधिक संख्याओं को जोड़ने के लिए प्रयुक्त

```
javascript
let num1 = 5;
let num2 = 10;
let sum = num1 + num2;
```

Subtraction (-)— एक संख्या को दूसरी संख्या से घटाने के लिए

```
javascript
let num1 = 15;
let num2 = 7;
let difference = num1 - num2;
```

Multiplication (*)— दो या अधिक संख्याओं को गुणा करने के लिए

```
javascript
let num1 = 6;
let num2 = 8;
let product = num1 * num2;
```

Division (/)— एक संख्या को दूसरी से भाग देने के लिए

```
javascript
```

```
let num1 = 20;
let num2 = 4;
let quotient = num1 / num2;
```

Modulus (%)— दो संख्याओं के भाग देने पर शेषफल प्रदान करता है

```
javascript
let num1 = 21;
let num2 = 4;
let remainder = num1 % num2;
```

Increment (++) और Decrement (--)— वेरिएबल के मान को 1 से बढ़ाने या घटाने के लिए

```
javascript
let count = 5;
count++; // 1 से बढ़ाएँ
let result = count--; // 1 से घटाएँ (पोस्ट-डिक्रिमेंट)
```

**Assignment Operators (+=, -=, =, /=)—* किसी वेरिएबल पर गणितीय क्रिया करके उसका नया मान उसी वेरिएबल में असाइन करने के लिए

```
javascript
let num = 10;
num += 5; // num = num + 5 के समान
```

अभ्यास 13.3

- निम्नलिखित गणितीय ऑपरेटरों का उद्देश्य और सिंटैक्स लिखिए —
  - Modulus %
  - Subtraction -
  - Increment ++ और Decrement --
  - Assignment Operators (+=, -=, *=, /=)
  - Division /

13.10 ऑपरेटर प्रेसीडेंस (Operator Precedence)

JavaScript यह निर्धारित करता है कि जब किसी अभिव्यक्ति (expression) में एक से अधिक ऑपरेटर हों, तो किस क्रम में उनका मूल्यांकन किया जाए। ऑपरेटर प्रेसीडेंस को समझना **सही और पूर्वानुमेय कोड** लिखने के लिए आवश्यक है। JavaScript में ऑपरेटर की प्राथमिकता का सामान्य विवरण नीचे चित्र 13.16 में दिया गया है—

- **Grouping Parentheses ()** — सबसे उच्च प्राथमिकता वाले; पहले हल किए जाते हैं
- **Member Access . और Computed Member Access []** — ऑब्जेक्ट या ऐरे की प्रविष्टियाँ एक्सेस करने के लिए
- **Function Invocation ()** — फ़ंक्शन को कॉल करने पर पहले उसके आर्गुमेंट्स का मूल्यांकन होता है
- **New Operator new** — नया ऑब्जेक्ट बनाने के लिए
- **Increment/Decrement ++, -- (Postfix)** — वेरिएबल के मान को बाद में बढ़ाता या घटाता है
- **Logical NOT !** — Boolean मान को उलटता है
- **Unary Plus + और Unary Negation -** — मान को संख्या में बदलने के लिए
- **Exponentiation **** — घातांक निकालने के लिए

• **Multiplication *, Division /, Modulus %** — गुणा, भाग और शेषफल हेतु
• **Addition +, Subtraction -** — जोड़ और घटाव हेतु
• **Concatenation + (strings के लिए)** — स्ट्रिंग्स को जोड़ने हेतु
• **Relational Operators <, >, <=, >=, instanceof, in** — मानों की तुलना के लिए
• **Equality Operators ==, !=, ===, !==** — समानता या असमानता की तुलना के लिए
• **Logical AND &&** — दोनों सत्य होने पर दूसरा मान लौटाता है
• **Logical OR ||** — किसी एक मान के सत्य होने पर उसे लौटाता है
• **Ternary Operator ? —** — if...else का संक्षिप्त रूप
• **Assignment Operators =, +=, -=, *=, /=, %= **=** — मान असाइन करने एवं गणना करने हेतु
• **Comma ,** — कई अभिव्यक्तियों को क्रमवार मूल्यांकित करने हेतु

जब अभिव्यक्तियों में अलग-अलग प्रेसीडेंस स्तर वाले ऑपरेटर होते हैं, तो JavaScript उपरोक्त नियमों का पालन करते हुए मूल्यांकन का क्रम निर्धारित करता है। यदि आवश्यक हो, तो आप **पेरेंथेसिस** का उपयोग करके स्पष्ट रूप से मूल्यांकन क्रम को नियंत्रित कर सकते हैं।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mathematical Operators Example</title>
</head>
<body>
    <h1>Mathematical Operators Example</h1>

    <script>
        let a = 10
        let b = 5

        let addition = a + b // 15
        let subtraction = a - b // 5
        let multiplication = a * b // 50
        let division = a / b // 2
        let modulus = a % b // 0
        let incrementPost = a++ // 10 (post increament)
        let decrementPost = a-- // 11 (post decrement)

        let exponentiation = 2 ** 3 // 8 (2 to the power of 3)

        let isGreaterThan = a > b // true
        let isEqualTo = a === b // false
        let logicalAnd = true && false // false
        let logicalOr = true || false // true

        let conditional = a > b ? "a is greater" : "b is greater" // a is greater
```

```
        console.log("Addition:", addition)
        console.log("Subtraction:", subtraction)
        console.log("Multiplication:", multiplication)
        console.log("Division:", division)
        console.log("Modulus:", modulus)
        console.log("Increment (Post):", incrementPost)
        console.log("Decrement (Post):", decrementPost)
        console.log("Exponentiation:", exponentiation)
        console.log("Is Greater Than:", isGreaterThan)
        console.log("Is Equal To:", isEqualTo)
        console.log("Logical AND:", logicalAnd)
        console.log("Logical Or:", logicalOr)
        console.log("Conditional:", conditional)
    </script>
</body>
</html>
```

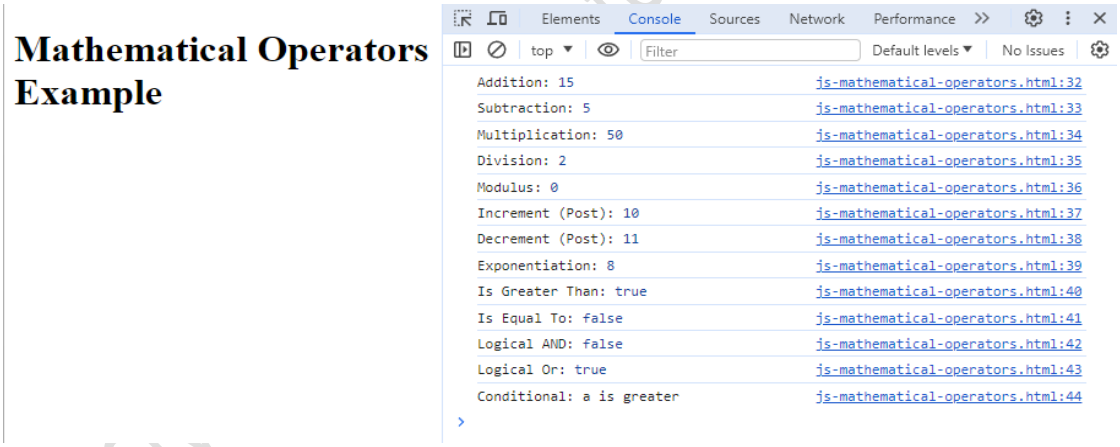
चित्र 13.16— JavaScript में गणितीय ऑपरेटर प्रेसीडेंस

स्पष्टीकरण

इस HTML उदाहरण में जैसा कि चित्र 13.17 में दिखाया गया है—

- HTML संरचना में एक शीर्षक है।
- <script> ब्लॉक के अंदर हम वेरिएबल a और b घोषित करते हैं तथा विभिन्न ऑपरेटरों के प्रयोग से गणनाएँ करते हैं।
- परिणाम ब्राउज़र के डिवेलपर कंसोल में लॉग होते हैं।

आउटपुट



चित्र 13.17— JavaScript में गणितीय ऑपरेटर प्रेसीडेंस का आउटपुट

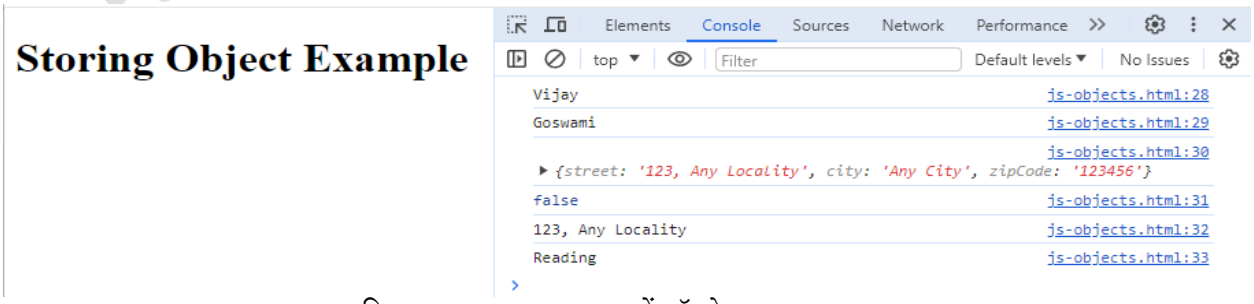
13.11 जावास्क्रिप्ट ऑब्जेक्ट्स को वेरिएबल्स में स्टोर करना

JavaScript में, आप किसी अन्य डेटा प्रकार की तरह ही **ऑब्जेक्ट्स को वेरिएबल्स में स्टोर** कर सकते हैं, जैसा कि चित्र 13.18 में दर्शाया गया है। JavaScript में ऑब्जेक्ट्स का प्रयोग कुंजी–मूल्य (key–value) जोड़ों का संग्रह प्रदर्शित करने के लिए किया जाता है, और इनमें स्ट्रिंग, संख्या, अन्य ऑब्जेक्ट्स, फ़ंक्शन्स और बहुत कुछ शामिल हो सकते हैं। नीचे बताया गया है कि आप JavaScript ऑब्जेक्ट्स को वेरिएबल्स में कैसे स्टोर कर सकते हैं।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript Objects Example</title>
</head>
<body>
   <h1>Storing Object Example</h1>
   <script>
        // Define an object
        let person = {
            firstName: "Vijay",
            lastName: "Goswami",
            age: 38,
            isStudent: false,
            address: {
                street: "123, Any Locality",
                city: "Any City",
                zipCode: "123456"
            },
            hobbies: ["Reading", "Travelling", "Music"]
        }
        // Store the object in a variable
        let myObject = person

        // Access object properties using the variable
        console.log(myObject.firstName) // Vijay
        console.log(myObject.lastName) // Goswami
        console.log(myObject.address) // 38
        console.log(myObject.isStudent) // false
        console.log(myObject.address.street) // 123, Any Locality
        console.log(myObject.hobbies[0]) // Reading
   </script>
</body>
</html>
```

चित्र 13.18— JavaScript में ऑब्जेक्ट्स का संग्रहन

आउटपुट



चित्र 13.18— JavaScript में ऑब्जेक्ट्स का आउटपुट

सारांश (SUMMARY)

- JavaScript वेब पृष्ठों को अंतःक्रियाशील (interactive) और गत्यात्मक (dynamic) सामग्री से सुसज्जित करता है।
- यह मुख्यतः एक क्लाइंट-साइड स्क्रिप्टिंग भाषा है, जिसका उपयोग वेब ब्राउज़र में किया जाता है।
- JavaScript गत्यात्मक रूप से टाइप की गई भाषा (dynamically typed language) है, जो वेरिएबल प्रकारों में लचीले रूपांतरण की अनुमति देती है।
- यह फॉर्म प्रमाणीकरण (form validation), एनिमेशन और घटना प्रबंधन (event handling) के लिए आवश्यक है।
- यह घटना-संचालित (event-driven) है, जो प्रयोक्ता की क्रियाओं पर घटना श्रोता (event listeners) के माध्यम से प्रतिक्रिया देता है।
- JavaScript वेब पृष्ठ की सामग्री को डॉक्यूमेंट ऑब्जेक्ट मॉडल (Document Object Model — DOM) के माध्यम से संशोधित कर सकता है।
- वेरिएबल को var, let या const का उपयोग करके परिभाषित किया जा सकता है। फ़ंक्शन कोड के समावेशन (encapsulation) और पुनः उपयोग के लिए मौलिक होते हैं।
- यदि...अन्यथा (if...else), for लूप तथा while लूप जैसी नियंत्रण प्रवाह संरचनाएँ उपलब्ध हैं।
- असमकालिक प्रोग्रामन (Asynchronous programming) को कॉलबैक्स (callbacks), प्रॉमिस Promises) और असिंक/अवेट (async/await) के माध्यम से संचालित किया जाता है।
- jQuery और React जैसी लाइब्रेरी (libraries) एवं रूपरेखाओं (frameworks) से वेब विकास सरल होता है।
- Node.js सर्वर-साइड (server-side) JavaScript विकास की सुविधा देता है।
- JavaScript लाइब्रेरी विभिन्न ब्राउज़रों में संगतता (cross-browser compatibility) सुनिश्चित करती हैं।
- JavaScript एक अक्षर-संवेदनशील (case-sensitive) भाषा है, जो बड़े और छोटे अक्षरों में अंतर करती है।
  इसे सीधे HTML दस्तावेज़ों में <script> टैग के माध्यम से सम्मिलित किया जा सकता है।

अपनी प्रगति जाँचें (CHECK YOUR PROGRESS)

क. बहुविकल्पीय प्रश्न (MULTIPLE CHOICE QUESTIONS)

1. वेब विकास में JavaScript का प्रमुख कार्य क्या है?
   (क) सर्वर-साइड स्क्रिप्टिंग
   (ख) वेब पृष्ठों की शैली निर्धारण
   (ग) वेब पृष्ठों में अंतःक्रियाशीलता जोड़ना
   (घ) डेटाबेस प्रबंधन
2. JavaScript किस प्रकार की भाषा है?
   (क) टाइट तरीके से टाइप की गई
   (ख) लूज़ तरीके से टाइप की गई
   (ग) स्थिर टाइप की गई
   (घ) संकलित
3. JavaScript मुख्यतः एक क्लाइंट-साइड स्क्रिप्टिंग भाषा है। इसका क्या अर्थ है?
   (क) यह सर्वर पर चलती है
   (ख) यह वेब ब्राउज़र में चलती है

(ग) यह डेटाबेस प्रबंधन के लिए प्रयुक्त होती है
(घ) यह एक स्थिर टाइप की गई भाषा है

4. JavaScript में घटना श्रोताओं (event listeners) की प्रमुख भूमिका क्या होती है?
(क) फॉर्म सत्यापन
(ख) प्रयोक्ता की क्रियाओं को संभालना
(ग) डेटा प्रकार परिभाषित करना
(घ) सर्वर-साइड कार्यों का प्रबंधन

5. Document Object Model को किस संक्षेपाक्षर से जाना जाता है?
(क) HTML
(ख) DOM
(ग) CSS
(घ) URL

6. JavaScript में वेरिएबल घोषित करने के तीन तरीके क्या हैं?
(क) var, const, let
(ख) int, string, bool
(ग) variable, let, const
(घ) variable1, variable2, variable3

7. JavaScript में असमकालिक प्रोग्रामन को संभालने के लिए कौन-सा तत्व प्रयुक्त होता है?
(क) प्रॉमिस (Promises)
(ख) कॉलबैक (Callbacks)
(ग) असिंक/अवेट (async/await)
(घ) उपरोक्त सभी

8. कौन-सी JavaScript लाइब्रेरी/रूपरेखा प्रयोक्ता इंटरफ़ेस को बेहतर बनाती है?
(क) jQuery
(ख) React
(ग) Angular
(घ) उपरोक्त सभी

9. कौन-सा रनटाइम परिवेश JavaScript को सर्वर पर चलाने की सुविधा देता है?
(क) Node.js
(ख) Java
(ग) Python
(घ) Ruby

10. निम्नलिखित में से कौन JavaScript में वेरिएबल घोषित करने का वैध तरीका नहीं है?
(क) let
(ख) variable
(ग) const
(घ) var

ख. रिक्त स्थान भरिए (FILL IN THE BLANKS)

1. JavaScript में वेरिएबल var, let या _____ के माध्यम से परिभाषित किए जा सकते हैं।
2. फ़ंक्शन सामान्यतः कोड के समावेशन के लिए _____ में प्रयुक्त होते हैं।
3. _____ JavaScript को सर्वर-साइड परिवेश में चलाने की सुविधा देता है।
4. JavaScript अक्षरमात्रा के प्रति _____ होता है।
5. JavaScript में पॉप-अप संवाद बॉक्स प्रदर्शित करने के लिए _____ फ़ंक्शन का प्रयोग किया जाता है।
6. navigator ऑब्जेक्ट प्रयोक्ता के _____ और प्रणाली की जानकारी प्रदान करता है।
7. history ऑब्जेक्ट आपको ब्राउज़र के _____ में नेविगेट करने की सुविधा देता है।
8. JavaScript प्रयोक्ता के कंप्यूटर में संग्रहित छोटे डेटा हिस्सों, जिन्हें _____ कहा जाता है, के साथ कार्य करने की विधियाँ प्रदान करता है।
9. _____ ऑब्जेक्ट वर्तमान URL की जानकारी प्रदान करता है और विभिन्न URL पर नेविगेट करने की सुविधा देता है।
10. JavaScript की नियंत्रण प्रवाह संरचनाएँ डेवलपर को अपने प्रोग्राम के प्रवाह और _____ को नियंत्रित करने की सुविधा देती हैं।

ग. सही या गलत बताइए

1. JavaScript लाइब्रेरी और फ्रेमवर्क विभिन्न वेब ब्राउज़र में कोड निष्पादन की सुसंगतता सुनिश्चित करते हैं।
2. JavaScript कक्षा-आधारित वस्तुनिष्ठ प्रोग्रामिंग मॉडल का उपयोग करता है।
3. JavaScript एक कैस-संवेदनशील भाषा है।
4. JavaScript मुख्य रूप से एक सर्वर-साइड स्क्रिप्टिंग भाषा है।
5. JavaScript को सीधे HTML दस्तावेज़ों में सम्मिलित किया जा सकता है।
6. JavaScript का उपयोग वेब पृष्ठों पर फॉर्म मान्यता और एनिमेशन के लिए किया जाता है।
7. JavaScript एक केस-असंवेदनशील भाषा है।
8. Document Object Model (DOM) वेब ब्राउज़रों द्वारा ब्राउज़र के साथ संपर्क स्थापित करने के लिए उपलब्ध कराई गई वस्तुओं का एक सेट है।
9. navigator ऑब्जेक्ट प्रयोक्ता की स्क्रीन के बारे में जानकारी प्रदान करता है।
10. JavaScript में setTimeout() फ़ंक्शन किसी निर्दिष्ट विलंब के बाद किसी फ़ंक्शन को निष्पादित करने के लिए अनुसूचित करता है।

घ. लघु उत्तर प्रश्न (SHORT ANSWER QUESTIONS)

1. वेब विकास में JavaScript क्या भूमिका निभाता है?
2. क्लाइंट-साइड और सर्वर-साइड स्क्रिप्टिंग में क्या अंतर है?
3. Document Object Model (DOM) क्या है, और इसका JavaScript से क्या संबंध है?
4. JavaScript में वेरिएबल घोषित करने के तीन तरीके बताइए।
5. JavaScript में इवेंट लिसनर्स (event listeners) का क्या उद्देश्य होता है?
6. JavaScript असमकालिक प्रोग्रामिंग को कैसे संचालित करता है, और यह क्यों महत्त्वपूर्ण है?
7. वेब विकास के लिए JavaScript की तीन प्रमुख पुस्तकालयों या रूपरेखाओं के नाम बताइए।
8. JavaScript विकास में Node.js का क्या महत्व है?
9. JavaScript को केस-संवेदनशील (case-sensitive) भाषा क्यों माना जाता है?
10. ऑपरेटर प्राथमिकता (Operator precedence) को समझाइए।

सत्र 14
सशर्त तर्क और प्रवाह नियंत्रण
(Conditional Logic and Flow Control)

अंकित पहेलियों और खेलों को बहुत पसंद करता था, उसने "सशर्त तर्क और प्रवाह नियंत्रण" (Conditional Logic and Flow Control) के बारे में जाना, जो समझदारी से निर्णय लेने से जुड़ा हुआ था। यह कुछ ऐसा था जैसे वह अपनी ही रोमांचक कहानी का निर्देशक बन गया हो, जहाँ वह सवाल पूछ सकता था, उत्तरों के आधार पर निर्णय ले सकता था और किसी कहानी या खेल के प्रवाह को निर्देशित कर सकता था — ठीक उसी तरह जैसे एक *"अपना-अनुभव-खुद चुनें"* वाली किताब में होता है। इन उपकरणों की सहायता से उसने मज़ेदार खेल और कहानियाँ बनाईं जिनमें खिलाड़ी विकल्प चुन सकते थे, जिससे डिजिटल दुनिया और अधिक संवादात्मक (interactive) और मनोरंजक बन गई — ठीक किसी पहेली या रोमांचक खेल की तरह। जैसा कि चित्र 14.1 में दिखाया गया है।

चित्र 14.1— अंकित सशर्त तर्क और प्रवाह नियंत्रण का उपयोग करते हुए

इस अध्याय में, आप **अभिव्यक्ति में सशर्त तर्क**, **JavaScript में प्रवाह नियंत्रण**, और **लूप्स** के बारे में जानेंगे।

14.1 अभिव्यक्ति में सशर्त तर्क (Conditional Logic in Expression)

JavaScript में सशर्त तर्क (conditional logic) का उपयोग शर्तों के आधार पर निर्णय लेने और विभिन्न कोड खंडों (code blocks) को निष्पादित करने के लिए किया जाता है, यह इस बात पर निर्भर करता है कि शर्त सही (true) है या गलत (false)। आमतौर पर इसका उपयोग if, else if, और else कथनों के माध्यम से किया जाता है। नीचे सशर्त तर्क का विवरण और कुछ उदाहरण दिए गए हैं—

1. if कथन—

JavaScript में, "if" कथन का उपयोग तब किया जाता है जब कोई विशिष्ट शर्त सत्य (true) हो। यदि शर्त असत्य (false) है, तो कोड खंड निष्पादित नहीं होता।

सिंटैक्स—

```
javascript
if (condition) {
   // यदि शर्त सत्य है तो यह कोड निष्पादित होगा
}
```

उदाहरण—

```
javascript
let age = 20;
```

```
if (age >= 18) {
   console.log("You are an adult.");
}
```

2. else if कथन—

"else if" कथन का उपयोग तब किया जाता है जब पहले दी गई if शर्त असत्य हो, और हम कोई अन्य शर्त जाँचना चाहते हों।

सिंटैक्स—

```
javascript
if (condition1) {
   // यदि condition1 सत्य है, तो यह कोड निष्पादित होगा
} else if (condition2) {
   // यदि condition2 सत्य है, तो यह कोड निष्पादित होगा
}
```

उदाहरण—

```
javascript
let temperature = 25;
if (temperature < 0) {
   console.log("It's freezing!");
} else if (temperature >= 0 && temperature < 20) {
   console.log("It's cool.");
} else {
   console.log("It's warm.");
}
```

3. else कथन—

else कथन का उपयोग तब किया जाता है जब if और else if शर्तें दोनों असत्य हों। ऐसे में else खंड निष्पादित होता है।

सिंटैक्स—

```
javascript
if (condition) {
   // यदि शर्त सत्य है तो यह कोड निष्पादित होगा
} else {
   // यदि शर्त असत्य है तो यह कोड निष्पादित होगा
}
```

उदाहरण—

```
javascript
let isRaining = false;
if (isRaining) {
   console.log("Don't forget your umbrellक.");
} else {
```

```
    console.log("No need for an umbrella today.");
}
```

4. टर्नरी ऑपरेटर (? :)—

टर्नरी ऑपरेटर एक संक्षिप्त तरीका प्रदान करता है सशर्त अभिव्यक्तियों को लिखने के लिए। यह किसी शर्त का मूल्यांकन करता है और उसके आधार पर दो में से कोई एक मान प्रदान करता है — यह इस बात पर निर्भर करता है कि शर्त सत्य है या असत्य।

सिंटैक्स—

```
javascript
let result = condition ? valueIfTrue : valueIfFalse;
```

उदाहरण—

```
javascript
let age = 20;
let message = age >= 18 ? "You are an adult." : "You are not an adult.";
console.log(message);
```

क्या आप जानते हैं?

सशर्त तर्क एक शक्तिशाली उपकरण है जिसका उपयोग आपके कोड के प्रवाह को नियंत्रित करने के लिए किया जा सकता है। सशर्त कथनों का उपयोग करके, आप अपने कोड को अधिक लचीला और विभिन्न स्थितियों के प्रति उत्तरदायी बना सकते हैं।

5. तुलनात्मक ऑपरेटर (Comparison Operators)

JavaScript में तुलनात्मक ऑपरेटरों का उपयोग मानों की तुलना करने और बूलियन (Boolean) परिणाम (true या false) प्राप्त करने के लिए किया जाता है। सामान्य तुलनात्मक ऑपरेटरों में निम्न शामिल हैं—

== (समानता)— यह जाँचता है कि क्या दो मान समान हैं।

!= (असमानता)— यह जाँचता है कि क्या दो मान समान नहीं हैं।

=== (कठोर समानता)— यह जाँचता है कि दो मान और उनका डेटा प्रकार दोनों समान हैं या नहीं।

!== (कठोर असमानता)— यह जाँचता है कि दो मान या उनका डेटा प्रकार समान नहीं हैं।

< (छोटा)— यह निर्धारित करता है कि क्या बाईं ओर दिया गया मान दाईं ओर के मान से छोटा है।

(बड़ा)— यह जाँचता है कि क्या बाईं ओर दिया गया मान दाईं ओर के मान से बड़ा है।

<= (छोटा या बराबर)— यह जाँचता है कि क्या बाईं ओर दिया गया मान दाईं ओर के मान से छोटा या बराबर है।

= (बड़ा या बराबर)— यह जाँचता है कि क्या बाईं ओर दिया गया मान दाईं ओर के मान से बड़ा या बराबर है।

उदाहरण

```
javascript
let x = 5;
let y = 10;
console.log(x === y); // आउटपुट— false
```

6. तार्किक ऑपरेटर (Logical Operators)

JavaScript में तार्किक ऑपरेटरों का उपयोग बूलियन मानों पर तार्किक क्रियाएँ करने के लिए किया जाता है। सामान्य तार्किक ऑपरेटरों में शामिल हैं—

&& (तार्किक AND)— यदि दोनों मान true हों तो true लौटाता है।
|| (तार्किक OR)— यदि कम से कम एक मान true हो तो true लौटाता है।
! (तार्किक NOT)— दिए गए मान को उलट देता है (negate करता है)।

उदाहरण

javascript

```
let isRaining = true;
let isCold = false;
if (isRaining && !isCold) {
   console.log("छाता लेकर जाएँ।");
} else if (isCold || !isRaining) {
   console.log("जैकेट पहनें।");
}
```

7. नेस्टेड इफ स्टेटमेंट्स (Nested if Statements)

जब आप एक से अधिक स्तरों पर शर्तों की जाँच करना चाहते हैं, तो नेस्टेड if स्टेटमेंट्स का उपयोग किया जाता है। इसमें एक if स्टेटमेंट के अंदर दूसरा if स्टेटमेंट होता है।

उदाहरण

javascript

```
let x = 10;
if (x > 5) {
   if (x < 15) {
      console.log("x का मान 5 और 15 के बीच है।");
   }
}
```

8. स्विच स्टेटमेंट्स (Switch Statements)

स्विच स्टेटमेंट्स का उपयोग विभिन्न स्थितियों के आधार पर अलग-अलग क्रियाएँ करने के लिए किया जाता है। यह कई if...else if...else स्टेटमेंट्स के मुकाबले एक अधिक सुव्यवस्थित और दक्ष विकल्प प्रदान करता है, जैसा कि **चित्र 14.2** में दर्शाया गया है।

```
let day = "Monday";
switch (day) {
    case "Monday":
        console.log("It's the start of the workweek.");
        break;
    case "Friday":
        console.log("It's almost the weekend.");
        break;
    default:
        console.log("It's some other day.");
}
```

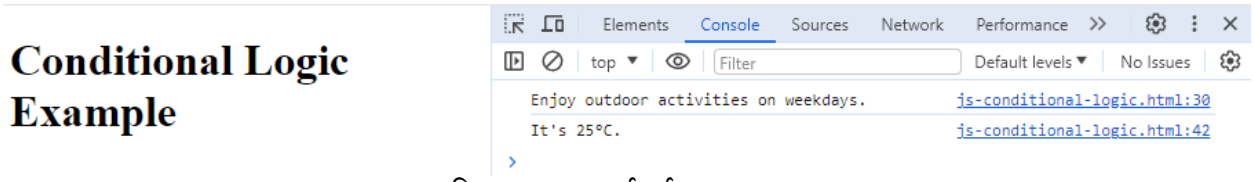
चित्र 14.2— स्विच स्टेटमेंट्स

सशर्त तर्क (Conditional Logic) का एक संपूर्ण उदाहरण चित्र 14.3 में दर्शाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Conditional Logic Example</title>
</head>
<body>
<h1>Conditional Logic Example</h1>
<script>
        // Variables
        let temperature = 25;
        let isSunny = true;
        let isWeekend = false;
        // Comparison operators
        let isHot = temperature > 30;
        // Ternary operator
        let weatherMessage = isSunny ? "It's a sunny day!" : "It's not sunny today.";
        // Logical operators
        let outdoorActivity = isSunny && !isHot;
        // Nested if statements
        if (outdoorActivity) {
            if (isWeekend) {
                console.log("Enjoy outdoor activities on the weekend.");
            } else {
                console.log("Enjoy outdoor activities on weekdays.");
            }
        } else {
            console.log("Consider indoor activities.");
        }
        // Switch statement
        switch (temperature) {
            case 30:
                console.log("It's 30°C.");
                break;
            case 25:
                console.log("It's 25°C.");
                break;
            default:
                console.log("Temperature is not 30°C or 25°C.");
        }
</script>
</body>
</html>
```

चित्र 14.3— सशर्त तर्क

आउटपुट



चित्र 14.4— सशर्त तर्क का आउटपुट

कोड व्याख्या (Code Explanation)

1. **HTML संरचना—** इस भाग में वेबपृष्ठ की मूल HTML संरचना को परिभाषित किया गया है, जिसमें डाक्यूमेंट टाइप (<!DOCTYPE html>), HTML टैग, कैरेक्टर सेट और पृष्ठ का शीर्षक शामिल होता है।

2. **JavaScript कोड—** JavaScript कोड <script> टैग के अंदर लिखा गया है और यह वेबपृष्ठ के प्रसंग (context) में चलता है।
3. **चर घोषणाएँ—**
   let temperature = 25; — temperature नामक चर को 25 मान के साथ घोषित किया गया है, जो वर्तमान तापमान को दर्शाता है।
   let isSunny = true; — isSunny नामक चर true मान के साथ घोषित किया गया है, जो धूप होने को दर्शाता है।
   let isWeekend = false; — isWeekend नामक चर false मान के साथ घोषित किया गया है, जो सप्ताहांत न होने को दर्शाता है।
4. **तुलनात्मक ऑपरेटर—**
   let isHot = temperature > 30; — तापमान को 30 से तुलना करने के लिए > ऑपरेटर का उपयोग किया गया है। यदि तापमान 30 से अधिक है तो isHot का मान true होगा, अन्यथा false।
5. **टर्नरी ऑपरेटर—**
   let weatherMessage = isSunny ? "It's a sunny day!" — "It's not sunny today."; — टर्नरी ऑपरेटर ? — का उपयोग isSunny के true या false होने के आधार पर weatherMessage को निर्धारित करने हेतु किया गया है।
6. **तार्किक ऑपरेटर—**
   let outdoorActivity = isSunny && !isHot; — AND (&&) और NOT (!) ऑपरेटर का उपयोग कर यह निर्धारित किया गया है कि मौसम बाहरी गतिविधियों के लिए उपयुक्त है या नहीं।
7. **नेस्टेड if स्टेटमेंट्स—**
   इन स्टेटमेंट्स के माध्यम से यह जाँचा जाता है कि क्या outdoorActivity true है और क्या isWeekend भी true है। इन शर्तों के आधार पर कंसोल में विभिन्न संदेश प्रदर्शित किए जाते हैं।
8. **स्विच स्टेटमेंट—**
   स्विच स्टेटमेंट temperature के मान की जाँच करता है और उस पर आधारित विभिन्न कोड खंड निष्पादित करता है।

यह कोड JavaScript में सशर्त तर्क (Conditional Logic) के कार्य को दर्शाता है — जो विभिन्न स्थितियों के आधार पर निर्णय लेने और संदेश प्रदर्शित करने में सहायक होता है।

प्रयोगात्मक अभ्यास 14.1
- निम्नलिखित स्टेटमेंट्स की संरचना (Syntax) लिखिए —
- if
- else if
- switch
- nested if

14.2 JavaScript में प्रवाह नियंत्रण (Flow Control)

JavaScript में प्रवाह नियंत्रण (Flow Control) से तात्पर्य उन तंत्रों और संरचनाओं से है जो प्रोग्राम के निष्पादन क्रम को नियंत्रित करते हैं। यह आपको निर्णय लेने, कोड को दोहराने (loop करने) और विभिन्न स्थितियों के आधार पर अलग-अलग कोड खंड निष्पादित करने की सुविधा देता है। प्रोग्रामिंग में नियंत्रण संरचनाएँ (control structures) वे निर्माण और स्टेटमेंट्स होती हैं जो आपके कोड के निष्पादन प्रवाह को नियंत्रित करने की क्षमता प्रदान करती हैं। ये संरचनाएँ निर्णय लेने, कार्यों को दोहराने और आपके प्रोग्राम की तर्क-संगत प्रगति को प्रबंधित करने में सहायक होती हैं।

JavaScript में प्रवाह नियंत्रण की कुछ प्रमुख अवधारणाएँ निम्नलिखित हैं —

1. **सशर्त स्टेटमेंट्स का कोड**, जैसा कि **चित्र 14.5** में दर्शाया गया है।

```
// If Statements: An if statement allows you to execute a block of code if a specified condition is true
if (condition) {
    // Code to execute if the condition is true
}

// Else Statements: You can use an else statement to specify a block of code to be executed if the
// conditionin the if statement is false.
if (condition) {
    // Code to execute if the condition is true
}else{
    // Code to execute if the condition is false
}

// Else If Statements: To handle multiple conditions, you can use else if statements in conjunction with
// if and else statements.
if (condition1) {
    // Code to execute if the condition1 is true
}else if (condition2) {
    // Code to execute if the condition2 is true
}else{
    // Code to execute if none of the conditions is true
}
```

चित्र 14.5— सशर्त स्टेटमेंट्स

2. **स्विच स्टेटमेंट का कोड**, जैसा कि **चित्र 14.6** में दिखाया गया है।
   स्विच स्टेटमेंट आपको किसी अभिव्यक्ति को अनेक संभावित केस मानों के विरुद्ध जाँचने और मिलान होने पर अलग-अलग कोड ब्लॉक निष्पादित करने की सुविधा देता है।

```
switch (expression) {
    case value1:
        //Code to execute if expression matches value1
        break;
    case value2:
        //Code to execute if expression matches value2
        break;
    default:
        //Code to execute if no case matches the expression
}
```

चित्र 14.6— स्विच स्टेटमेंट्स

3. **break और continue**
   break स्टेटमेंट का उपयोग किसी लूप से बीच में ही बाहर निकलने के लिए किया जाता है, जबकि continue स्टेटमेंट वर्तमान पुनरावृत्ति (iteration) को छोड़कर अगले पर जाने के लिए उपयोग किया जाता है।

break का उदाहरण—

javascript

```
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    break; // जब i का मान 5 हो जाए तो लूप से बाहर निकलें
  }
  // प्रत्येक पुनरावृत्ति में निष्पादित होने वाला कोड
}
```

continue का उदाहरण—

javascript

```
for (let i = 0; i < 10; i++) {
```

```
  if (i === 5) {
    continue; // जब i का मान 5 हो, तब इस पुनरावृत्ति का शेष कोड छोड़ें
  }
  // i = 5 को छोड़कर सभी पुनरावृत्तियों में निष्पादित होने वाला कोड
}
```

क्या आप जानते हैं?
JavaScript में प्रवाह नियंत्रण (Flow Control) से तात्पर्य **कार्यक्रम में स्टेटमेंट्स के निष्पादन के क्रम** से होता है।

4. लूप्स (Loops)

प्रोग्रामिंग में लूप्स नियंत्रण संरचनाएँ (control structures) होती हैं, जो किसी कोड खंड (block of code) को बार-बार निष्पादित करने की अनुमति देती हैं। ये उन कार्यों को निष्पादित करने के लिए आवश्यक होती हैं, जिनमें दोहराव की आवश्यकता होती है या जिनमें डेटा संग्रहों को संसाधित करना होता है। JavaScript में कई प्रकार के लूप्स होते हैं—

i. for लूप—

for लूप तब उपयोग किया जाता है जब पहले से यह ज्ञात हो कि कोड के किसी खंड को कितनी बार दोहराना है। यह तीन भागों से मिलकर बनता है— प्रारंभन (initialization), शर्त (condition), और पुनरावृत्ति (iteration)।

सिंटैक्स—

```
javascript
for (initialization; condition; iteration) {
  // प्रत्येक पुनरावृत्ति में निष्पादित होने वाला कोड
}
```

उदाहरण—

```
javascript
for (let i = 0; i < 5; i++) {
  console.log("Iteration " + i);
}
```

उपरोक्त उदाहरण में, यह लूप पाँच बार निष्पादित होगा, जिसमें i चर (variable) का मान 0 से 4 तक होगा।

ii. while लूप—

while लूप तब उपयोग किया जाता है जब आप किसी कोड खंड को तब तक दोहराना चाहते हैं, जब तक कि कोई निर्दिष्ट शर्त सत्य हो। यह प्रत्येक पुनरावृत्ति से पहले शर्त का मूल्यांकन करता है।

सिंटैक्स—

```
javascript
while (condition) {
  // जब तक शर्त सत्य है, तब तक निष्पादित होने वाला कोड
}
```

उदाहरण—

```
javascript
let count = 0;
while (count < 3) {
  console.log("Count is " + count);
  count++;
```

}

इस उदाहरण में लूप तीन बार निष्पादित होगा।

iii. do...while लूप—

do...while लूप while लूप के समान होता है, लेकिन यह इस बात की गारंटी देता है कि शर्त जाँचे जाने से पहले कोड खंड कम-से-कम एक बार निष्पादित होगा। यह प्रत्येक पुनरावृत्ति के बाद शर्त का मूल्यांकन करता है।

सिंटैक्स—

```
javascript
do {
   // कम-से-कम एक बार निष्पादित होने वाला कोड
} while (condition);
```

उदाहरण—

```
javascript
let x = 0;
do {
   console.log("x is " + x);
   x++;
} while (x < 3);
```

अभ्यास 14.2

- निम्नलिखित के सिंटैक्स को सूचीबद्ध कीजिए—
- do…..while लूप
- while लूप
- break और continue
- for लूप
- for...of और for...in लूप्स

यह लूप भी तीन बार निष्पादित होगा।

iv. for...of और for...in लूप्स (संग्रहों पर पुनरावृत्ति हेतु)—

for...of लूप का उपयोग किसी इटेरेबल (जैसे— array, string) के तत्वों पर पुनरावृत्ति करने के लिए किया जाता है।

for...in लूप का उपयोग किसी ऑब्जेक्ट की प्रॉपर्टीज़ पर पुनरावृत्ति करने के लिए किया जाता है।

ये लूप्स डेटा संग्रहों पर पुनरावृत्ति की प्रक्रिया को सरल बनाते हैं।

नीचे प्रत्येक लूप के उद्देश्य और उसके उपयोग की स्थिति का संक्षिप्त सारांश दिया गया है—

- for लूप— जब पुनरावृत्तियों की संख्या पहले से ज्ञात हो।
- while लूप— जब कोड को किसी शर्त के आधार पर दोहराना हो।
- do...while लूप— जब कोड को कम-से-कम एक बार निष्पादित करना आवश्यक हो।
- for...of लूप— arrays और अन्य इटेरेबल्स के तत्वों पर पुनरावृत्ति के लिए।
- for...in लूप— ऑब्जेक्ट की प्रॉपर्टीज़ पर पुनरावृत्ति के लिए।

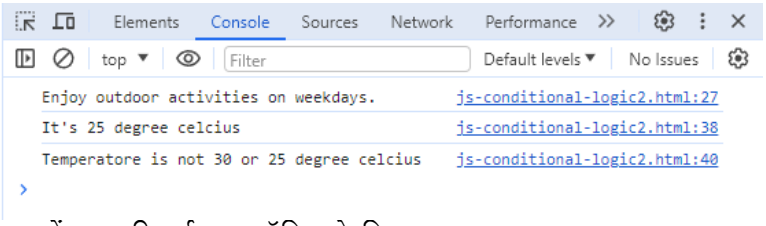
लूप्स दोहराए जाने वाले कार्यों को स्वचालित करने और डेटा संसाधित करने के लिए शक्तिशाली उपकरण हैं, जो प्रोग्रामिंग का एक मूलभूत भाग हैं। सभी लूप्स का एक समेकित उदाहरण **चित्र 14.7** में प्रदर्शित किया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Conditional Logic Example</title>
</head>
<body>
    <h1>Conditional Logic Example</h1>

    <script>
        // Variables
        let temperature = 25
        let isSunny = true
        let isWeekend = false
        // Comparison Operators
        let isHot = temperature > 30
        // Ternary Operator
        let weatherMessage = isSunny ? "It's a sunny day!" : "It's not a sunny day today."
        // Logical Operators
        let outdoorActivity = isSunny && !isHot


        // Nested if Statements
        if (outdoorActivity) {
            if (isWeekend) {
                console.log("Enjoy outdoor activities on the weekend.")
            }else{
                console.log("Enjoy outdoor activities on weekdays.")
            }
        }else{
            console.log("Consider indoor activities")
        }
        // Switch statement
        switch (temperature) {
            case 30:
                console.log("It's 30 degree celcius")
                break
            case 25:
                console.log("It's 25 degree celcius")
            default:
                console.log("Temperatore is not 30 or 25 degree celcius")
        }
    </script>
</body>
</html>
```

चित्र 14.7— JavaScript में लूप्स की शर्त युक्त (Conditional) लॉजिक

आउटपुट

Conditional Logic Example



चित्र 14.7— JavaScript में लूप्स की शर्त युक्त लॉजिक के लिए आउटपुट

कोड व्याख्या—

- इन पंक्तियों में तीन वेरिएबल्स (temperature, isSunny, और dayOfWeek) की घोषणा और आरंभिक मान निर्दिष्ट किए गए हैं।
- temperature को 28 पर सेट किया गया है, isSunny को true पर और dayOfWeek को "Saturday" पर। ये वेरिएबल वर्तमान तापमान, मौसम की स्थिति और सप्ताह के दिन की जानकारी स्टोर करते हैं।
- यहाँ पर if, else if, और else का उपयोग करके शर्त युक्त कथनों (conditional statements) की एक श्रृंखला है।
- पहला if कथन जाँचता है कि क्या तापमान 30 से अधिक है। यदि हाँ, तो यह "It's a hot day!" लॉग करता है।
- else if कथन जाँचता है कि क्या तापमान 20 से अधिक है। यदि हाँ, तो यह "It's a warm day." लॉग करता है।
- यदि उपरोक्त दोनों शर्तें असत्य हैं, तो else ब्लॉक निष्पादित होता है और "It's a cool day." लॉग किया जाता है।
- इसके बाद एक switch कथन है, जो dayOfWeek के मान की जाँच करता है।
- यदि dayOfWeek "Saturday" या "Sunday" है, तो यह "It's the weekend!" लॉग करता है।
- यदि dayOfWeek सप्ताह के कार्यदिवसों (Monday से Friday) में से कोई है, तो यह "It's a weekday." लॉग करता है।
- यदि dayOfWeek किसी भी केस से मेल नहीं खाता, तो default ब्लॉक निष्पादित होता है और "Invalid day." लॉग किया जाता है।
- इन पंक्तियों में विभिन्न प्रकार के लूप्स प्रदर्शित किए गए हैं—
  - पहला लूप एक for लूप है जो 0 से 4 तक (कुल पाँच बार) पुनरावृत्ति करता है और "Iteration" के बाद वर्तमान i मान को लॉग करता है।
  - दूसरा लूप एक while लूप है जो तब तक चलता है जब तक count 3 से कम है और "Count is" के बाद वर्तमान count मान को लॉग करता है। प्रत्येक पुनरावृत्ति में count को बढ़ाया जाता है।
  - तीसरा लूप do...while लूप है, जो कोड को कम-से-कम एक बार निष्पादित करता है और फिर शर्त की जाँच करता है। यह "x is" के बाद वर्तमान x मान को लॉग करता है और तब तक x को बढ़ाता है जब तक वह 3 से कम है। यह अनुभाग एक for लूप के साथ शर्त युक्त लॉजिक को जोड़ता है।
- यह 0 से 4 तक पुनरावृत्ति करता है और प्रत्येक i मान के लिए जाँच करता है कि वह सम (even) है या विषम (odd)।
- यदि i सम है (i % 2 == 0), तो यह "i is even." लॉग करता है।
- यदि i विषम है, तो यह "i is odघ." लॉग करता है।

ये कोड पंक्तियाँ विभिन्न नियंत्रण संरचनाओं और लूप्स का प्रदर्शन करती हैं, तथा यह भी दिखाती हैं कि कैसे इन्हें निर्णय लेने, डेटा पर पुनरावृत्ति करने और कार्यक्रम के प्रवाह को नियंत्रित करने के लिए प्रयोग में लाया जा सकता है।

सारांश (SUMMARY)

- JavaScript में शर्त युक्त लॉजिक (Conditional Logic) में if, else if, और else कथनों का उपयोग करके निर्णय लिए जाते हैं।
- Ternary ऑपरेटर का प्रयोग शर्त युक्त अभिव्यक्तियों को संक्षेप में लिखने के लिए किया जाता है।
- तुलनात्मक ऑपरेटर (comparison operators) का उपयोग मानों की तुलना के लिए किया जाता है।
- तार्किक ऑपरेटर (logical operators) बूलियन मानों पर तार्किक क्रियाएँ करते हैं।
- नेस्टेड if कथनों का उपयोग बहु-स्तरीय शर्तों की जाँच के लिए किया जाता है।
- switch कथन कई if...else if...else कथनों के संगठित विकल्प प्रदान करता है।
- JavaScript में प्रवाह नियंत्रण (Flow control) में शर्त युक्त कथन, switch कथन, break और continue तथा विभिन्न प्रकार के लूप्स सम्मिलित हैं।
- for, while, और do...while जैसे लूप्स शर्तों या पूर्व-निर्धारित पुनरावृत्तियों के आधार पर कोड को दोहराने की अनुमति देते हैं।
- for...of और for...in लूप्स का उपयोग क्रमवाचक डेटा और ऑब्जेक्ट प्रॉपर्टीज़ पर पुनरावृत्ति हेतु किया जाता है।

ये नियंत्रण संरचनाएँ JavaScript में कार्यों के स्वचालन और प्रोग्राम प्रवाह के प्रबंधन हेतु अनिवार्य हैं।

अपनी प्रगति जाँचें (CHECK YOUR PROGRESS)

क. बहुविकल्पीय प्रश्न

1. JavaScript में उस संरचना का नाम बताइए जो किसी शर्त के सत्य होने पर ही कोड के खंड को निष्पादित करती है —
   (क) switch कथन
   (ख) while लूप
   (ग) if कथन
   (घ) for लूप

2. JavaScript में वह ऑपरेटर कौन-सा है जो मूल्य (value) और डेटा प्रकार (data type) दोनों की *सख्त समानता* की जाँच करता है —
   (क) ==
   (ख) ===
   (ग) !=
   (घ) !==

3. "else if" कथन का उपयोग तब किया जाता है जब पहले दिया गया "if" कथन —
   (क) सत्य हो
   (ख) असत्य हो
   (ग) समान हो
   (घ) अपरिभाषित हो

4. JavaScript में वह तर्कसंगत ऑपरेटर कौन-सा है जो केवल तभी *true* देता है जब दोनों ऑपरेण्ड *true* हों —
   (क) &&

(ख) ||
(ग) !
(घ) &

5. JavaScript में किसी लूप को समय से पहले समाप्त करने के लिए किस कुंजीशब्द का प्रयोग किया जाता है —
(क) exit
(ख) quit
(ग) break
(घ) skip

6. JavaScript में कौन-सा लूप तब प्रयोग किया जाता है जब पहले से ज्ञात हो कि कोड को कितनी बार दोहराना है —
(क) while लूप
(ख) do...while लूप
(ग) for लूप
(घ) for...of लूप

7. "switch" कथन में वह कौन-सा कुंजीशब्द है जिसका प्रयोग तब किया जाता है जब कोई भी केस मान अभिव्यक्ति से मेल नहीं खाता —
(क) default
(ख) none
(ग) other
(घ) unmatched

8. JavaScript में "continue" कथन का क्या कार्य है —
(क) लूप से बाहर निकलना
(ख) वर्तमान पुनरावृत्ति के शेष कोड को छोड़ना और अगली पर जाना
(ग) लूप को रीसेट करना
(घ) लूप को आरंभ से दोहराना

9. निम्न में से कौन-सा JavaScript में सामान्य तुलना ऑपरेटर नहीं है —
(क) <
(ख) >=
(ग) ||
(घ) !==

10. JavaScript में टर्नरी ऑपरेटर का उपयोग करते समय, उस प्रतीक का नाम बताइए जो *सत्य* होने पर लौटाई जाने वाली मान और *असत्य* होने पर लौटाई जाने वाली मान को अलग करता है —
(क) ;
(ख) —

(ग) |
(घ) ,

ख. रिक्त स्थान भरें

1. JavaScript में "if" कथन का प्रयोग केवल तब कोड के खंड को चलाने के लिए होता है जब कोई विशेष शर्त _____ होती है।
2. टर्नरी ऑपरेटर संक्षिप्त रूप में _____ अभिव्यक्तियाँ बनाने की विधि प्रदान करता है।
3. JavaScript में तुलना ऑपरेटर का उपयोग मूल्यों को _____ और बूलियन परिणाम लौटाने के लिए किया जाता है।
4. JavaScript में तर्कसंगत ऑपरेटर का उपयोग _____ मानों पर तर्कसंगत कार्य करने के लिए किया जाता है।
5. _____ if कथनों का प्रयोग तब किया जाता है जब आपको अनेक स्तरों पर शर्तों की जाँच करनी होती है।
6. _____ कथन का प्रयोग विभिन्न शर्तों के आधार पर भिन्न कार्य करने के लिए किया जाता है।
7. _____ कथन का उपयोग किसी लूप को समय से पहले समाप्त करने के लिए किया जाता है।
8. _____ कथन का उपयोग लूप की वर्तमान पुनरावृत्ति को छोड़ने और अगली पुनरावृत्ति पर जाने के लिए किया जाता है।
9. for...of लूप का उपयोग किसी इटरेबल जैसे कि ऐरे या स्ट्रिंग के तत्वों पर _____ करने के लिए किया जाता है।
10. _____ लूप का उपयोग किसी ऑब्जेक्ट के गुणों पर पुनरावृत्ति करने के लिए किया जाता है।

ग. सत्य या असत्य

1. JavaScript में "if" कथन का उपयोग कोड के उस खंड को चलाने के लिए किया जाता है जो केवल किसी विशेष शर्त के सत्य होने पर निष्पादित होता है।
2. "else if" कथन का उपयोग केवल तभी अतिरिक्त शर्तों की जाँच करने के लिए किया जाता है जब पूर्व की "if" शर्त सत्य हो।
3. JavaScript में टर्नरी ऑपरेटर शर्त युक्त अभिव्यक्तियों को संक्षिप्त रूप में लिखने का एक तरीका है।
4. JavaScript में तुलना ऑपरेटर का प्रयोग मूल्यों की तुलना कर बूलियन परिणाम प्राप्त करने के लिए किया जाता है।
5. JavaScript में तर्कसंगत ऑपरेटर का उपयोग बूलियन मानों पर तर्कसंगत कार्य करने के लिए किया जाता है।
6. नेस्टेड if कथनों का उपयोग अनेक स्तरों की शर्तों की जाँच के लिए किया जाता है।
7. JavaScript में switch कथन का उपयोग कई "if...else if...else" कथनों के स्थान पर किया जा सकता है।
8. "break" कथन का प्रयोग किसी लूप को समय से पहले समाप्त करने के लिए किया जाता है।
9. "continue" कथन का प्रयोग लूप की वर्तमान पुनरावृत्ति को छोड़ने और अगली पुनरावृत्ति पर जाने के लिए किया जाता है।
10. "for" लूप का प्रयोग तब किया जाता है जब पहले से ज्ञात हो कि कोड को कितनी बार दोहराना है।

घ. लघुउत्तरीय प्रश्न

1. जावास्क्रिप्ट में सशर्त तर्क क्या है, और प्रोग्रामिंग में यह क्यों महत्त्वपूर्ण है?
2. जावास्क्रिप्ट में "if" कथन की मूल वाक्यरचना और उपयोग को समझाइए।
3. "else if" कथन "if" कथन से किस प्रकार भिन्न है, और इसका उपयोग कब किया जाता है?
4. जावास्क्रिप्ट में "else" कथन का उद्देश्य और वाक्यरचना को वर्णित कीजिए।
5. जावास्क्रिप्ट में टर्नरी ऑपरेटर क्या है, और यह कैसे कार्य करता है?
6. जावास्क्रिप्ट में कुछ सामान्य तुलना ऑपरेटरों की सूची बनाइए और उनके उपयोगों को समझाइए।
7. जावास्क्रिप्ट में तार्किक ऑपरेटर क्या हैं, और उनके उपयोगों के उदाहरण दीजिए।
8. नेस्टेड if कथनों का उपयोग कब उपयुक्त होता है, और वे कैसे कार्य करते हैं?
9. स्विच कथन क्या है, और यह कई if...else if...else कथनों की तुलना में कोड को कैसे सरल बनाता है?
10. जावास्क्रिप्ट में लूप के प्रकारों को संक्षेप में समझाइए और प्रत्येक के लिए एक उपयोग का उदाहरण दीजिए।

सत्र 15

ऐरे (Arrays) और फ़ंक्शन (Functions)

अंकित को पहेलियाँ और चीज़ें बनाना बहुत पसंद था, "ऐरे (Arrays) और फ़ंक्शन (Functions)" की दुनिया में पहुँचा। यह उसके लिए ऐसे था मानो उसके पास चीज़ों से भरा एक ख़ज़ाना और उन पर कार्य करने वाले जादुई उपकरण हों। ऐरे ने उसे चीज़ों को व्यवस्थित तरीके से रखने में मदद की, जबकि फ़ंक्शन उसके रोबोट मित्र जैसे थे जो उसके आदेशों का पालन करते थे। उसने फ़ंक्शन की मदद से ऐसे खेल बनाए जिनमें खिलाड़ी वस्तुएँ एकत्र करते और स्कोर गिनते थे, यहाँ तक कि उसने अंकों को जोड़ने वाला एक कैलकुलेटर फ़ंक्शन भी बनाया। ऐरे और फ़ंक्शन उसके गुप्त सहायक बन गए थे, जो डिजिटल दुनिया को और अधिक रोमांचक बनाते और उसे हर प्रकार की डिजिटल साहसिक चीज़ें बनाने की शक्ति देते — जैसे कि खज़ानों से भरी एक डिजिटल दुनिया। जैसा कि चित्र 15.1 में दर्शाया गया है।

चित्र 15.1— अंकित, ऐरे और फ़ंक्शन का उपयोग करते हुए

इस अध्याय में, आप ऐरे, ऐरे विधियाँ, ऐरे और लूप, फ़ंक्शन, फ़ंक्शन रिटर्न, फ़ंक्शन और फ़ंक्शन के पैरामीटर एवं आर्ग्युमेंट्स के बीच का अंतर सीखेंगे।

15.1 ऐरे (Arrays)

ऐरे बनाना और उनमें डेटा भरना प्रोग्रामिंग की एक मौलिक अवधारणा है, जिससे आप डेटा के संग्रह को संगठित और प्रबंधित कर सकते हैं। JavaScript में, आप विभिन्न तरीकों से ऐरे बना और भर सकते हैं—

ऐरे लिटरल (Array Literal)—

ऐरे बनाने का सबसे सामान्य तरीका वर्ग कोष्ठक [] का उपयोग करना है। इसमें मानों को कॉमा से अलग करके रखा जाता है।

उदाहरण—

javascript

```
let fruits = ["apple", "banana", "cherry"];
```

ऐरे कंस्ट्रक्टर (Array Constructor)—

आप Array कंस्ट्रक्टर का उपयोग करके ऐरे बना सकते हैं और उसमें मान आर्ग्युमेंट के रूप में दे सकते हैं।

उदाहरण—

javascript

```
let colors = new Array("red", "green", "blue");
```

रिक्त ऐरे (Empty Array)—

आप एक खाली ऐरे बनाकर, बाद में विशिष्ट इंडेक्स पर मान असाइन कर सकते हैं।

उदाहरण—

javascript

```
let cars = [];
cars[0] = "Toyota";
cars[1] = "Honda";
```

ऐरे विधियाँ (Array Methods)—

JavaScript में push(), unshift(), concat(), splice() जैसी विधियाँ हैं जिनसे मौजूदा ऐरे में तत्व जोड़े जा सकते हैं।

उदाहरण (push() का उपयोग)—

```
javascript
let animals = ["dog", "cat"];
animals.push("elephant");
```

लूप का उपयोग (Using Loops)—

आप for या while लूप का उपयोग करके ऐरे में गतिशील रूप से डेटा जोड़ सकते हैं।

उदाहरण (for लूप के साथ)—

```
javascript
let numbers = [];
for (let i = 1; i <= 5; i++) {
  numbers.push(i);
}
```

ऐरे डीस्ट्रक्चरिंग (Array Destructuring)—

ऐरे को विघटित (destructure) करके, आप अलग-अलग वैरिएबल में मान असाइन कर सकते हैं।

उदाहरण—

```
javascript
let [firstName, lastName] = ["John", "Doe"];
```

स्प्रेड ऑपरेटर (Spread Operator)—

स्प्रेड ऑपरेटर ... का उपयोग करके एक ऐरे के तत्वों को दूसरे ऐरे में जोड़ा जा सकता है।

उदाहरण—

```
javascript
let fruits = ["apple", "banana"];
let moreFruits = [...fruits, "cherry", "orange"];
```

ऐरे कंप्रिहेंशन (Array Comprehension, ES6)—

आधुनिक JavaScript (ES6 और आगे) में आप ऐरे कंप्रिहेंशन का उपयोग करके संक्षिप्त तरीके से ऐरे बना सकते हैं।

उदाहरण—

```
javascript
let numbers = [1, 2, 3, 4, 5];
let squaredNumbers = [x * x for (x of numbers)];
```

Array.from() का उपयोग (ES6)—

Array.from() मेथड किसी iterable या ऐरे-जैसे ऑब्जेक्ट से ऐरे बना सकता है।

उदाहरण—

javascript

```
let arrayFromSet = Array.from(new Set([1, 2, 2, 3, 4]));
```

15.2 ऐरे विधियाँ (Array Methods)

ऐरे विधियाँ JavaScript में अंतर्निर्मित फ़ंक्शन होते हैं जिनकी मदद से आप ऐरे को संशोधित कर सकते हैं। नीचे कुछ सामान्यत— उपयोग में आने वाली ऐरे विधियाँ दी गई हैं—

push()— ऐरे के अंत में एक या एक से अधिक तत्व जोड़ता है और ऐरे की नई लंबाई लौटाता है।

javascript

```
let fruits = ["apple", "banana"];
fruits.push("cherry");
// fruits अब है ["apple", "banana", "cherry"]
```

pop()— ऐरे से अंतिम तत्व को हटाता है और उसे लौटाता है।

javascript

```
let fruits = ["apple", "banana", "cherry"];
let removedFruit = fruits.pop();
// fruits अब है ["apple", "banana"], और removedFruit है "cherry"
```

shift()— ऐरे से पहला तत्व हटाता है और उसे लौटाता है।

javascript

```
let fruits = ["apple", "banana", "cherry"];
let removedFruit = fruits.shift();
// fruits अब है ["banana", "cherry"], और removedFruit है "apple"
```

unshift()— ऐरे की शुरुआत में एक या अधिक तत्व जोड़ता है और ऐरे की नई लंबाई लौटाता है।

javascript

```
let fruits = ["apple", "banana"];
fruits.unshift("cherry");
// fruits अब है ["cherry", "apple", "banana"]
```

concat()— दो या अधिक ऐरे को जोड़कर एक नया ऐरे लौटाता है।

javascript

```
let fruits1 = ["apple", "banana"];
let fruits2 = ["cherry", "orange"];
let combinedFruits = fruits1.concat(fruits2);
// combinedFruits है ["apple", "banana", "cherry", "orange"]
```

slice()— मूल ऐरे का एक भाग लेकर एक नया ऐरे लौटाता है।

javascript

```
let fruits = ["apple", "banana", "cherry", "orange"];
let slicedFruits = fruits.slice(1, 3);
// slicedFruits है ["banana", "cherry"]
```

forEach()— ऐरे के प्रत्येक तत्व पर दिए गए फ़ंक्शन को एक बार निष्पादित करता है।

```
javascript
let numbers = [1, 2, 3];
numbers.forEach(function (number) {
  console.log(number);
});
// आउटपुट— 1, 2, 3
```

map()— ऐरे के प्रत्येक तत्व पर दिए गए फ़ंक्शन को लागू कर एक नया ऐरे बनाता है।

```
javascript
let numbers = [1, 2, 3];
let doubledNumbers = numbers.map(function (number) {
  return number * 2;
});
// doubledNumbers है [2, 4, 6]
```

filter()— दिए गए परीक्षण को पास करने वाले सभी तत्वों के साथ एक नया ऐरे बनाता है।

```
javascript
let numbers = [1, 2, 3, 4, 5];
let evenNumbers = numbers.filter(function (number) {
  return number % 2 === 0;
});
// evenNumbers है [2, 4]
```

अभ्यास 15.1

- निम्नलिखित ऐरे विधियों का विवरण सूचीबद्ध करें —
  - push()
  - filter()
  - map()
  - concat()
  - slice()

15.3 ऐरे (Arrays) और लूप्स (Loops)

ऐरे और लूप अक्सर एक साथ उपयोग किए जाते हैं। आप लूप का उपयोग करके ऐरे के तत्वों पर इटरेशन (दोहराव) कर सकते हैं और उन पर विभिन्न क्रियाएँ कर सकते हैं। नीचे उन उदाहरणों को दर्शाया गया है जिनमें लूप का उपयोग करके ऐरे पर कार्य किया गया है, जैसा कि **चित्र 15.2** में दर्शाया गया है—

for लूप—

```
javascript
let fruits = ["apple", "banana", "cherry"];
for (let i = 0; i < fruits.length; i++) {
   console.log(fruits[i]);
}
// आउटपुट— "apple", "banana", "cherry"
```

forEach() विधि—

```
javascript
let fruits = ["apple", "banana", "cherry"];
fruits.forEach(function (fruit) {
   console.log(fruit);
});
// आउटपुट— "apple", "banana", "cherry"
```

for...of लूप (ES6)—

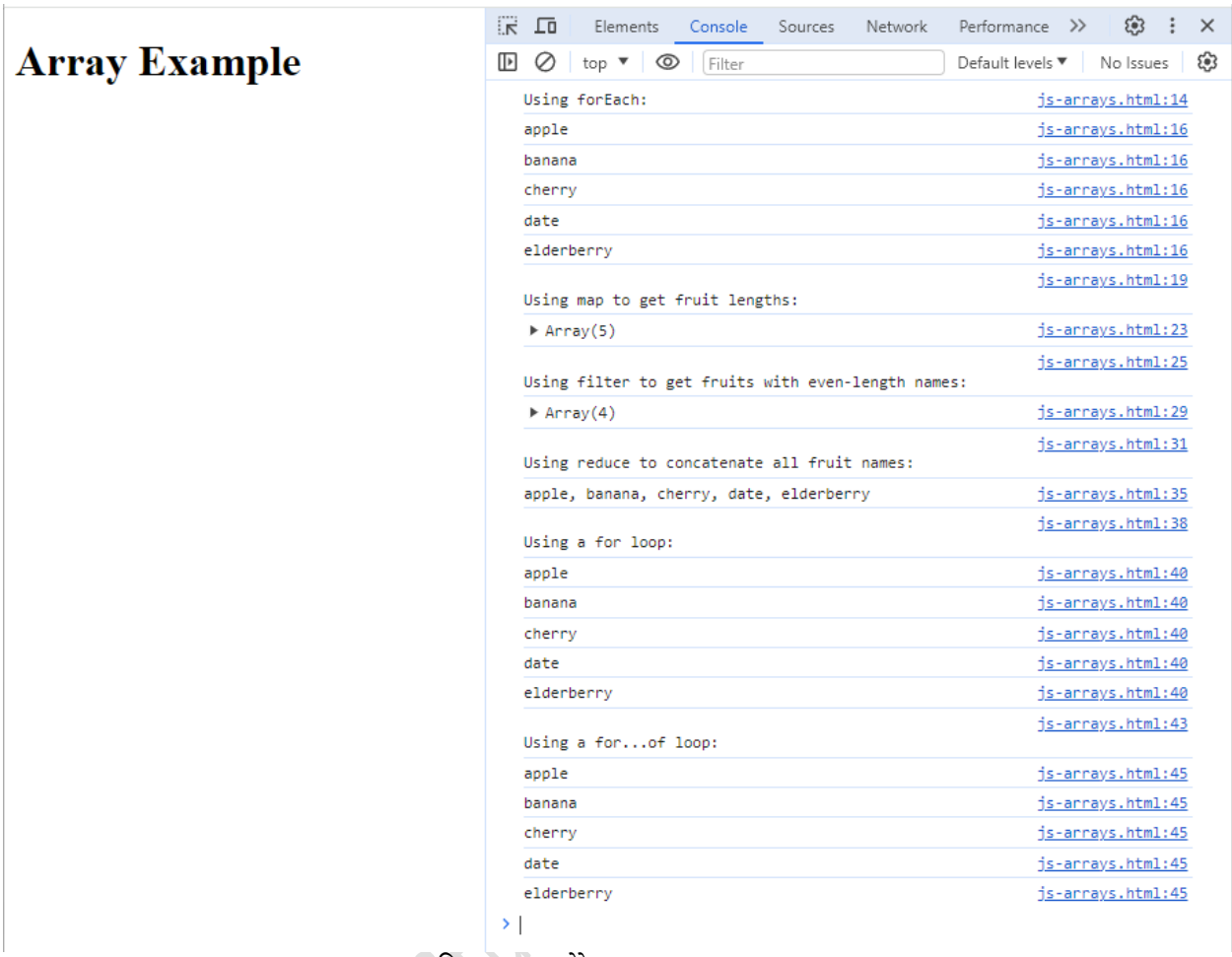
```
javascript
let fruits = ["apple", "banana", "cherry"];
for (let fruit of fruits) {
   console.log(fruit);
}
// आउटपुट— "apple", "banana", "cherry"
```

क्या आप जानते हैं?
ऐरे और लूप्स, जावास्क्रिप्ट (JavaScript) में डेटा संग्रहों (data collections) के साथ कार्य करने के लिए एक बहुउपयोगी और दक्ष तरीका प्रदान करते हैं।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Array Example</title>
</head>
<body>
    <h1>Array Example</h1>
    <script>
        // Creating an array
        let fruits = ["apple", "banana", "cherry", "date", "elderberry"];
        // 1. Using forEach to log each fruit
        console.log("Using forEach:");
        fruits.forEach(function (fruit) {
            console.log(fruit);
        });
        // 2. Using map to create a new array with fruit lengths
        console.log("\nUsing map to get fruit lengths:");
        let fruitLengths = fruits.map(function (fruit) {
            return fruit.length;
        });
        console.log(fruitLengths);
        // 3. Using filter to get fruits with even-length names
        console.log("\nUsing filter to get fruits with even-length names:");
        let evenLengthFruits = fruits.filter(function (fruit) {
            return fruit.length % 2 === 0;
        });
        console.log(evenLengthFruits);
        // 4. Using reduce to concatenate all fruit names
        console.log("\nUsing reduce to concatenate all fruit names:");
        let allFruits = fruits.reduce(function (accumulator, fruit) {
            return accumulator + ", " + fruit;
        });
        console.log(allFruits);
        // Using array loops
        // 5. Using a for loop to log each fruit
        console.log("\nUsing a for loop:");
        for (let i = 0; i <fruits.length; i++) {
            console.log(fruits[i]);
        }
        // 6. Using a for...of loop to log each fruit
        console.log("\nUsing a for...of loop:");
        for (let fruit of fruits) {
            console.log(fruit);
        }
    </script>
</body>
</html>
```

चित्र 15.2— ऐरे का उदाहरण

आउटपुट



चित्र 15.3— ऐरे उदाहरण का आउटपुट

कोड व्याख्या (Code Explanation)

- यह पंक्ति fruits नामक एक ऐरे को घोषित करती है और उसे फलों के नामों की सूची से आरंभ करती है।
- इस खंड में ऐरे विधियों के उपयोग को दर्शाया गया है—
  - forEach() का उपयोग fruits ऐरे के प्रत्येक तत्व (fruit) पर इटरेशन के लिए किया गया है।
  - प्रत्येक फल पर यह प्रदान की गई फ़ंक्शन को निष्पादित करता है, जो फल का नाम कंसोल में प्रदर्शित करता है।
- यहाँ हम map() विधि का उपयोग करके fruitLengths नामक एक नया ऐरे बनाते हैं—
  - map() fruits ऐरे के प्रत्येक तत्व पर इटरेशन करता है।
  - प्रत्येक फल के लिए यह एक फ़ंक्शन लागू करता है, जो फल के नाम की लंबाई की गणना करता है।
  - परिणामी ऐरे में सभी फलों के नामों की लंबाइयाँ सम्मिलित होती हैं।
- हम filter() विधि का उपयोग करके evenLengthFruits नामक एक नया ऐरे बनाते हैं—
  - filter() fruits ऐरे के प्रत्येक तत्व पर इटरेशन करता है।
  - प्रत्येक फल के लिए यह एक फ़ंक्शन लागू करता है, जो जांचता है कि क्या फल के नाम की लंबाई सम (even) है।

- परिणामी ऐरे में केवल वे फल होते हैं जिनके नाम की लंबाई सम होती है।

- हम reduce() विधि का उपयोग करके सभी फलों के नामों को जोड़कर allFruits नामक एक एकल स्ट्रिंग बनाते हैं—
  - reduce() fruits ऐरे के प्रत्येक तत्व पर इटरेशन करता है।
  - यह एक प्रदान की गई फ़ंक्शन लागू करता है, जो एक संचित मान (accumulator) — प्रारंभ में एक रिक्त स्ट्रिंग — को प्रत्येक फल के नाम से जोड़ता है।
  - अंतिम परिणाम एक एकल स्ट्रिंग होता है जिसमें सभी फल के नाम कॉमा द्वारा विभाजित होते हैं।
- यह भाग for लूप के उपयोग द्वारा fruits ऐरे के माध्यम से इटरेशन को प्रदर्शित करता है—
  - लूप i को 0 से आरंभ करता है और जब तक i fruits की लंबाई से कम रहता है, तब तक चलता है।
  - प्रत्येक पुनरावृत्ति में यह वर्तमान फल को उसके अनुक्रमांक i द्वारा कंसोल में प्रदर्शित करता है।
- अंततः, हम for...of लूप का उपयोग करते हैं ताकि forEach() विधि के समान ही इटरेशन प्राप्त किया जा सके—
  - यह सीधे fruits ऐरे के प्रत्येक तत्व (fruit) के माध्यम से इटरेशन करता है।
  - प्रत्येक पुनरावृत्ति में यह फल को कंसोल में प्रदर्शित करता है।

यह उदाहरण ऐरे के साथ कार्य करने की विभिन्न तकनीकों, डेटा संसाधन (manipulation) के लिए ऐरे विधियों, तथा ऐरे तत्वों पर इटरेशन के लिए विभिन्न प्रकार के लूप्स को प्रदर्शित करता है।

15.4 फ़ंक्शन्स (Functions)

जावास्क्रिप्ट में फ़ंक्शन्स पुनः उपयोग योग्य कोड के खंड (code blocks) होते हैं। इन्हें किसी विशिष्ट कार्य को निष्पादित करने के लिए डिज़ाइन किया गया है। वे जावास्क्रिप्ट प्रोग्रामिंग का एक मूलभूत हिस्सा हैं और कोड को संरचित व व्यवस्थित करने में महत्त्वपूर्ण भूमिका निभाते हैं। फ़ंक्शन्स इनपुट (inputs) ले सकते हैं, संचालन (operations) कर सकते हैं, और आउटपुट (outputs) प्रदान कर सकते हैं। नीचे फ़ंक्शन्स का विस्तृत विवरण तथा उन्हें उपयोग करने का तरीका दिया गया है—

15.4.1 फ़ंक्शन घोषणा (Function Declaration)

किसी फ़ंक्शन को घोषित करने के लिए function कीवर्ड का उपयोग किया जाता है, इसके बाद फ़ंक्शन का नाम, एक जोड़ी गोल कोष्ठक () और एक जोड़ी घुंघराले कोष्ठक {} लिखे जाते हैं, जिनके अंदर फ़ंक्शन का कोड ब्लॉक होता है।

javascript

```
function greet(name) {
  console.log("Hello, " + name + "!");
}
```

- function वह कीवर्ड है जिसका उपयोग किसी फ़ंक्शन को घोषित करने के लिए किया जाता है।
- greet फ़ंक्शन का नाम है।
- (name) एक पैरामीटर name को परिभाषित करता है। पैरामीटर वे चरों (variables) होते हैं जो फ़ंक्शन को दिए गए मानों को धारण करते हैं।
- { ... } में वह कोड होता है जिसे फ़ंक्शन निष्पादित करता है।

15.5 फ़ंक्शन आह्वान (Function Invocation – Calling a Function)

किसी फ़ंक्शन को निष्पादित करने के लिए, आप उसके नाम के बाद कोष्ठक लगाकर उसे **"कॉल" (call)** करते हैं और आप फ़ंक्शन के पैरामीटरों को मान (arguments) प्रदान कर सकते हैं।

greet("Alice"); // greet फ़ंक्शन को "Alice" आर्गुमेंट के साथ कॉल करना

- greet("Alice") एक फ़ंक्शन आह्वान (function invocation) है। यहाँ "Alice" एक आर्गुमेंट (argument) है जो name पैरामीटर को प्रदान किया गया है।

फ़ंक्शन रिटर्न (Function Return)— फ़ंक्शन return कथन का उपयोग करके मान वापस कर सकते हैं। किसी फ़ंक्शन में शून्य या एक return कथन हो सकता है। जब कोई फ़ंक्शन return निष्पादित करता है, तो वह तुरंत समाप्त हो जाता है और निर्दिष्ट मान को कॉल करने वाले को वापस कर देता है।

javascript

```
function add(a, b) {
  return a + b;
}
let result = add(5, 3); // result का मान 8 होगा
```

- add फ़ंक्शन में return a + b; कथन a और b का योग वापस करता है।
- add(5, 3) का परिणाम 8 होता है, जो कि result नामक चर (variable) में स्टोर होता है।

फ़ंक्शन स्कोप (Function Scope)— किसी फ़ंक्शन के अंदर घोषित चर (variables), उस फ़ंक्शन के लिए **स्थानीय (local)** होते हैं, अर्थात् वे केवल उसी फ़ंक्शन के अंदर पहुँच योग्य होते हैं। इस अवधारणा को **"फ़ंक्शन स्कोप"** कहा जाता है।

javascript

```
function multiply(x, y) {
  let product = x * y; // product एक स्थानीय चर है
  return product;
}
console.log(product); // त्रुटि— product फ़ंक्शन के बाहर परिभाषित नहीं है
```

- product नामक चर को multiply फ़ंक्शन के अंदर घोषित किया गया है और यह केवल उसी फ़ंक्शन के अंदर उपलब्ध है।

फ़ंक्शन पैरामीटर और आर्गुमेंट्स के बीच भिन्नताएँ

पैरामीटर (Parameters)	**आर्गुमेंट्स (Arguments)**
फ़ंक्शन परिभाषा में प्लेसहोल्डर (स्थानीक मानधारक) जो फ़ंक्शन को दिए जाने वाले मानों का प्रतिनिधित्व करते हैं।	फ़ंक्शन को कॉल करते समय प्रदान किए गए वास्तविक मान या अभिव्यक्तियाँ।
फ़ंक्शन के हस्ताक्षर या घोषणा में परिभाषित किए जाते हैं।	फ़ंक्शन को आह्वान (call) करते समय दिए जाते हैं।
फ़ंक्शन के स्कोप के अंदर स्थानीय वेरिएबल की तरह कार्य करते हैं।	फ़ंक्शन द्वारा प्रक्रिया किए जाने वाले डेटा को प्रदान करते हैं।
इनकी संख्या शून्य या अधिक हो सकती है।	फ़ंक्शन को कॉल करते समय उनकी संख्या पैरामीटरों से मेल खानी चाहिए।
इनके नाम होते हैं जो फ़ंक्शन स्कोप में उपयोग किए जाते हैं।	इनका कोई नाम फ़ंक्शन स्कोप के अंदर नहीं होता।

पैरामीटर (Parameters)	आर्गुमेंट्स (Arguments)
डिफ़ॉल्ट मान दिए जा सकते हैं ताकि अनुपस्थित आर्गुमेंट्स को नियंत्रित किया जा सके।	आवश्यक पैरामीटर के लिए आर्गुमेंट अनुपस्थित होने पर उनका मान undefined होता है।
इन्हें फ़ंक्शन के अंदर संशोधित किया जा सकता है और इससे मूल आर्गुमेंट्स प्रभावित नहीं होते।	इन्हें सीधे फ़ंक्शन के अंदर संशोधित नहीं किया जा सकता।

तालिका 15.1— फ़ंक्शन पैरामीटर और आर्गुमेंट्स के बीच भिन्नताएँ

15.6 अंतर्निर्मित फ़ंक्शन (Built-in Functions)

अंतर्निर्मित फ़ंक्शन, जिन्हें **मूल फ़ंक्शन (native functions)** या **मानक लाइब्रेरी फ़ंक्शन (standard library functions)** भी कहा जाता है, वे फ़ंक्शन होते हैं जो JavaScript में पूर्वनिर्धारित रूप से उपलब्ध होते हैं। ये फ़ंक्शन विभिन्न प्रयोजनों की पूर्ति करते हैं और डेटा के साथ कार्य करने, गणनाएँ करने, स्ट्रिंग को संशोधित करने, तिथि (date) संभालने, DOM (Document Object Model) से अंतःक्रिया करने आदि जैसी क्षमताएँ प्रदान करते हैं। नीचे JavaScript में उपयोग होने वाले अंतर्निर्मित फ़ंक्शन की कुछ सामान्य श्रेणियाँ **चित्र 15.4** में दी गई हैं—

गणितीय फ़ंक्शन (Math Functions)—

- Math.sqrt(x) — x का वर्गमूल (square root) लौटाता है।
- Math.pow(x, y) — x को y की घात (power) में लौटाता है।
- Math.random() — 0 (सम्मिलित) और 1 (बहिर्गामी) के बीच एक यादृच्छिक संख्या उत्पन्न करता है।

स्ट्रिंग फ़ंक्शन (String Functions)—

- str.length — स्ट्रिंग की लंबाई देता है।
- str.toUpperCase() — स्ट्रिंग को बड़े अक्षरों (uppercase) में बदलता है।
- str.toLowerCase() — स्ट्रिंग को छोटे अक्षरों (lowercase) में परिवर्तित करता है।
- str.indexOf(substring) — स्ट्रिंग में किसी उपस्ट्रिंग (substring) के पहले स्थान का सूचकांक प्रदान करता है।

ऐरे फ़ंक्शन (Array Functions)—

- array.length — ऐरे में उपस्थित तत्वों की संख्या लौटाता है।
- array.push(element) — एक तत्व को ऐरे के अंत में जोड़ता है।
- array.pop() — ऐरे से अंतिम तत्व को हटाता है और लौटाता है।
- array.join(separator) — ऐरे को स्ट्रिंग में परिवर्तित करता है, जहाँ तत्वों को एक नियत विभाजक (separator) से जोड़ा जाता है।

तिथि फ़ंक्शन (Date Functions)—

- new Date() — वर्तमान तिथि और समय का प्रतिनिधित्व करने वाला नया Date ऑब्जेक्ट बनाता है।
- date.getFullYear() — Date ऑब्जेक्ट का वर्ष लौटाता है।
- date.getMonth() — Date ऑब्जेक्ट का माह (0–11) लौटाता है।
- date.getDate() — Date ऑब्जेक्ट का दिन (1–31) लौटाता है।

DOM हेरफेर फ़ंक्शन (DOM Manipulation Functions)—

- document.getElementById(id) — निर्दिष्ट id वाले HTML तत्व को प्राप्त करता है।

- document.querySelector(selector) — चयनकर्ता (selector) से मेल खाने वाला पहला HTML तत्व प्राप्त करता है।
- element.addEventListener(event, handler) — किसी HTML तत्व पर घटना-संवहनकर्ता (event handler) संलग्न करता है।

प्रकार रूपांतरण फ़ंक्शन (Type Conversion Functions)—

- parseInt(string) — स्ट्रिंग को पार्स कर पूर्णांक (integer) लौटाता है।
- parseFloat(string) — स्ट्रिंग को पार्स कर फ्लोटिंग-पॉइंट संख्या लौटाता है।
- String(value) — किसी मान को स्ट्रिंग में बदलता है।

रेगुलर एक्सप्रेशन फ़ंक्शन (Regular Expression Functions)—

- regex.test(string) — जांचता है कि कोई स्ट्रिंग किसी रेगुलर एक्सप्रेशन से मेल खाती है या नहीं।
- string.match(regex) — स्ट्रिंग में किसी रेगुलर एक्सप्रेशन पैटर्न को खोजता है और परिणाम लौटाता है।

वैश्विक फ़ंक्शन (Global Functions)—

- isNaN(value) — जांचता है कि कोई मान NaN (Not-a-Number) है या नहीं।
- parseInt(string, radix) — किसी स्ट्रिंग को एक निर्दिष्ट रेडिक्स (आधार) के साथ पूर्णांक के रूप में पार्स करता है।

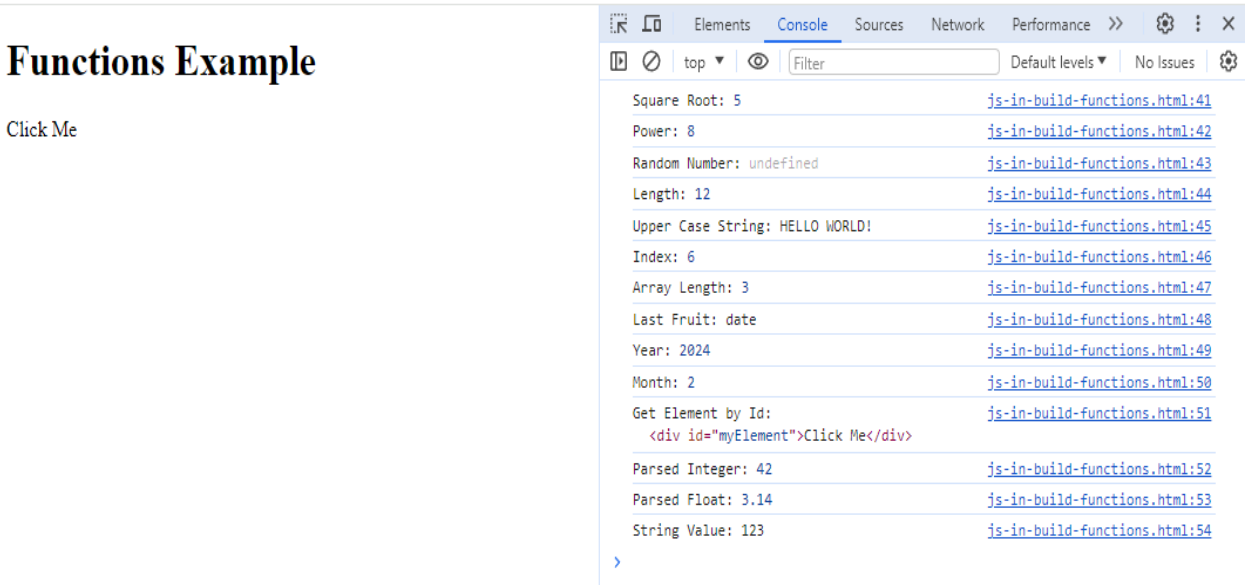
ये JavaScript में उपलब्ध कई अंतर्निर्मित फ़ंक्शनों में से कुछ सामान्य उदाहरण हैं। इनका प्रयोग कर आप अपने कोड को सरल बना सकते हैं तथा सामान्य कार्यों को अधिक दक्षतापूर्वक संपन्न कर सकते हैं।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript in-built Functions</title>
</head>
<body>
    <h1>Functions Example</h1>
    <!-- Add an HTML element with the ID "myElement" -->
    <div id="myElement">Click Me</div>
```

```
    <script>
        // Wrap your JavaScript code in the DOMContentLoaded event listener
        document.addEventListener("DOMContentLoaded", function() {
            // Math Functions
            let squareRoot = Math.sqrt(25)
            let power = Math.pow(2, 3)
            let randomNum = Math.randomNum
            // String Functions
            let str = "Hello World!"
            let length = str.length
            let uppercaseStr = str.toUpperCase()
            let index = str.indexOf("World")
            // Array Functions
            let fruits = ["apple", "banana", "cherry"]
            let arrayLength = fruits.length
            fruits.push("date")
            let lastFruit = fruits.pop()
            // Date Functions
            let currentDate = new Date()
            let year = currentDate.getFullYear()
            let month = currentDate.getMonth()
            // DOM Manipulation Functions
            let elementById = document.getElementById("myElement")
            // Type Conversion Functions
            let parsedInt = parseInt("42")
            let parsedFloat = parseFloat("3.14")
            let stringValue = String(123)
            // Print values to the console
            console.log("Square Root:", squareRoot)
            console.log("Power:", power)
            console.log("Random Number:", randomNum)
            console.log("Length:", length)
            console.log("Upper Case String:", uppercaseStr)
            console.log("Index:", index)
            console.log("Array Length:", arrayLength)
            console.log("Last Fruit:", lastFruit)
            console.log("Year:", year)
            console.log("Month:", month)
            console.log("Get Element by Id:", elementById)
            console.log("Parsed Integer:", parsedInt)
            console.log("Parsed Float:", parsedFloat)
            console.log("String Value:", stringValue)
        });
    </script>
</body>
</html>
```

चित्र 15.4— JavaScript में अंतर्निर्मित फ़ंक्शन

आउटपुट



चित्र 15.4— JavaScript में अंतर्निर्मित फ़ंक्शन का आउटपुट

चित्र 15.5 में प्रयोक्ता द्वारा परिभाषित फ़ंक्शन (User Defined Function) का उदाहरण

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Example of User Defined Functions</title>
</head>
<body>
    <h1>User-Defined Functions Example</h1>

    <script>
        // User-Defined function to calculate the square of a number
        function calculateSquare(number) {
            return number * number
        }

        // Call the User-Defined function and store the result in a variable
        let result = calculateSquare(5)

        // Display the result
        console.log("The square of 5 is: ", result)
    </script>
</body>
</html>
```

चित्र 15.6— प्रयोक्ता परिभाषित फ़ंक्शन का आउटपुट

सारांश

- जावास्क्रिप्ट (JavaScript) में ऐरे (Arrays) मौलिक डेटा संरचनाएँ हैं, जो आंकड़ों के समूह को स्टोर और प्रबंधित करने की अनुमति देती हैं।
- ऐरे को ऐरे लिटरल (array literals), ऐरे कंस्ट्रक्टर (Array constructor), या तरीकों, लूप और स्प्रेड ऑपरेटर (spread operators) का उपयोग करके गतिशील रूप से तैयार किया जा सकता है।
- push, pop, shift, unshift, concat, slice, forEach, map, और filter जैसी ऐरे विधियाँ (array methods) ऐरे के हेरफेर के लिए प्रभावशाली उपकरण प्रदान करती हैं।
- for, forEach और for...of जैसे लूप ऐरे तत्वों के ऊपर आवर्तन (iterate) करने और उन पर क्रियाएँ निष्पादित करने के लिए उपयोग किए जाते हैं।
- फ़ंक्शन (Functions) पुनः उपयोग योग्य कोड के खंड होते हैं, जिन्हें पैरामीटर और कोड ब्लॉक के साथ परिभाषित किया जाता है, और जिन्हें आर्ग्युमेंट्स (arguments) के साथ बुलाया (invoke) जा सकता है।
- फ़ंक्शन return स्टेटमेंट का उपयोग करके मान वापस कर सकते हैं, और फ़ंक्शन के अंदर घोषित चर (variables) उस फ़ंक्शन के स्कोप के लिए "स्थानीय" (local) होते हैं।
- फ़ंक्शन पैरामीटर और आर्ग्युमेंट्स के बीच अंतर को समझना प्रभावी फ़ंक्शन उपयोग के लिए अत्यंत आवश्यक है।
- जावास्क्रिप्ट में निर्मित (built-in) फ़ंक्शन आवश्यक क्षमताएँ प्रदान करते हैं, जिनमें गणना (math), स्ट्रिंग हेरफेर (string manipulation), ऐरे संचालन (array operations), दिनांक प्रबंधन (date handling) और DOM हेरफेर शामिल हैं।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. जावास्क्रिप्ट में ऐरे बनाने का एक सामान्य तरीका क्या है?
   (क) कोष्ठक () का उपयोग करके
   (ख) वर्ग कोष्ठक [] का उपयोग करके
   (ग) कोण कोष्ठक <> का उपयोग करके
   (घ) घुंघराले कोष्ठक {} का उपयोग करके

2. ऐरे के अंत में तत्व जोड़ने के लिए कौन-सी विधि प्रयुक्त होती है?
   (क) concat()
   (ख) push()
   (ग) unshift()
   (घ) pop()

3. आप ऐरे से अंतिम तत्व को कैसे हटा सकते हैं और उसका मान प्राप्त कर सकते हैं?
   (क) pop()
   (ख) shift()
   (ग) remove()
   (घ) splice()

4. जावास्क्रिप्ट में ऐरे के तत्वों पर आवर्तन के लिए सामान्यतः कौन-सा लूप उपयोग होता है?
   (क) for-in लूप
   (ख) while लूप
   (ग) for लूप
   (घ) forEach लूप

5. जावास्क्रिप्ट ऐरे में map() विधि का उद्देश्य क्या है?
   (क) यह मूल ऐरे को संशोधित करता है
   (ख) यह रूपांतरित तत्वों के साथ एक नया ऐरे बनाता है
   (ग) यह ऐरे से विशेष तत्वों को हटाता है
   (घ) यह ऐरे के क्रम को उलट देता है

6. जावास्क्रिप्ट फ़ंक्शन के अंदर घोषित वेरिएबल किस प्रकार के स्कोप में आते हैं?
   (क) वैश्विक
   (ख) फ़ंक्शन के लिए स्थानीय
   (ग) सम्पूर्ण डॉक्युमेंट में
   (घ) केवल ऑब्जेक्ट्स में

7. जावास्क्रिप्ट में फ़ंक्शन घोषणा का मुख्य उद्देश्य क्या है?
   (क) अन्य फ़ंक्शन को कॉल करना
   (ख) वेरिएबल को घोषित करना
   (ग) किसी विशेष कार्य को निष्पादित करना
   (घ) ऐरे बनाना

8. जावास्क्रिप्ट फ़ंक्शन को आर्ग्युमेंट्स के साथ कॉल करने का सही तरीका क्या है?
   (क) myFunction(arguments)
   (ख) myFunction.arguments()
   (ग) myFunction(arguments)
   (घ) myFunction(arguments, values)

9. जावास्क्रिप्ट में फ़ंक्शन को घोषित करने के लिए कौन-सा कीवर्ड प्रयुक्त होता है?
   (क) func
   (ख) declare
   (ग) function
   (घ) def

10. जावास्क्रिप्ट फ़ंक्शन में return स्टेटमेंट का उपयोग कैसे किया जाता है?
    (क) यह फ़ंक्शन को समाप्त करने के लिए उपयोग होता है
    (ख) यह फ़ंक्शन के नाम को परिभाषित करने के लिए उपयोग होता है

(ग) यह फ़ंक्शन से मान वापस करने के लिए उपयोग होता है

(घ) यह लूप बनाने के लिए उपयोग होता है

ख. रिक्त स्थान भरिए

1. _____ विधि का उपयोग ऐरे से अंतिम तत्व को हटाने के लिए किया जाता है।
2. फ़ंक्शन _____ कोड के खंड होते हैं जिन्हें विशिष्ट कार्यों को निष्पादित करने के लिए पुनः उपयोग किया जा सकता है।
3. किसी फ़ंक्शन को निष्पादित करने के लिए, आपको उसके नाम के बाद _____ की एक जोड़ी का उपयोग करना होता है।
4. फ़ंक्शन में _____ स्टेटमेंट आपको फ़ंक्शन से मान वापस करने की अनुमति देता है।
5. जावास्क्रिप्ट में पहले से परिभाषित फ़ंक्शन सामान्यतः _____ फ़ंक्शन कहलाते हैं।
6. _____ फ़ंक्शन का उपयोग 0 (समावेशी) और 1 (बहिर्भुक्त) के बीच एक रैंडम संख्या उत्पन्न करने के लिए किया जाता है।
7. _____ फ़ंक्शन किसी मान को स्ट्रिंग में परिवर्तित करता है।
8. _____ विधि एक ऐरे के अंत में तत्व जोड़ती है और ऐरे की नई लंबाई लौटाती है।
9. _____ विधि ऐरे से पहले तत्व को हटाती है और उस तत्व को लौटाती है।
10. _____ जावास्क्रिप्ट में एक विशेष कार्य को निष्पादित करने के लिए डिज़ाइन किया गया होता है।

ग. सही या गलत बताइए

1. ऐरे का उपयोग डेटा के समूहों को स्टोर और प्रबंधित करने के लिए किया जाता है।
2. आप जावास्क्रिप्ट में किसी ऐरे के विशिष्ट इंडेक्स पर मान असाइन करके एक रिक्त ऐरे बना सकते हैं।
3. Array कंस्ट्रक्टर का उपयोग करके मानों के साथ ऐरे बनाया जा सकता है।
4. पैरामीटर वे मान होते हैं जो फ़ंक्शन को कॉल करते समय दिए जाते हैं।
5. जावास्क्रिप्ट में एक फ़ंक्शन में एक से अधिक return स्टेटमेंट हो सकते हैं।
6. एक फ़ंक्शन सीधे उसमें दिए गए आर्ग्युमेंट्स को संशोधित कर सकता है।
7. push() विधि ऐरे की शुरुआत में तत्व जोड़ती है।
8. shift() विधि ऐरे से अंतिम तत्व को हटाती है।
9. जावास्क्रिप्ट में फ़ंक्शन पुनः उपयोग योग्य कोड के खंड होते हैं।
10. आप Array.from() विधि का उपयोग करके ऐरे को घोषित और आरंभ कर सकते हैं।

घ. लघु उत्तर प्रश्न

1. जावास्क्रिप्ट में ऐरे क्या है, और यह क्यों उपयोगी है?
2. जावास्क्रिप्ट में ऐरे लिटरल का उपयोग करके ऐरे कैसे बनाया जा सकता है?
3. push() विधि का उपयोग करके ऐरे के अंत में तत्व कैसे जोड़े जाते हैं, समझाइए।
4. pop() विधि क्या करती है, और इसका ऐरे के साथ उपयोग कैसे किया जाता है?
5. जावास्क्रिप्ट में "फ़ंक्शन स्कोप" की अवधारणा का वर्णन कीजिए।
6. फ़ंक्शन पैरामीटर और आर्ग्युमेंट्स के बीच क्या अंतर है?
7. जावास्क्रिप्ट में फ़ंक्शन को कैसे घोषित किया जाता है, और उसकी संरचना क्या होती है?
8. किसी फ़ंक्शन में return स्टेटमेंट का उद्देश्य क्या है, और यह कैसे कार्य करता है?
9. जावास्क्रिप्ट में किसी निर्मित (built-in) फ़ंक्शन का उदाहरण दीजिए और उसके उद्देश्य को समझाइए।
10. जावास्क्रिप्ट में निर्मित (built-in) फ़ंक्शन क्या होते हैं?

सत्र 16

स्ट्रिंग में रूपांतरण (String Manipulation)

शब्दों से प्रेम करने वाले अमन नामक बच्चे ने "स्ट्रिंग में रूपांतरण" (String Manipulation) की खोज की, जो उसके लिए एक शब्दों का खेल जैसा था। स्ट्रिंग में रूपांतरण का अर्थ होता है शब्दों और वाक्यों को रोचक तरीके से बदलना, जैसे किसी पहेली में अक्षरों को मिलाना। उसे दो मज़ेदार तरकीबें सीखने को मिलीं—स्ट्रिंग को जोड़ना, जिससे वह शब्दों को मिलाकर नए शब्द बना सकता था, और अक्षरों को बदलना, जिससे वह शब्दों के अक्षरों को बदल या पुनः व्यवस्थित कर सकता था। इन तरकीबों के साथ, उसने मज़ेदार कहानियाँ बनाईं और अपनी स्वयं की एक गुप्त भाषा भी बना डाली, जिससे वह स्वयं को शब्दों का जादूगर समझने लगा, जिसके पास एक जादुई शब्दों का उपकरण-बॉक्स था। यह शब्दों के साथ खेलने का एक रचनात्मक और आनंददायक तरीका था—जैसे पहेली हल करना या मज़ेदार कहानियाँ लिखना। जैसा कि चित्र 16.1 में दिखाया गया है।

चित्र 16.1— अमन स्ट्रिंग में रूपांतरण का उपयोग करते हुए

इस अध्याय में, आप स्ट्रिंग में रूपांतरण (String Manipulation), दिनांक और समय (Dates and Time), तथा फ़ॉर्म्स (Forms) के बारे में सीखेंगे।

16.1 स्ट्रिंग में रूपांतरण

जावास्क्रिप्ट में स्ट्रिंग रूपांतरण का अर्थ है—स्ट्रिंग पर विभिन्न क्रियाएँ करना ताकि उसमें से जानकारी निकाली जा सके या उसमें बदलाव किया जा सके। नीचे कुछ सामान्य स्ट्रिंग रूपांतरण तकनीकों और उनके उदाहरणों को चित्र 16.2 में दर्शाया गया है—

Concatenation (संकलन)— दो या अधिक स्ट्रिंग को जोड़ना

```
javascript
let str1 = "Hello";
let str2 = "World";
let combinedStr = str1 + " " + str2; // "Hello World"
```

String Length (स्ट्रिंग की लंबाई)— स्ट्रिंग की लंबाई निकालना

```
javascript
let text = "This is a sample text";
let length = text.length; // 21
```

Substring (उपस्ट्रिंग)— स्ट्रिंग के किसी भाग को निकालना

```
javascript
let text = "Hello, World";
let substring = text.substring(0, 5); // "Hello"
```

String Case (अक्षरों का रूपांतरण)— स्ट्रिंग को बड़े या छोटे अक्षरों में बदलना

javascript

```
let str = "JavaScript";
let upperCaseStr = str.toUpperCase(); // "JAVASCRIPT"
let lowerCaseStr = str.toLowerCase(); // "javascript"
```

String Search (स्ट्रिंग खोज)— स्ट्रिंग में किसी उपस्ट्रिंग की स्थिति पता करना

javascript

```
let text = "This is a sample text";
let position = text.indexOf("sample"); // 10
```

Replacing Substrings (उपस्ट्रिंग बदलना)— स्ट्रिंग के अंदर किसी उपस्ट्रिंग को बदलना

javascript

```
let text = "Hello, World";
let replacedText = text.replace("World", "Universe"); // "Hello, Universe"
```

Splitting Strings (स्ट्रिंग विभाजन)— किसी स्ट्रिंग को एक सीमांकक (delimiter) के आधार पर उपस्ट्रिंग्स के ऐरे में बाँटना

javascript

```
let csvData = "John,Doe,30";
let dataArray = csvDatक.split(","); // ["John", "Doe", "30"]
```

String Conversion (स्ट्रिंग में रूपांतरण)— अन्य डेटा प्रकारों को स्ट्रिंग में बदलना

javascript

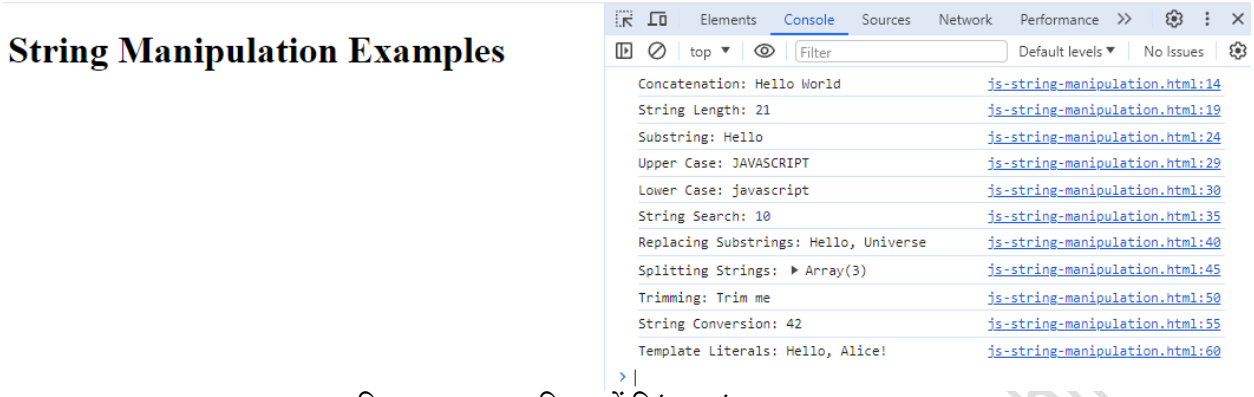
```
let num = 42;
let strNum = String(num); // "42"
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>String Manipulation Examples</title>
</head>
<body>
<h1>String Manipulation Examples</h1>
<script>
        // 1. Concatenation
        let str1 = "Hello";
        let str2 = "World";
        let combinedStr = str1 + " " + str2;
        console.log("Concatenation:", combinedStr);

        // 2. String Length
        let text = "This is a sample text";
        let length = text.length;
        console.log("String Length:", length);
```

```
        // 3. Substring
        let text2 = "Hello, World";
        let substring = text2.substring(0, 5);
        console.log("Substring:", substring);
        // 4. String Case
        let str3 = "JavaScript";
        let upperCaseStr = str3.toUpperCase();
        let lowerCaseStr = str3.toLowerCase();
        console.log("Upper Case:", upperCaseStr);
        console.log("Lower Case:", lowerCaseStr);

        // 5. String Search
        let text3 = "This is a sample text";
        let position = text3.indexOf("sample");
        console.log("String Search:", position);

        // 6. Replacing Substrings
        let text4 = "Hello, World";
        let replacedText = text4.replace("World", "Universe");
        console.log("Replacing Substrings:", replacedText);

        // 7. Splitting Strings
        let csvData = "John,Doe,30";
        let dataArray = csvData.split(",");
        console.log("Splitting Strings:", dataArray);

        // 8. Trimming
        let text5 = "   Trim me   ";
        let trimmedText = text5.trim();
        console.log("Trimming:", trimmedText);

        // 9. String Conversion
        let num = 42;
        let strNum = String(num);
        console.log("String Conversion:", strNum);

        // 10. Template Literals
        let name = "Alice";
        let greeting = `Hello, ${name}!`;
console.log("Template Literals:", greeting);
</script>
</body>
```

चित्र 16.2— जावास्क्रिप्ट में स्ट्रिंग में रूपांतरण

आउटपुट



चित्र 16.3— जावास्क्रिप्ट में स्ट्रिंग रूपांतरण का आउटपुट

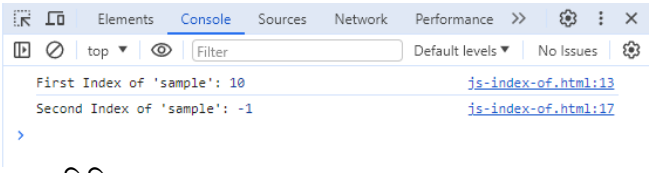
indexOf()— indexOf() विधि का उपयोग किसी स्ट्रिंग में किसी उपस्ट्रिंग की पहली उपस्थिति की स्थिति (इंडेक्स) जानने के लिए किया जाता है, जैसा कि चित्र 16.4 में दर्शाया गया है।

उदाहरण

```
html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>indexOf Example</title>
</head>
<body>
<h1>indexOf Example</h1>
<script>
    let text = "This is a sample text. Sample text is here.";
    // "sample" की पहली उपस्थिति की स्थिति खोजें
    let firstIndex = text.indexOf("sample");
    console.log("First Index of 'sample':", firstIndex);
    // "sample" की दूसरी उपस्थिति की स्थिति खोजें
    let secondIndex = text.indexOf("sample", firstIndex + 1);
    console.log("Second Index of 'sample':", secondIndex);
</script>
</body>
</html>
```

आउटपुट

indexOf Example



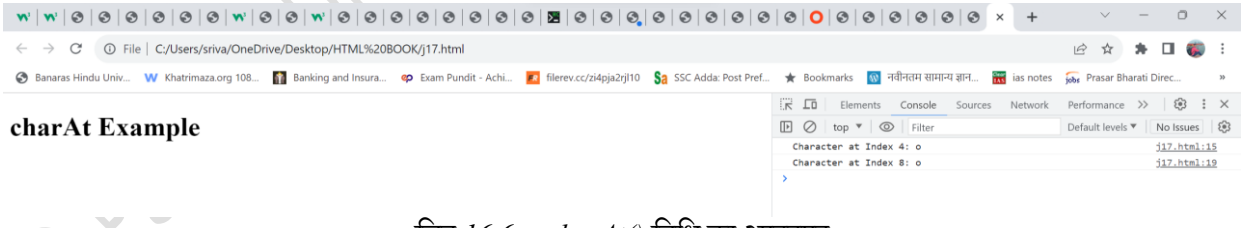
चित्र 16.4— indexOf() विधि का आउटपुट

charAt()— charAt() विधि का उपयोग किसी स्ट्रिंग में किसी विशेष इंडेक्स पर स्थित अक्षर को प्राप्त करने के लिए किया जाता है, जैसा कि चित्र 16.5 में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>charAt Example</title>
</head>
<body>
    <h1>charAt Example</h1>

    <script>
        let text = "Hello, World";

        // Get the character at index 4
        let charAtIndex4 = text.charAt(4);
        console.log("Character at Index 4:", charAtIndex4);

        // Get the character at index 8
        let charAtIndex8 = text.charAt(8);
        console.log("Character at Index 8:", charAtIndex8);
    </script>
</body>
</html>
```

चित्र 16.5— charAt() विधि

आउटपुट



चित्र 16.6— charAt() विधि का आउटपुट

substr()—

JavaScript की substr() विधि स्ट्रिंग का कोई भाग निकालती है। यह विधि एक निर्दिष्ट स्थिति से शुरू होती है और निर्दिष्ट संख्या में अक्षरों को लौटाती है। यह मूल स्ट्रिंग को नहीं बदलती।

substr() विधि की संरचना—

string.substr(start, length)

जहाँ,

start— प्रारंभिक स्थिति (0 से प्रारंभ)

length— लौटाए जाने वाले अक्षरों की संख्या

उदाहरण—

javascript

```
"Hello, world!".substr(0, 5);     // "Hello"
"Hello, world!".substr(5);        // "world!"
"Hello, world!".substr(-6);       // "world!"
"Hello, world!".substr(-6, 5);    // "world"
```

split()—

split() विधि का उपयोग किसी स्ट्रिंग को उपस्ट्रिंग्स के ऐरे में बाँटने के लिए किया जाता है। यह दो वैकल्पिक पैरामीटर लेती है—पहला विभाजक (separator), और दूसरा अधिकतम तत्वों की संख्या।

javascript

```
var str = "The quick brown fox jumps over the lazy dog.";
var substrings = str.split(" ");
```

क्या आप जानते हैं?

join()—

join() विधि ऐरे के सभी तत्वों को एक स्ट्रिंग में जोड़ती है, जो एक विभाजक द्वारा अलग किए जाते हैं। इस विधि का उपयोग दो या अधिक स्ट्रिंग्स को एक में जोड़ने के लिए भी किया जा सकता है।

उदाहरण 1—

javascript

```
const str1 = "Hello";
const str2 = "World";
const separator = ", ";
const result = str1.concat(str2, separator);
console.log(result); // "Hello, World"
```

उदाहरण 2—

javascript

```
const arr = ["Hello", "World"];
const separator = ", ";
const result = arr.join(separator);
console.log(result); // "Hello, World"
```

अभ्यास16.1

- निम्नलिखित विधियों (methods) के उद्देश्य लिखिए—
  - o join()
  - o charAt()

- o indexOf()
- o substr()

16.2 तिथि और समय (Dates and Time)

JavaScript में तिथियाँ और समय Date ऑब्जेक्ट के माध्यम से निरूपित किए जाते हैं। Date ऑब्जेक्ट तिथि और समय की जानकारी प्रदान करता है तथा इसमें अनेक विधियाँ (methods) उपलब्ध होती हैं।

क्या आप जानते हैं?

JavaScript में तिथि EcmaScript युग (epoch) को परिभाषित करती है, जो 1 जनवरी 1970 UTC से मिलीसेकंड्स को निरूपित करती है।

नीचे JavaScript में तिथि और समय से संबंधित कुछ सामान्य कार्य दिए गए हैं—

Date ऑब्जेक्ट बनाना

आप किसी विशेष तिथि और समय को निरूपित करने के लिए Date ऑब्जेक्ट बना सकते हैं—

```
javascript
let currentDate = new Date(); // वर्तमान तिथि और समय
let specificDate = new Date("2023-10-05T12:00:00"); // विशेष तिथि और समय
```

Date कॉम्पोनेंट्स प्राप्त करना

आप Date ऑब्जेक्ट से वर्ष, माह, दिन आदि घटकों को निकाल सकते हैं—

```
javascript
let year = currentDate.getFullYear(); // वर्ष प्राप्त करें
let month = currentDate.getMonth(); // माह प्राप्त करें (0–11)
let day = currentDate.getDate(); // तिथि (1–31)
let hours = currentDate.getHours(); // घंटे (0–23)
let minutes = currentDate.getMinutes(); // मिनट (0–59)
let seconds = currentDate.getSeconds(); // सेकंड (0–59)
```

तिथि को फॉर्मेट करना

आप toLocaleDateString() जैसी विधियों या moment.js जैसी लाइब्रेरी का उपयोग करके तिथियों को स्ट्रिंग में फॉर्मेट कर सकते हैं—

```
javascript
let formattedDate = currentDate.toLocaleDateString("en-US"); // "MM/DD/YYYY" प्रारूप में
```

टाइमस्टैम्प के साथ कार्य करना

टाइमस्टैम्प 1 जनवरी 1970 (UTC) से मिलीसेकंड्स को निरूपित करता है—

```
javascript
let timestamp = Date.now(); // वर्तमान टाइमस्टैम्प
```

तिथियों की तुलना करना

आप यह जाँचने के लिए तिथियों की तुलना कर सकते हैं कि कोई तिथि दूसरी से पहले, बाद में या उसके बराबर है—

```
javascript
let date1 = new Date("2023-10-05");
let date2 = new Date("2023-10-10");
```

```
if (date1 < date2) {
  console.log("date1 is before date2");
} else if (date1 > date2) {
  console.log("date1 is after date2");
} else {
  console.log("date1 is equal to date2");
}
```

समय क्षेत्रों के साथ कार्य करना

JavaScript का Date ऑब्जेक्ट स्थानीय समय क्षेत्र (local time zone) के साथ कार्य करता है। आप मैन्युअल समायोजन या किसी लाइब्रेरी की सहायता से समय क्षेत्र परिवर्तित कर सकते हैं—

```
javascript
// विशेष समय क्षेत्र (जैसे UTC) के लिए समायोजन
currentDate.setHours(currentDate.getHours() - 4);
```

तिथि का सत्यापन (Date Validation)

आप यह सुनिश्चित करने के लिए प्रयोक्ता द्वारा दी गई तिथियों का सत्यापन कर सकते हैं कि वे वैध प्रारूप में हैं—

```
javascript
function isValidDate(dateStr) {
  return !isNaN(Date.parse(dateStr));
}
```

तिथि गणित (Date Arithmetic)

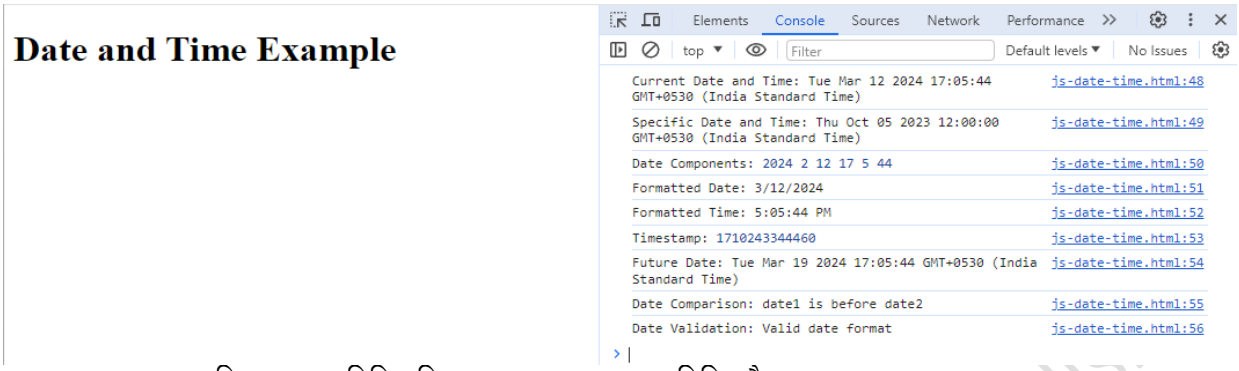
आप तिथि में दिनों को जोड़ना या घटाना जैसे कार्य कर सकते हैं—

```
javascript
let futureDate = new Date();
futureDate.setDate(currentDate.getDate() + 7); // वर्तमान तिथि में 7 दिन जोड़ें
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript Date and Time Example</title>
</head>
<body>
    <h1>Date and Time Example</h1>
    <script>
        // 1. Creating a Date Object
        let currentDate = new Date()
        let specificDate = new Date("2023-10-05T12:00:00")
        // 2. Getting Date Components
        let year = currentDate.getFullYear()
        let month = currentDate.getMonth()
        let day = currentDate.getDate()
        let hours = currentDate.getHours()
        let minutes = currentDate.getMinutes()
        let seconds = currentDate.getSeconds()
        // 3. Formatting Dates
        let formattedDate = currentDate.toLocaleDateString("en-US")
        let formattedTime = currentDate.toLocaleTimeString("en-US")
        // 4. Working with TimeStamps
        let timestamp = Date.now()
        // 5. Date Arithmatic
        let futureDate = new Date()
        futureDate.setDate(currentDate.getDate() + 7)
        // 6. Comparing Dates
        let date1 = new Date("2023-10-05")
        let date2 = new Date("2023-10-10")
        let comparisonResult = ""
        if (date1 < date2) {
            comparisonResult = "date1 is before date2"
        }else if (date1 > date2) {
            comparisonResult = "date1 is after date2"
        }else{
            comparisonResult = "date1 is equal to date2"
        }
        // 7. Date Validation
        function isValidDate(dateStr) {
            return !isNaN(Date.parse(dateStr))
        }
        let userInput = "2023-10-15"
        let isValid = isValidDate(userInput)
        let validationMessage = isValid ? "Valid date format" : "Invalid date format"
        // Displaying Results
        console.log("Current Date and Time:", currentDate)
        console.log("Specific Date and Time:", specificDate)
        console.log("Date Components:", year, month, day, hours, minutes, seconds)
        console.log("Formatted Date:", formattedDate)
        console.log("Formatted Time:", formattedTime)
        console.log("Timestamp:", timestamp)
        console.log("Future Date:", futureDate)
        console.log("Date Comparison:", comparisonResult)
        console.log("Date Validation:", validationMessage)
    </script>
</body>
</html>
```

चित्र 16.7— तिथि गणित (Date Arithmetic) तिथि और समय का उदाहरण

आउटपुट



चित्र 16.8— तिथि गणित (Date Arithmetic) तिथि और समय उदाहरण का आउटपुट

16.3 प्रपत्र (Forms)

JavaScript में प्रपत्र (Forms) बनाने और उनके साथ कार्य करने में Document Object Model (DOM) का उपयोग करके प्रपत्र तत्व (form elements) बनाना, इवेंट श्रोता (event listeners) जोड़ना और प्रयोक्ता इनपुट को नियंत्रित करना शामिल होता है।

validateForm फ़ंक्शन

- यह फ़ंक्शन सामान्यतः फॉर्म के onsubmit हैंडलर के रूप में उपयोग होता है, जो फॉर्म सबमिट होने से पहले सभी फ़ील्ड्स का सत्यापन करता है।
- यदि कोई सत्यापन विधि false लौटाती है, तो यह फ़ॉर्म को सबमिट होने से रोकता है, जिससे संकेत मिलता है कि त्रुटियाँ हैं।
- इस फ़ंक्शन के अंदर आप विशिष्ट इनपुट फ़ील्ड्स के लिए अन्य सत्यापन फ़ंक्शन को कॉल कर सकते हैं और केवल तभी true लौटाते हैं जब सभी सत्यापन सफल हों।

checkEmail फ़ंक्शन

- यह फ़ंक्शन ईमेल इनपुट फ़ील्ड का सत्यापन करने के लिए उपयोग होता है, ताकि उसका स्वरूप वैध ईमेल प्रारूप से मेल खाए।
- यह ईमेल इनपुट फ़ील्ड को पैरामीटर के रूप में लेता है और इसके मान को एक रेगुलर एक्सप्रेशन पैटर्न से जाँचता है।
- यदि ईमेल मान वैध है, तो यह फ़ंक्शन true लौटाता है; अन्यथा false।

checkRadio फ़ंक्शन

- यह फ़ंक्शन रेडियो बटन के समूह का सत्यापन करता है ताकि कम से कम एक विकल्प चुना गया हो।
- यह रेडियो बटन समूह (आमतौर पर रेडियो बटन एलिमेंट्स की ऐरे) को पैरामीटर के रूप में लेता है।
- फ़ंक्शन रेडियो बटन पर लूप करता है और जाँचता है कि क्या कम से कम एक बटन चेक किया गया है। यदि हाँ, तो true लौटाता है; अन्यथा false।

checkDropdown फ़ंक्शन

- यह फ़ंक्शन ड्रॉपडाउन/सेलेक्ट एलिमेंट का सत्यापन करता है ताकि यह सुनिश्चित किया जा सके कि कोई वैध विकल्प चुना गया है।
- यह ड्रॉपडाउन एलिमेंट को पैरामीटर के रूप में लेता है और जाँचता है कि उसका selectedIndex शून्य से अधिक है या नहीं।
- यदि वैध विकल्प चुना गया हो, तो फ़ंक्शन true लौटाता है; अन्यथा false।

checkCheckbox फ़ंक्शन

- यह फ़ंक्शन चेकबॉक्स इनपुट का सत्यापन करता है कि क्या वह चेक (चयनित) है।

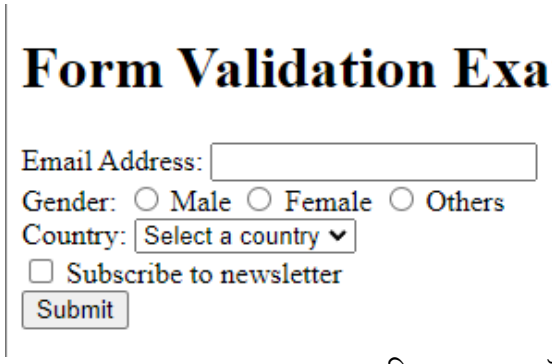
- यह चेकबॉक्स इनपुट एलिमेंट को पैरामीटर के रूप में लेता है और जाँचता है कि वह चेक है या नहीं।
- यदि चेकबॉक्स चेक है, तो यह true लौटाता है; अन्यथा false।
- इन सभी फ़ंक्शनों का उपयोग अकेले या validateForm फ़ंक्शन के अंदर सामूहिक रूप से किया जा सकता है ताकि विभिन्न प्रकार के फ़ॉर्म सत्यापन को निष्पादित किया जा सके। इन्हें संयोजित करके आप सुनिश्चित कर सकते हैं कि प्रयोक्ता द्वारा दर्ज किया गया इनपुट इच्छित स्वरूप और आवश्यकताओं के अनुरूप है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation Example</title>
</head>
<body>
    <h1>Form Validation Example</h1>
    <form onsubmit="return validateForm()">
        <label for="email">Email Address:</label>
        <input type="email" id="email" name="email" required>
        <span id="emailError" style="color: red;"></span><br>
        <label>Gender:</label>
        <input type="radio" id="male" name="gender" value="male">
        <label for="male">Male</label>
        <input type="radio" id="female" name="gender" value="female">
        <label for="female">Female</label>
        <input type="radio" id="others" name="gender" value="others">
        <label for="others">Others</label>
        <span id="genderError" style="color: red;"></span><br>
        <label for="country">Country:</label>
        <select name="country" id="country" required>
            <option value="">Select a country</option>
            <option value="india">India</option>
            <option value="usa">USA</option>
        </select>
        <span id="countryError"></span><br>
        <input type="checkbox" id="subscribe" name="subscribe" required>
        <label for="subscribe">Subscribe to newsletter</label>
        <span id="subscribeError"></span><br>
        <button type="submit">Submit</button>
    </form>
    <script>
```

```
        function validateForm() {
            // Reset Error Messages
            document.getElementById("emailError").textContent = ""
            document.getElementById("genderError").textContent = ""
            document.getElementById("countryError").textContent = ""
            document.getElementById("subscribeError").textContent = ""
            let isValid = true
            // Email Address Validation
            const emailInput = document.getElementById("email")
            const emailError = document.getElementById("emailError")
            const emailPattern = "/^[^\s@]+@[^\s@]+\.[^s@]+$/"
            if (!emailPattern.test(emailInput.value)) {
                emailError.textContent = "Invalid email format"
                isValid = false
            }
            // Radio Button (Gender) validation
            const genderInputs = document.querySelectorAll("input[name='gender']")
            const genderError = document.getElementById("genderError")
            let isGenderSelected = false
            genderInputs.forEach(input => {
                if (input.checked) {
                    isGenderSelected = true
                }
            })
            if (!isGenderSelected) {
                genderError.textContent = "Please select a gander"
                isValid = false
            }
            // Dropdown (Country) Validation
            const countrySelect = document.getElementById("country")
            const countryError = document.getElementById("countryError")
            if (countrySelect.selectedIndex === 0) {
                countryError.textContent = "Please select a country"
                isValid = false
            }
            // Checkbox (Subscribe) Validation
            const subscribeCheckbox = document.getElementById("subscribe")
            const subscribeError = document.getElementById("subscribeError")
            if (!subscribeCheckbox.checked) {
                subscribeError.textContent = "You must subscribe to continue"
                isValid = false
            }
            return isValid
        }
    </script>
</body>
</html>
```

चित्र 16.9— फॉर्म सत्यापन का उदाहरण

आउटपुट



चित्र 16.10— फॉर्म सत्यापन उदाहरण का आउटपुट

इस उदाहरण में—

- हमारे पास एक फ़ॉर्म है जिसमें ईमेल इनपुट, जेंडर के लिए रेडियो बटन, देश चयन के लिए ड्रॉपडाउन और सदस्यता के लिए चेकबॉक्स है।
- प्रत्येक इनपुट तत्व के लिए एक संबंधित त्रुटि संदेश span (जैसे— emailError, genderError) है जो सत्यापन त्रुटियाँ दिखाने के लिए उपयोग होता है।
- validateForm फ़ंक्शन फॉर्म सबमिट करने पर कॉल किया जाता है।
- यह ईमेल स्वरूप का सत्यापन करता है, जाँचता है कि जेंडर विकल्प चुना गया है, सुनिश्चित करता है कि देश चुना गया है, और यह भी कि चेकबॉक्स चयनित है।
- यदि कोई सत्यापन विफल होता है, तो संबंधित त्रुटि संदेश लाल रंग में प्रदर्शित होता है।

यदि कोई भी सत्यापन असफल होता है तो फॉर्म सबमिट होने से रोक दिया जाता है (validateForm में return false)।
यह उदाहरण यह दर्शाता है कि JavaScript का उपयोग करके विभिन्न प्रकार के फॉर्म सत्यापन कैसे किए जा सकते हैं और प्रयोक्ताओं को अमान्य डेटा दर्ज करने पर कैसे प्रतिक्रिया दी जाती है।

सारांश

- JavaScript स्ट्रिंग जोड़ना, उपस्ट्रिंग निकालना और केस बदलना जैसे कई स्ट्रिंग हेरफेर तकनीकों की सुविधा प्रदान करता है।
- JavaScript का उपयोग HTML फॉर्म को बनाने, सत्यापित करने और नियंत्रित करने के लिए किया जाता है।
- validateForm(), checkEmail(), checkRadio(), checkDropdown(), और checkCheckbox जैसे फ़ंक्शनों का उपयोग फॉर्म सत्यापन के लिए किया जाता है।
- फॉर्म सत्यापन यह सुनिश्चित करता है कि प्रयोक्ता इनपुट सबमिट करने से पहले वांछित प्रारूप और आवश्यकताओं के अनुरूप हो।
- Date ऑब्जेक्ट के माध्यम से तिथि और समय को नियंत्रित करना सरल बन जाता है, जिसमें फॉर्मेटिंग, तुलना, और समय क्षेत्र समायोजन जैसी विशेषताएँ होती हैं।
- JavaScript में तिथि और समय का प्रबंधन Date ऑब्जेक्ट के माध्यम से किया जाता है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. जावास्क्रिप्ट में स्ट्रिंग मैनिपुलेशन (string manipulation) क्या है?
   (क) स्ट्रिंग का रंग बदलना

(ख) स्ट्रिंग पर क्रियाएँ करना
(ग) नई स्ट्रिंग बनाना
(घ) दस्तावेज़ से स्ट्रिंग हटाना

2. जावास्क्रिप्ट में स्ट्रिंग को जोड़ने के लिए कौन-सा ऑपरेटर उपयोग किया जाता है?
(क) +
(ख) *
(ग) /
(घ) –

3. जावास्क्रिप्ट में किसी स्ट्रिंग की लंबाई कैसे ज्ञात की जाती है?
(क) str.len()
(ख) str.size()
(ग) str.length
(घ) str.count()

4. जावास्क्रिप्ट में substring विधि क्या करती है?
(क) स्ट्रिंग को ऐरे में विभाजित करती है
(ख) स्ट्रिंग के एक भाग को निकालती है
(ग) स्ट्रिंग को लोअरकेस में बदलती है
(घ) स्ट्रिंग में उप-स्ट्रिंग को प्रतिस्थापित करती है

5. जावास्क्रिप्ट में किसी स्ट्रिंग को अपरकेस में कैसे बदला जाता है?
(क) str.toUpperCaseCase()
(ख) str.toUppercase()
(ग) str.upper()
(घ) str.toUpperCase()

6. किस विधि का उपयोग किसी स्ट्रिंग के अंदर उप-स्ट्रिंग की स्थिति जानने के लिए किया जाता है?
(क) str.find()
(ख) str.index()
(ग) str.indexOf()
(घ) str.position()
7. जावास्क्रिप्ट में replace विधि क्या करती है?
(क) स्ट्रिंग से सभी अक्षरों को हटाती है
(ख) किसी उप-स्ट्रिंग को किसी अन्य स्ट्रिंग से प्रतिस्थापित करती है
(ग) स्ट्रिंग को ऐरे में विभाजित करती है
(घ) स्ट्रिंग के केस को बदलती है

8. जावास्क्रिप्ट में किसी स्ट्रिंग को डिलिमीटर (delimiter) का उपयोग करके ऐरे में विभाजित कैसे किया जाता है?
   (क) str.break(delimiter)
   (ख) str.split(delimiter)
   (ग) str.divide(delimiter)
   (घ) str.separate(delimiter)

9. जावास्क्रिप्ट में charAt विधि का उद्देश्य क्या है?
   (क) दो स्ट्रिंग को जोड़ना
   (ख) किसी विशेष इंडेक्स पर अक्षर प्राप्त करना
   (ग) स्ट्रिंग को लोअरकेस में बदलना
   (घ) स्ट्रिंग की लंबाई जानना

10. जावास्क्रिप्ट विधि split() उप-स्ट्रिंग के ऐरे को किस आधार पर लौटाती है?
   (क) स्ट्रिंग के पहले अक्षर के आधार पर
   (ख) डिलिमीटर की घटनाओं की संख्या पर
   (ग) उप-स्ट्रिंग की स्थिति पर
   (घ) निर्दिष्ट डिलिमीटर पर

ख. रिक्त स्थान भरिए

1. Concatenation दो या अधिक स्ट्रिंग को _____ करने की प्रक्रिया है।
2. किसी स्ट्रिंग की लंबाई ज्ञात करने के लिए प्रयुक्त विधि _____ है।
3. किसी स्ट्रिंग को लोअरकेस में बदलने के लिए आप _____ विधि का उपयोग कर सकते हैं।
4. किसी स्ट्रिंग के अंदर उप-स्ट्रिंग की स्थिति जानने के लिए जावास्क्रिप्ट विधि _____ है।
5. किसी स्ट्रिंग के एक भाग को निकालने के लिए प्रयुक्त जावास्क्रिप्ट विधि _____ है।
6. validateForm फ़ंक्शन को सामान्यतः एक फॉर्म के onsubmit हैंडलर के रूप में _____ के लिए प्रयोग किया जाता है।
7. किसी स्ट्रिंग को उप-स्ट्रिंग के ऐरे में विभाजित करने के लिए प्रयुक्त जावास्क्रिप्ट विधि _____ है।
8. आप किसी विशिष्ट तिथि और समय का प्रतिनिधित्व करने के लिए new _____ का उपयोग करके एक Date ऑब्जेक्ट बना सकते हैं।
9. किसी तिथि स्ट्रिंग के प्रारूप को सत्यापित करने के लिए प्रयुक्त जावास्क्रिप्ट फ़ंक्शन _____ है।
10. किसी स्ट्रिंग के अंदर किसी विशेष इंडेक्स पर अक्षर प्राप्त करने के लिए प्रयुक्त जावास्क्रिप्ट विधि _____ है।

ग. सही या गलत बताइए

1. स्ट्रिंग मैनिपुलेशन में किसी स्ट्रिंग का लिंग बदलना शामिल होता है।
2. replace विधि का उपयोग स्ट्रिंग में उप-स्ट्रिंग की स्थिति ज्ञात करने के लिए किया जाता है।
3. जावास्क्रिप्ट में स्ट्रिंग के केस को बदलने की कोई विधियाँ नहीं होतीं।
4. जावास्क्रिप्ट में concatenation विधि का उपयोग किसी स्ट्रिंग को ऐरे में विभाजित करने के लिए किया जाता है।
5. join विधि ऐरे के सभी तत्वों को जोड़कर एक स्ट्रिंग में बदल देती है।
6. जावास्क्रिप्ट में Date ऑब्जेक्ट का उपयोग किसी विशिष्ट तिथि और समय को प्रदर्शित करने के लिए किया जा सकता है।
7. टाइमस्टैम्प (Timestamps) 1 जनवरी, 1970 (UTC) से लेकर अब तक के मिलीसेकंड को दर्शाते हैं।
8. जावास्क्रिप्ट में .join() विधि एक डिलिमीटर को अपने पैरामीटर के रूप में लेती है।
9. .substring() विधि मूल स्ट्रिंग को बदल देती है।

10. validateForm फ़ंक्शन का उपयोग जावास्क्रिप्ट में फॉर्म तत्वों को बनाने के लिए किया जाता है।

घ. लघु उत्तर प्रश्न

1. स्ट्रिंग कंकैटेनेशन (concatenation) क्या है, और इसे जावास्क्रिप्ट में कैसे किया जाता है?
2. आप जावास्क्रिप्ट में किसी स्ट्रिंग की लंबाई कैसे जान सकते हैं?
3. indexOf() विधि का जावास्क्रिप्ट में क्या उद्देश्य है?
4. जावास्क्रिप्ट में किसी स्ट्रिंग को अपरकेस और लोअरकेस में कैसे बदला जाता है?
5. .split() विधि क्या करती है, और इसे स्ट्रिंग के साथ कैसे उपयोग किया जाता है?
6. जावास्क्रिप्ट में Date ऑब्जेक्ट क्या होता है, और आप इसे कैसे बना सकते हैं?
7. टाइमस्टैम्प क्या होते हैं, और जावास्क्रिप्ट में इन्हें कैसे दर्शाया जाता है?
8. आप जावास्क्रिप्ट में किसी मान के NaN होने की जाँच कैसे करते हैं?
9. जावास्क्रिप्ट में .join() विधि का उद्देश्य क्या है, और इसे कैसे उपयोग किया जाता है?
10. आप प्रयोक्ता द्वारा प्रविष्ट तिथि को सत्यापित कैसे करते हैं कि वह मान्य प्रारूप में है या नहीं?

सत्र 17
जावास्क्रिप्ट का उपयोग करके छवियों में हेरफेर करना

स्वभाव से एक अन्वेषक सुरेश ने "जावास्क्रिप्ट में इमेज और जिओलोकेशन में हेरफेर" के बारे में सीखा, और यह मानो रोमांचों का एक खजाना खोलने जैसा था। इमेज हेरफेर के साथ, वह चित्रों का आकार बदल सकता था और उन पर आकर्षक फिल्टर जोड़ सकता था, जिससे वे जादुई कलाकृतियों जैसे बन जाते थे। जिओलोकेशन का उपयोग करके, यह किसी खजाने के नक्शे जैसा था, जो उसे स्थान खोजने, रोमांच की योजना बनाने और रोचक स्थानों की खोज करने में मदद करता था। इन उपकरणों के साथ, उसने अद्भुत चित्र गैलरी बनाई और मित्रों के साथ शानदार यात्राओं की योजना बनाई, एक डिजिटल अन्वेषक और कलाकार बन गया, जैसे छिपे खजाने को खोजकर सुंदर फ़ोटो लेना। इससे इंटरनेट एक और अधिक रंगीन और रोमांचक स्थान बन गया जिसे खोजा जा सके! जैसा कि चित्र 17.1 में दर्शाया गया है।

चित्र 17.1— जावास्क्रिप्ट में इमेज और जिओलोकेशन में हेरफेर करता हुआ सुरेश

इस अध्याय में, आप इमेज में हेरफेर, जावास्क्रिप्ट से नई छवियाँ बनाना, इमेज रोलओवर प्रभाव, और टाइमर एवं छवियों के बारे में जानेंगे।

17.1 इमेज में हेरफेर करना

जावास्क्रिप्ट में इमेज में हेरफेर HTML5 के <canvas> एलिमेंट और इसके संबंधित 2D ड्रॉइंग कॉन्टेक्स्ट का उपयोग करके प्राप्त किया जा सकता है। इस पद्धति से, आप किसी वेब पेज पर छवियों को लोड, ड्रॉ, संशोधित और सेव कर सकते हैं। नीचे कुछ सामान्य इमेज हेरफेर कार्य दिए गए हैं और उन्हें जावास्क्रिप्ट और <canvas> एलिमेंट का उपयोग करके कैसे किया जाता है—

चित्र लोड करना — किसी छवि में हेरफेर करने के लिए, पहले उसे लोड करना आवश्यक है। आप एक Image ऑब्जेक्ट बना सकते हैं और उसकी src विशेषता को इमेज फ़ाइल के URL पर सेट कर सकते हैं। जब छवि लोड हो जाती है, तो आप उस पर विभिन्न क्रियाएँ कर सकते हैं।

```
javascript
const img = new Image();
img.src = 'image.jpg';
img.onload = function () {
  // छवि लोड हो गई है और यहाँ हेरफेर किया जा सकता है
};
```

कैनवास पर छवि ड्रॉ करना — लोड की गई छवि को कैनवास पर प्रदर्शित करने के लिए, आप 2D ड्रॉइंग कॉन्टेक्स्ट की drawImage विधि का उपयोग कर सकते हैं। यह आपको छवि, स्थिति, और आकार को निर्दिष्ट करने की अनुमति देता है।

javascript
```
const canvas = document.getElementById('myCanvas');
const ctx = canvas.getContext('2d');
ctx.drawImage(img, 0, 0, canvas.width, canvas.height);
```

छवि पिक्सेल को संशोधित करना — आप getImageData और putImageData विधियों का उपयोग करके किसी छवि के व्यक्तिगत पिक्सेल को एक्सेस और संशोधित कर सकते हैं। यह आपको छवि पर विभिन्न फिल्टर और प्रभाव लागू करने की अनुमति देता है।

javascript
```
const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
const data = imageDatक.data; // पिक्सेल डेटा (RGBA मान) युक्त ऐरे
for (let i = 0; i < datक.length; i += 4) {
  // यहाँ पिक्सेल डेटा में हेरफेर करें (जैसे— रंग बदलना)
  data[i] = 255 - data[i]; // रेड चैनल को उल्टा करें
}
ctx.putImageData(imageData, 0, 0);
```

छवि का आकार बदलना — आप छवि को एक भिन्न आकार के कैनवास पर ड्रॉ करके उसका आकार बदल सकते हैं। इसके लिए आपको वांछित आयामों वाला एक नया कैनवास बनाना होगा और उस पर छवि को ड्रॉ करना होगा।

javascript
```
const newCanvas = document.createElement('canvas');
const newCtx = newCanvas.getContext('2d');
newCanvas.width = newWidth;
newCanvas.height = newHeight;
newCtx.drawImage(img, 0, 0, newWidth, newHeight);
```

संशोधित छवि को सहेजना — संशोधित छवि को सहेजने के लिए, आप कैनवास की सामग्री को toDataURL विधि का उपयोग करके डेटा URL में बदल सकते हैं। फिर आप एक लिंक बना सकते हैं जो प्रयोक्ताओं को छवि डाउनलोड करने की अनुमति देता है।

javascript
```
const modifiedImageDataUrl = canvas.toDataURL('image/jpeg'); // या 'image/png'
const downloadLink = document.createElement('a');
downloadLink.href = modifiedImageDataUrl;
downloadLink.download = 'modified_image.jpg'; // वांछित फ़ाइल नाम निर्दिष्ट करें
downloadLink.click();
```

फिल्टर और प्रभाव लागू करना — आप पिक्सेल डेटा में हेरफेर करके कस्टम फिल्टर और प्रभाव बना सकते हैं। उदाहरण के लिए, आप ग्रेस्केल, ब्लर, सेपिया या कोई अन्य छवि फिल्टर लागू कर सकते हैं।

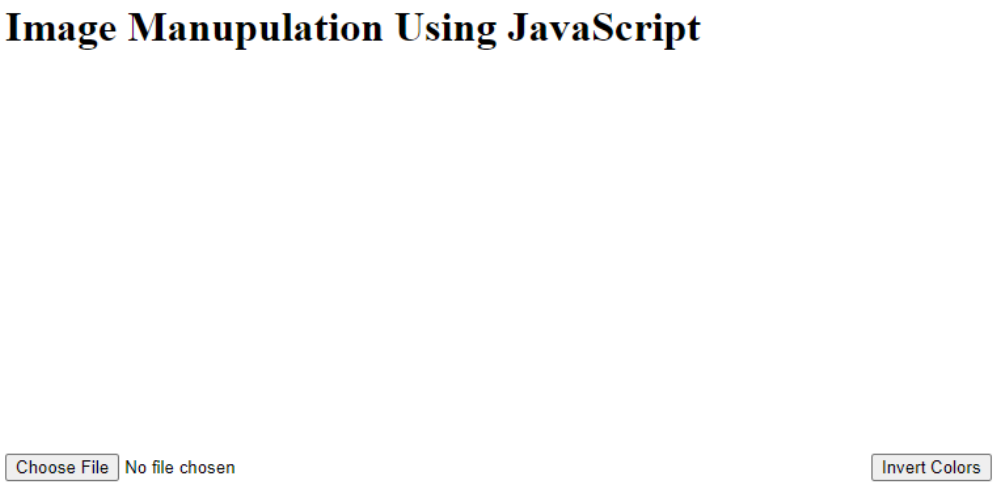
जावास्क्रिप्ट में इमेज हेरफेर अत्यंत शक्तिशाली हो सकता है, जो आपको इंटरैक्टिव अनुप्रयोग बनाने और वेब पेज पर प्रदर्शित छवियों पर वास्तविक समय में संशोधन करने की अनुमति देता है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Manupulation Example</title>
</head>
<body>
    <h1>Image Manupulation Using JavaScript</h1>

    <input type="file" id="imageInput" accept="image*">
    <canvas id="canvas" width="400" height="300"></canvas>
    <button id="invertButton">Invert Colors</button>
    <a id="downloadLink" style="display: none;">Download Modified Image</a>

    <script>
        const canvas = document.getElementById("canvas")
        const ctx = canvas.getContext("2d")
        const imageInput = document.getElementById("imageInput")
        const invertButton = document.getElementById("invertButton")
        const downloadLink = document.getElementById("downloadLink")
        let img = new Image()
        // Event Listener for Image Input
        imageInput.addEventListener("change", function(event) {
            const file = event.target.files[0]
            if (file) {
                const reader = new FileReader()
                reader.onload = function(e) {
                    img.src = e.target.result
                }
                reader.readAsDataURL(file)
            }
        })
        // Event Listener for image load
        img.onload = function () {
            canvas.width = img.width
            canvas.height = img.height
            ctx.drawImage(img, 0, 0, canvas.width, canvas.height)
        }
        // Event Listener for Invert Button
        invertButton.addEventListener("click", function() {
            const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height)
            const data = imageData.data
            // Invert colors by subtracting each color channel from 255
            for (let i = 0; i < data.length; i+=4) {
                data[i] = 255 - data[i] // Red
                data[i+1] = 255 - data[i+1] // Red
                data[i+2] = 255 - data[i+2] // Red
                // Alpha (data[i+3]) remains unchanged
            }
            ctx.putImageData(imageData, 0, 0)
            // Set the download link attributes
            downloadLink.href = canvas.toDataURL("image/jpeg")
            downloadLink.download = "modified_image.jpg"
            downloadLink.style.display = "block"
        })
    </script>
</body>
</html>
```

चित्र 17.2— जावास्क्रिप्ट और <canvas> एलिमेंट का उपयोग करते हुए

आउटपुट

पृष्ठ लोड होने के बाद जैसा कि चित्र 17.3 में दिखाया गया है।

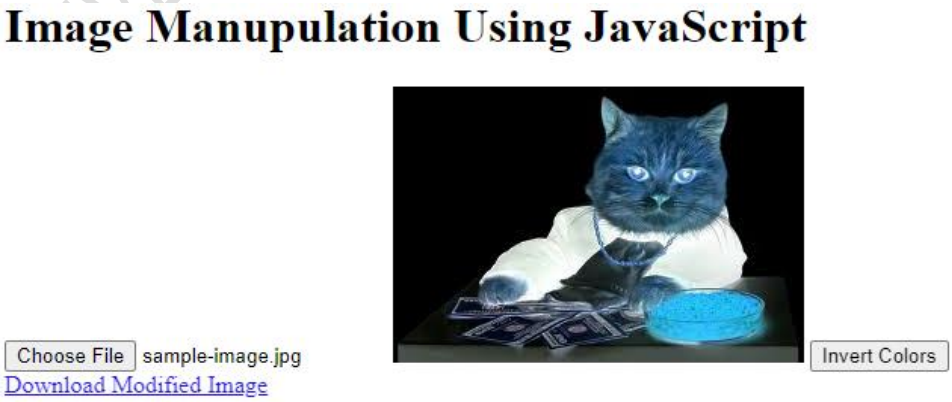


चित्र 17.3— जावास्क्रिप्ट का उपयोग करते हुए इमेज हेरफेर के बाद लोडिंग

छवि चयन के बाद जैसा कि चित्र 17.4 में दिखाया गया है।

चित्र 17.4— जावास्क्रिप्ट का उपयोग करते हुए इमेज हेरफेर के बाद

इन्वर्ट के बाद जैसा कि चित्र 17.5 में दिखाया गया है।



चित्र 17.5— जावास्क्रिप्ट का उपयोग करते हुए इन्वर्ट हेरफेर के बाद

अब हम डाउनलोड बटन पर क्लिक करके डाउनलोड कर सकते हैं।

कोड व्याख्या — प्रदान किए गए संशोधित कोड में, हमारे पास एक HTML वेबपेज है जो प्रयोक्ताओं को एक छवि अपलोड करने, उस पर रंग उलटने वाला फिल्टर लागू करने, और फिर संशोधित छवि को निर्दिष्ट नाम के साथ डाउनलोड करने की अनुमति देता है। आइए जानें कि कोड का प्रत्येक भाग कैसे कार्य करता है—

HTML संरचना

- हमारे पास एक <input> एलिमेंट है जिसका प्रकार "file" है (imageInput), जो प्रयोक्ताओं को उनके डिवाइस से छवि फ़ाइल का चयन करने की अनुमति देता है।
- एक <canvas> एलिमेंट (canvas) छवि को प्रदर्शित और हेरफेर करने के लिए उपयोग किया जाता है। "Invert Colors" नामक एक बटन (invertButton) रंग उलटने वाले फिल्टर को ट्रिगर करता है।
- एक <a> एलिमेंट (downloadLink) प्रारंभ में छिपा होता है और जब छवि संशोधित हो जाती है तब दृश्य होता है। इसका उपयोग संशोधित छवि को डाउनलोड करने के लिए किया जाता है।

जावास्क्रिप्ट की कार्यप्रणाली

- हम जावास्क्रिप्ट का उपयोग HTML एलिमेंट्स के साथ बातचीत करने और छवि में हेरफेर करने के लिए करते हैं।
- जब कोई प्रयोक्ता फ़ाइल इनपुट का उपयोग करके एक छवि चुनता है, तो एक इवेंट लिस्नर (imageInput.addEventListener) ट्रिगर होता है। यह चयनित छवि फ़ाइल को पढ़ता है, एक Image ऑब्जेक्ट (img) में लोड करता है, और जब यह लोड हो जाती है (img.onload), तब कैनवास पर प्रदर्शित करता है।
- जब "Invert Colors" बटन क्लिक किया जाता है (invertButton.addEventListener), तो यह एक इवेंट हैंडलर को ट्रिगर करता है जो निम्नलिखित चरणों को निष्पादित करता है—
- ctx.getImageData का उपयोग करके कैनवास का पिक्सेल डेटा प्राप्त करता है।
- पिक्सेल डेटा के माध्यम से इटरेट करता है और प्रत्येक रंग चैनल (रेड, ग्रीन, ब्लू) को 255 से घटाकर रंगों को उलटता है।
- ctx.putImageData का उपयोग करके संशोधित पिक्सेल डेटा को पुनः कैनवास पर रखता है।
- डाउनलोड लिंक (downloadLink) के href और download गुणों को सेट करता है ताकि डेटा URL और वांछित फ़ाइल नाम ("modified_image.jpg") निर्दिष्ट हो सके।
- डाउनलोड लिंक को दृश्य बनाता है (downloadLink.style.display = 'block')।

संशोधित छवि को डाउनलोड करना

- जब डाउनलोड लिंक दृश्य हो जाता है, तो प्रयोक्ता उस पर क्लिक करके संशोधित छवि को डाउनलोड कर सकते हैं। लिंक का download गुण फ़ाइल नाम निर्दिष्ट करता है, और href गुण संशोधित छवि के डेटा URL को रखता है।

रंग उलटने वाला फिल्टर

- रंग उलटने वाला फिल्टर छवि के पिक्सेल डेटा के माध्यम से लूप करके और प्रत्येक रंग चैनल के मान को 255 से घटाकर लागू किया जाता है। यह प्रक्रिया रंगों को उलटती है, प्रभावी रूप से एक नेगेटिव छवि बनाती है।

कुल मिलाकर, यह कोड एक सरल इमेज हेरफेर तकनीक को दर्शाता है जिसमें प्रयोक्ता एक छवि अपलोड कर सकते हैं, उस पर रंग उलटने वाला फिल्टर लागू कर सकते हैं, और फिर संशोधित छवि को निर्दिष्ट नाम के साथ डाउनलोड कर सकते हैं। <canvas> एलिमेंट का उपयोग छवि को ड्रॉ और हेरफेर करने के लिए किया गया है, और जावास्क्रिप्ट इंटरैक्शन और छवि प्रसंस्करण को संभालता है।

अभ्यास 17.1

- विभिन्न इमेज हेरफेर कार्यों को लिखिए और उन्हें जावास्क्रिप्ट तथा <canvas> एलिमेंट का उपयोग करते हुए संपादित कीजिए।
  - o चित्र लोड करना
  - o चित्र को कैनवास पर ड्रॉ करना
  - o चित्र के पिक्सेल को संशोधित करना

17.2 जावास्क्रिप्ट द्वारा नई छवियाँ बनाना

यह अवधारणा जावास्क्रिप्ट (JavaScript) का उपयोग करके किसी वेबपृष्ठ पर गतिशील रूप से छवियाँ (images) उत्पन्न करने और प्रदर्शित करने से संबंधित है। आप जावास्क्रिप्ट का प्रयोग करके स्रोत (URL) और अन्य विशेषताओं को निर्दिष्ट करके तुरंत (on-the-fly) छवियाँ बना सकते हैं। यह तकनीक तब विशेष रूप से उपयोगी होती है जब आपको प्रयोक्ता की क्रिया या सर्वर से प्राप्त डेटा के आधार पर छवियों को लोड करना होता है।

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Image Creation with JavaScript</title>
</head>
<body>
<h1>Image creation with JavaScript</h1>

<!-- 'box' ID के साथ एक प्लेसहोल्डर 'div' तत्व बनाएँ -->
<div id="box"></div>

<!-- JavaScript को <script> तत्व के अंदर एम्बेड करें -->
<script>
    // एक नई इमेज एलिमेंट बनाएँ
    const image = document.createElement('img');
    // छवि का स्रोत निर्धारित करें (अपनी छवि का URL बदलें)
    image.setAttribute('src', 'background.jpg');
    // छवि के लिए वैकल्पिक टेक्स्ट (alt text) निर्धारित करें
    image.setAttribute('alt', 'music');
    // छवि की ऊँचाई और चौड़ाई (पिक्सल में) सेट करें
    image.setAttribute('height', 350);
    image.setAttribute('width', 550);
    // छवि पर स्टाइल लागू करें (लाल बॉर्डर)
    image.style.border = '5px solid red';
    // यदि छवि लोड नहीं होती है तो त्रुटि को हैंडल करें
    image.onerror = function handleError() {
        console.log('Image could not be loaded');
```

```
        // वैकल्पिक रूप से बैकअप छवि सेट की जा सकती है या छवि को छिपाया जा सकता है
        // image.src = 'backup-image.png';
        // image.style.display = 'none';
    };
    // यदि छवि सफलतापूर्वक लोड हो जाती है तो उसे हैंडल करें
    image.onload = function handleImageLoaded() {
        console.log('Image loaded successfully');
    };
    // 'box' ID वाले तत्व को प्राप्त करें और उसमें छवि जोड़ें
    const box = document.getElementById('box');
    box.appendChild(image);
</script>
</body>
</html>
```

आउटपुट

चित्र 17.6 — जावास्क्रिप्ट द्वारा छवि निर्माण का आउटपुट

कोड व्याख्या

1. const image = document.createElement('img'); — यह पंक्ति एक नया <img> तत्व बनाती है और उसे image नामक स्थिरांक (constant) को सौंपती है।
2. image.setAttribute('src', 'background.jpg'); — यह <img> तत्व के src गुण को 'background.jpg' पर सेट करती है। सुनिश्चित करें कि यह छवि फ़ाइल सही पथ पर हो।
3. image.setAttribute('alt', 'music'); — छवि के लिए वैकल्पिक टेक्स्ट 'music' सेट किया जाता है।
4. image.setAttribute('height', 350); और image.setAttribute('width', 550); — ये पंक्तियाँ छवि की ऊँचाई और चौड़ाई को क्रमशः 350 और 550 पिक्सल पर सेट करती हैं।
5. image.style.border = '5px solid red'; — छवि पर लाल रंग की 5 पिक्सल मोटी बॉर्डर लागू करता है।
6. image.onerror = function handleError() { ... }; — यदि छवि लोड नहीं होती है तो यह हैंडलर एक त्रुटि संदेश लॉग करता है।

7. image.onload = function handleImageLoaded() { ... }; — जब छवि सफलतापूर्वक लोड हो जाती है, तब यह हैंडलर एक सफलता संदेश लॉग करता है।
8. const box = document.getElementById('box'); — HTML में ID box वाले तत्व को प्राप्त करता है।
9. box.appendChild(image); — अंततः image तत्व को box तत्व में जोड़ता है जिससे वह वेबपृष्ठ पर प्रदर्शित हो सके।

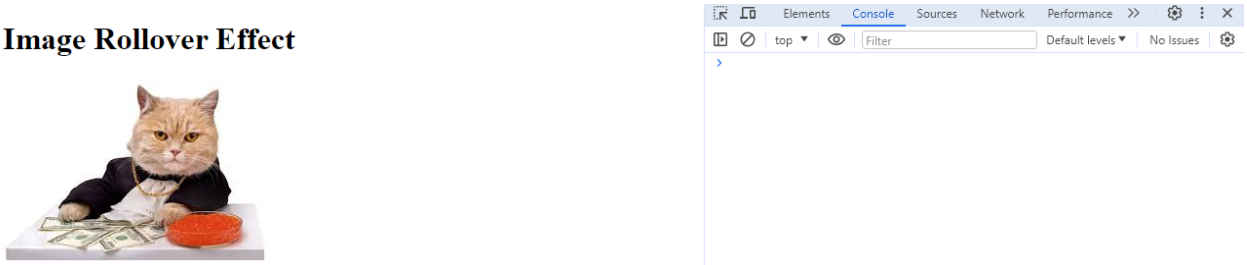
17.3 इमेज रोलओवर इफेक्ट

इमेज रोलओवर इफेक्ट (Image Rollover Effect) एक सामान्य वेबसाइट अंतःक्रिया है जिसमें जैसे ही प्रयोक्ता किसी छवि पर माउस कर्सर ले जाता है, छवि की उपस्थिति बदल जाती है। यह प्रभाव आम तौर पर जावास्क्रिप्ट और CSS के माध्यम से प्राप्त किया जाता है। जब माउस किसी छवि पर होता है, तब वह अपनी स्रोत छवि या दृश्य प्रभावों को बदल सकता है जैसे रंग बदलना या बॉर्डर जोड़ना।

उदाहरण

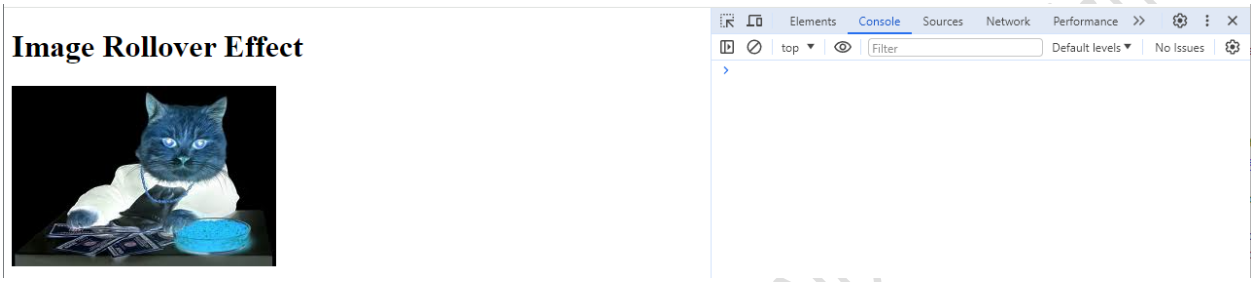
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Image Rollover Effect</title>
</head>
<body>
<h1>Image Rollover Effect</h1>

<!-- एक प्रारंभिक छवि के साथ <img> तत्व बनाएँ -->
<img id="rolloverImage" src="background.jpg" alt="Initial Image" onmouseover="changeImage();"
onmouseout="restoreImage();">

<script>
    // माउस होवर पर छवि स्रोत बदलने के लिए जावास्क्रिप्ट फ़ंक्शन
    function changeImage() {
      const imageElement = document.getElementById('rolloverImage');
      imageElement.src = 'modified_image.jpg'; // अपनी होवर इमेज का पथ यहाँ दें
    }
    // माउस हटाने पर मूल छवि पुनः लाने के लिए जावास्क्रिप्ट फ़ंक्शन
    function restoreImage() {
      const imageElement = document.getElementById('rolloverImage');
      imageElement.src = 'initial-image.jpg'; // अपनी प्रारंभिक छवि का पथ यहाँ दें
    }
</script>
</body>
</html>
```

आउटपुट



चित्र 17.7 — इमेज रोलओवर प्रभाव का आउटपुट

हवर करने के बाद जैसा कि चित्र 17.8 में दिखाया गया है।



चित्र 17.8 — हवर के बाद इमेज रोलओवर प्रभाव का आउटपुट

कोड व्याख्या

- <!DOCTYPE html> — यह पंक्ति HTML दस्तावेज़ के प्रकार और संस्करण की घोषणा करती है।
- <html lang="en"> — HTML दस्तावेज़ का आरंभिक टैग, जिसमें भाषा इंग्लिश निर्धारित की गई है।
- <head> — यह अनुभाग दस्तावेज़ से संबंधित मेटाडेटा (metadata) को समाहित करता है, जैसे कि वर्णमाला एन्कोडिंग और पृष्ठ शीर्षक।
- <meta charset="UTF-8"> — दस्तावेज़ की वर्णमाला एन्कोडिंग को UTF-8 के रूप में निर्दिष्ट करता है, जो विस्तृत वर्णों (कैरेक्टर) का समर्थन करता है।
- <title>Image Rollover Effect</title> — वेबपृष्ठ का शीर्षक सेट करता है, जो ब्राउज़र की टैब में दिखता है।
- <body> — वेबपृष्ठ की मुख्य सामग्री इस टैग के अंदर रहती है।
- <h1>Image Rollover Effect</h1> — यह हेडिंग "Image Rollover Effect" को मुख्य शीर्षक के रूप में प्रदर्शित करती है।
- <img id="rolloverImage" src="initial-image.jpg" alt="Initial Image" onmouseover="changeImage();" onmouseout="restoreImage();"> — यह पंक्ति एक <img> तत्व बनाती है, जिसमें निम्न विशेषताएँ होती हैं—
  - id="rolloverImage" — छवि को पहचानने के लिए ID सेट करता है।
  - src="initial-image.jpg" — छवि का प्रारंभिक स्रोत निर्दिष्ट करता है।
  - alt="Initial Image" — छवि लोड न होने की स्थिति में वैकल्पिक टेक्स्ट प्रदान करता है।
  - onmouseover="changeImage();" — माउस होवर करते समय changeImage फ़ंक्शन को कॉल करता है।
  - onmouseout="restoreImage();" — माउस हटने पर restoreImage फ़ंक्शन को कॉल करता है।
- <script> — यह HTML दस्तावेज़ में एक स्क्रिप्ट अनुभाग आरंभ करता है।

- function changeImage() { ... } — यह एक जावास्क्रिप्ट फ़ंक्शन परिभाषित करता है जो छवि स्रोत को बदलता है जब माउस उस पर होवर करता है।
- const imageElement = document.getElementById('rolloverImage'); — ID द्वारा छवि तत्व को प्राप्त करता है और उसे imageElement नामक स्थिरांक में स्टोर करता है।
- imageElement.src = 'hover-image.jpg'; — src को 'hover-image.jpg' में बदलता है ताकि होवर स्थिति में नई छवि दिखे।
- function restoreImage() { ... } — माउस हटने पर छवि को पुनः स्थापित करने हेतु यह फ़ंक्शन परिभाषित किया गया है।
- imageElement.src = 'initial-image.jpg'; — मूल छवि को पुनः src में सेट करता है।
- </script> — स्क्रिप्ट अनुभाग का समापन करता है।

यह कोड एक बुनियादी इमेज रोलओवर प्रभाव उत्पन्न करता है, जिससे जब प्रयोक्ता माउस छवि पर ले जाता है तो वह बदलती है और माउस हटाने पर अपनी प्रारंभिक स्थिति में लौट आती है।

17.4 टाइमर और छवियाँ

जावास्क्रिप्ट (JavaScript) में टाइमर (Timers) आपको कोड को निर्दिष्ट अंतरालों पर निष्पादित करने की अनुमति देते हैं। जब बात छवियों (images) की होती है, तो टाइमर का उपयोग विभिन्न प्रभाव (effects) उत्पन्न करने के लिए किया जा सकता है। उदाहरणस्वरूप, आप एक ऐसा स्लाइड शो बना सकते हैं जो निश्चित समयांतराल के बाद स्वतः छवियाँ बदलता है, या किसी छवि की स्थिति अथवा गुणों को समय के साथ अद्यतन करके एनिमेशन बना सकते हैं, जैसा कि चित्र 17.9 में दर्शाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Timers and Image Example</title>
</head>
<body>
    <h1>Timers and Image Example</h1>
    <!-- Create an <img> element with an initial image -->
    <img id="timerImage" src="background.jpg" alt="Initial Image">
    <script>
        // Get the image element by ID
        const imageElement = document.getElementById('timerImage');
        // Array of image URLs to cycle through
        const imageUrls = ['modified-image.jpg', 'sample-image.jpg', 'pause.jpg','play.jpg'];
        let currentIndex = 0; // Index of the currently displayed image
        // Function to change the image
        function changeImage() {
            // Change the image source to the next URL in the array
            imageElement.src = imageUrls[currentIndex];
            // Increment the index and wrap around if needed
            currentIndex = (currentIndex + 1) % imageUrls.length;
            // Set a timer to call the function again after 3 seconds (3000 milliseconds)
            setTimeout(changeImage, 3000);
        }
        // Start the timer to change the image initially
        setTimeout(changeImage, 3000);
    </script>
</body>
</html>
```

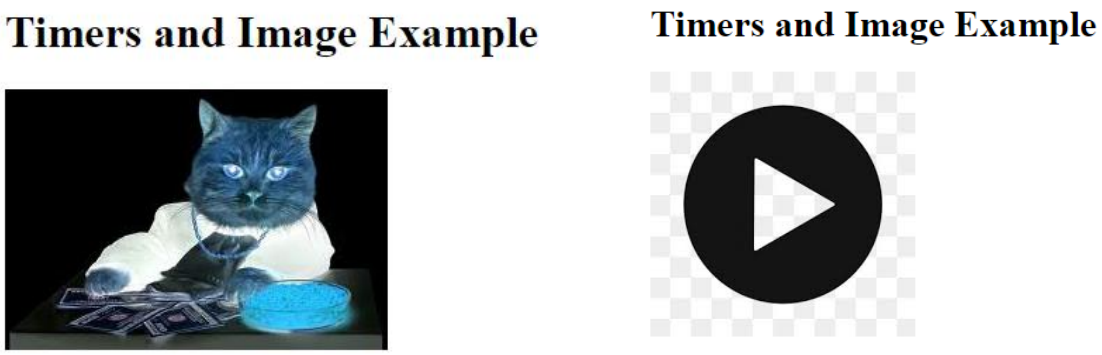
चित्र 17.9— जावास्क्रिप्ट में टाइमर

आउटपुट

Timers and Image Example

चित्र 17.10— जावास्क्रिप्ट में टाइमर का आउटपुट

3 सेकंड के बाद जैसा कि चित्र 17.11 में दिखाया गया है।



चित्र 17.11— जावास्क्रिप्ट में 3 सेकंड बाद का आउटपुट

यह प्रक्रिया निरंतर चलती रहती है।

कोड स्पष्टीकरण —

HTML संरचना — हम सामान्य HTML5 दस्तावेज़ संरचना से प्रारंभ करते हैं, जिसमें <!DOCTYPE> घोषणा, भाषा विशेषता (lang) सहित <html> टैग, और <head> तथा <body> अनुभाग शामिल होते हैं।

पृष्ठ शीर्षक और शीर्षक — <head> अनुभाग के अंदर, हम दस्तावेज़ का कैरेक्टर एनकोडिंग सेट करते हैं और पृष्ठ के लिए शीर्षक प्रदान करते हैं।
<body> अनुभाग में <h1> तत्व होता है जिसमें "Timers and Image Example" टेक्स्ट होता है, जो पृष्ठ शीर्षक के रूप में कार्य करता है।

छवि तत्व — हम एक <img> तत्व बनाते हैं जिसका आईडी "timerImage" होता है।
src विशेषता प्रारंभ में "background.jpg" पर सेट होती है, जो पृष्ठ लोड होने पर प्रदर्शित होने वाली छवि होती है। alt विशेषता वैकल्पिक टेक्स्ट प्रदान करती है, जो एक्सेसिबिलिटी के लिए उपयोगी होती है।
जावास्क्रिप्ट कोड — जावास्क्रिप्ट कोड <script> तत्व के अंदर एम्बेड किया जाता है, और यह document.getElementById('timerImage') का उपयोग करके "timerImage" आईडी वाले छवि तत्व को संदर्भित करने से प्रारंभ होता है।
imageUrls नामक एक ऐरे परिभाषित किया जाता है। इसमें उन छवियों के यूआरएल होते हैं जिन्हें स्लाइड शो में दिखाया जाना है। इस उदाहरण में चार छवियाँ हैं— "modified_image.jpg", "datascience.jpg", "pause.jpg", और "play.jpg"।
currentIndex एक चर (variable) है जो वर्तमान में प्रदर्शित छवि का अनुक्रमांक imageUrls ऐरे में ट्रैक करता है। इसे 0 से प्रारंभ किया जाता है, जो ऐरे की पहली छवि को दर्शाता है।
changeImage नामक फ़ंक्शन परिभाषित किया गया है। यह फ़ंक्शन प्रदर्शित छवि को बदलने और अगली छवि के लिए टाइमर सेट करने के लिए उत्तरदायी है।

changeImage के अंदर — छवि तत्व (imageElement) की src विशेषता imageUrls ऐरे में अगली छवि के यूआरएल पर सेट कर दी जाती है। इससे प्रदर्शित छवि प्रभावी रूप से बदल जाती है।
currentIndex को बढ़ाया जाता है, और % (मॉड्युलो ऑपरेटर) का उपयोग यह सुनिश्चित करने के लिए किया जाता है कि जब यह ऐरे के अंत तक पहुँच जाए तो फिर से 0 पर लौट आए— जिससे स्लाइड शो सभी छवियों के बीच चक्रीय रूप से चलता रहता है।
setTimeout का उपयोग changeImage को 3 सेकंड (3000 मिलीसेकंड) के विलंब के बाद पुनः कॉल करने के लिए किया जाता है। इससे प्रत्येक छवि के बीच 3 सेकंड के अंतराल पर एक स्वतः छवि स्लाइड शो प्रभाव उत्पन्न होता है।

अंततः, changeImage फ़ंक्शन के बाहर setTimeout(changeImage, 3000) को कॉल करके टाइमर आरंभ किया जाता है, जिससे पृष्ठ लोड होते ही स्लाइड शो प्रारंभ हो जाता है।
परिणामस्वरूप, एक स्वचालित छवि स्लाइड शो बनता है जो निर्दिष्ट छवियों को प्रत्येक 3 सेकंड में बदलता रहता है। यह उदाहरण किसी वेबपृष्ठ पर गतिशील (dynamic) सामग्री बनाने के लिए टाइमर के उपयोग को दर्शाता है।

17.5 जावास्क्रिप्ट द्वारा HTML तत्वों में हेरफेर

जावास्क्रिप्ट का उपयोग करके HTML तत्वों में हेरफेर का अर्थ है किसी वेब पेज पर HTML तत्वों की सामग्री, शैली (style) और संरचना को गतिशील रूप से बदलना। यह सामान्यतः इंटरैक्टिव और उत्तरदायी वेब अनुप्रयोग बनाने के लिए किया जाता है। नीचे HTML तत्वों में हेरफेर के लिए कुछ सामान्य कार्य और तकनीकें दी गई हैं—

तत्वों तक पहुँच प्राप्त करना (Accessing Elements)—

जावास्क्रिप्ट में विभिन्न विधियों जैसे getElementById, getElementsByClassName, getElementsByTagName, और querySelector का उपयोग करके HTML तत्वों तक पहुँचा जा सकता है। ये विधियाँ DOM (Document Object Model) तत्वों के संदर्भ लौटाती हैं।

```
const elementById = document.getElementById('myId');
const elementsByClass = document.getElementsByClassName('myClass');
const elementsByTag = document.getElementsByTagName('div');
const elementByQuery = document.querySelector('.mySelector');
```

सामग्री में संशोधन (Modifying Content)—

innerHTML या textContent गुणों का उपयोग करके तत्वों की सामग्री को बदला जा सकता है। innerHTML HTML सामग्री सेट करता है, जबकि textContent केवल सामान्य टेक्स्ट सेट करता है।

```
const element = document.getElementById('myElement');
element.innerHTML = '<strong>New Content</strong>';
element.textContent = 'New Text Content';
```

शैली में परिवर्तन (Changing Styles)—

style गुण का उपयोग करके तत्वों की शैली को बदला जा सकता है— जैसे backgroundColor, fontSize, color आदि।

```
const element = document.getElementById('myElement');
element.style.backgroundColor = 'blue';
element.style.fontSize = '16px';
```

क्लास जोड़ना/हटाना (Adding/Removing Classes)—

classList गुण का उपयोग करके किसी तत्व में CSS क्लास जोड़ी या हटाई जा सकती है। यह पूर्वनिर्धारित शैली लागू करने या एनिमेशन के लिए उपयोगी होता है।

```
const element = document.getElementById('myElement');
element.classList.add('newClass');
element.classList.remove('oldClass');
```

तत्व बनाना और जोड़ना (Creating and Appending Elements)—

createElement और appendChild विधियों का उपयोग करके नए HTML तत्व बनाए और DOM में जोड़े जा सकते हैं।

```
const newElement = document.createElement('div');
newElement.textContent = 'New Element';
```

```
document.body.appendChild(newElement);
```

घटनाओं का प्रबंधन (Event Handling)—

किसी तत्व पर ईवेंट लिस्नर (addEventListener) जोड़कर आप क्लिक, होवर जैसी प्रयोक्ता क्रियाओं के लिए प्रतिक्रिया प्रदान कर सकते हैं।

```
const button = document.getElementById('myButton');
button.addEventListener('click', function () {
  alert('Button Clicked!');
});
```

तत्व हटाना (Removing Elements)—

removeChild विधि का उपयोग करके किसी DOM तत्व को हटाया जा सकता है।

```
const elementToRemove = document.getElementById('toBeRemoved');
elementToRemove.parentNode.removeChild(elementToRemove);
```

डेटा विशेषताएँ (Data Attributes)—

data-* विशेषताओं का उपयोग करके HTML तत्वों में कस्टम डेटा स्टोर किया जा सकता है, जिसे जावास्क्रिप्ट से एक्सेस और संशोधित किया जा सकता है।

```
html
<div id="myElement" data-info="some data"></div>
javascript
const element = document.getElementById('myElement');
const data = element.getAttribute('data-info');
```

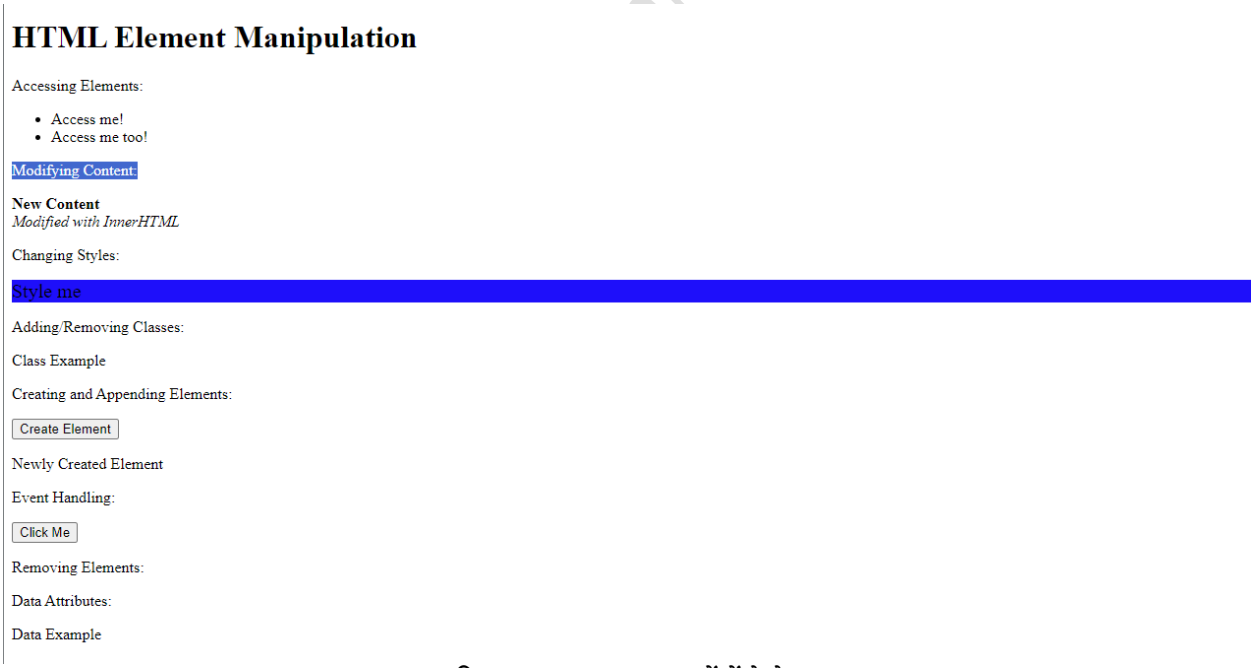
उदाहरण

```
html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML Element Manipulation</title>
  <style>
    .highlighted {
      background-color: yellow;
    }
  </style>
</head>
<body>
  <h1 id="myHeading">HTML Element Manipulation</h1>
  <p>Accessing Elements:</p>
  <ul>
    <li id="accessExample">Access me!</li>
```

```
  <li class="accessExample">Access me too!</li>
</ul>
<p>Modifying Content:</p>
<div id="contentExample">Initial Content</div>
<div id="innerHtmlExample">InnerHTML Example</div>
<p>Changing Styles:</p>
<div id="styleExample">Style me</div>
<p>Adding/Removing Classes:</p>
<div id="classExample">Class Example</div>
<p>Creating and Appending Elements:</p>
<button id="createElementBtn">Create Element</button>
<div id="container"></div>
<p>Event Handling:</p>
<button id="clickMeBtn">Click Me</button>
<p>Removing Elements:</p>
<div id="removeExample">Remove me</div>
<p>Data Attributes:</p>
<div id="dataExample" data-info="Custom Data Attribute">Data Example</div>
<script>
  // Accessing Elements
  const elementById = document.getElementById('accessExample');
  const elementsByClass = document.getElementsByClassName('accessExample');
  // Modifying Content
  const contentElement = document.getElementById('contentExample');
  contentElement.innerHTML = '<strong>New Content</strong>';
  const innerHtmlExample = document.getElementById('innerHtmlExample');
  innerHtmlExample.innerHTML = '<em>Modified with InnerHTML</em>';
  // Changing Styles
  const styleElement = document.getElementById('styleExample');
  styleElement.style.backgroundColor = 'blue';
  styleElement.style.fontSize = '20px';
  // Adding/Removing Classes
  const classElement = document.getElementById('classExample');
  classElement.classList.add('highlighted');
  classElement.classList.remove('highlighted');
  // Creating and Appending Elements
  const createElementBtn = document.getElementById('createElementBtn');
  const container = document.getElementById('container');
  createElementBtn.addEventListener('click', function () {
    const newElement = document.createElement('p');
```

```
    newElement.textContent = 'Newly Created Element';
    container.appendChild(newElement);
  });
  // Event Handling
  const clickMeBtn = document.getElementById('clickMeBtn');
  clickMeBtn.addEventListener('click', function () {
    alert('Button Clicked!');
  });
  // Removing Elements
  const removeExample = document.getElementById('removeExample');
  removeExample.parentNode.removeChild(removeExample);
  // Data Attributes
  const dataExample = document.getElementById('dataExample');
  const dataInfo = dataExample.getAttribute('data-info');
  console.log('Data Attribute:', dataInfo);
 </script>
</body>
</html>
```

आउटपुट



चित्र 17.12: HTML तत्वों में हेरफेर

व्याख्या

HTML संरचना — HTML फ़ाइल एक मूल संरचना स्थापित करती है जिसमें विभिन्न अनुभाग होते हैं, जो JavaScript के उपयोग से HTML तत्वों में हेरफेर (manipulation) को प्रदर्शित करने के लिए बनाए गए हैं। इसमें शीर्षक (headings), सूचियाँ (lists), बटन (buttons) और विशिष्ट ID तथा class वाले div तत्व शामिल हैं।

शैली — कोड में एक सरल CSS शैली शामिल है जो उन तत्वों को पीले रंग की पृष्ठभूमि के साथ उजागर करती है जिनमें "highlighted" नामक class होती है।

JavaScript — JavaScript कोड <script> टैग के अंदर शामिल किया गया है। यह विभिन्न क्रियाओं को निष्पादित करता है ताकि यह दिखाया जा सके कि HTML तत्वों में किस प्रकार हेरफेर किया जा सकता है।

- **तत्वों तक पहुँच** — यह getElementById और getElementsByClassName विधियों का उपयोग करके HTML तत्वों तक पहुँच प्राप्त करने से शुरू होता है। ये तत्व elementById और elementsByClass जैसे वेरिएबल्स में स्टोर किए जाते हैं ताकि आगे हेरफेर किया जा सके।
- **सामग्री में परिवर्तन** — यह दिखाया गया है कि तत्वों की सामग्री को कैसे बदला जा सकता है। उदाहरणस्वरूप, ID "contentExample" वाले div की सामग्री को "New Content" में बदला गया है और ID "innerHtmlExample" वाले div की सामग्री को innerHTML प्रॉपर्टी का उपयोग करते हुए बदला गया है।
- **शैली में परिवर्तन** — यह दिखाता है कि तत्वों की शैली (style) को कैसे बदला जा सकता है। ID "styleExample" वाले div की पृष्ठभूमि का रंग और फ़ॉन्ट आकार style प्रॉपर्टी के माध्यम से सेट किया गया है।
- **क्लास जोड़ना/हटाना** — कोड classElement पर classList प्रॉपर्टी का उपयोग करके क्लास को जोड़ता और हटाता है। इसमें "highlighted" क्लास को जोड़ा और फिर हटाया गया है।
- **तत्व का निर्माण और जोड़ना** — यह खंड तत्वों की गतिशील रूप से उत्पत्ति और उन्हें जोड़ने को दर्शाता है। "createElementBtn" ID वाले बटन पर क्लिक करने पर, "Newly Created Element" टेक्स्ट युक्त एक नया p तत्व तैयार किया जाता है और "container" ID वाले div में जोड़ा जाता है।
- **घटनाओं का प्रबंधन** — घटना प्रबंधन (event management) का उदाहरण "clickMeBtn" ID वाले बटन पर क्लिक इवेंट लिसनर जोड़कर दिया गया है। इस बटन को सक्रिय करने पर यह "Button Clicked!" संदेश के साथ एक अलर्ट दिखाता है।
- **तत्व हटाना** — कोड removeChild विधि का उपयोग करके "removeExample" ID वाले div को उसके मूल (parent) से हटा देता है।
- **डेटा विशेषताएँ (Data Attributes)** — यह getAttribute विधि का उपयोग करके "dataExample" div की एक डेटा विशेषता को एक्सेस करता है और उसे कंसोल पर लॉग करता है।

कार्यान्वयन — जब HTML पृष्ठ लोड होता है, तब JavaScript कोड चलाया जाता है और ऊपर वर्णित सभी HTML तत्वों में हेरफेर और घटना प्रबंधन क्रियाएँ संपन्न होती हैं।

संपूर्ण रूप से, यह कोड JavaScript के उपयोग से HTML तत्वों के साथ अंतःक्रिया और हेरफेर का एक व्यावहारिक उदाहरण प्रस्तुत करता है।

सारांश

- HTML का <canvas> तत्व और JavaScript का 2D ड्राइंग संदर्भ (context) छवि (image) में हेरफेर की अनुमति देता है।
- छवियों को लोड करने के लिए एक Image ऑब्जेक्ट बनाया जाता है और उसका src गुण सेट किया जाता है।
- छवियों को drawImage विधि का उपयोग करके कैनवास पर खींचा जा सकता है।
- छवि के पिक्सेल में परिवर्तन getImageData और putImageData के माध्यम से किया जा सकता है।

- कैनवास को इच्छित आयामों में बनाकर छवियों का आकार बदला जा सकता है।
- परिवर्तित छवियों को डेटा URL में कैनवास की सामग्री को परिवर्तित करके सहेजा जा सकता है।
- पिक्सेल डेटा में हेरफेर करके कस्टम फ़िल्टर और प्रभाव बनाए जा सकते हैं।
- जावास्क्रिप्ट में भू-स्थान (geolocation) में समर्थन की जाँच करना और प्रयोक्ता के स्थान का अनुरोध करना शामिल होता है।
- भू-स्थान विकल्प जैसे कि सटीकता और समय सीमा को कॉन्फ़िगर किया जा सकता है।
- watchPosition का उपयोग करके निरंतर स्थान ट्रैकिंग की जा सकती है।
- भू-स्थान के लिए प्रयोक्ता की अनुमति और ब्राउज़र सुरक्षा का प्रबंधन अत्यंत आवश्यक होता है।
- Google Maps एकीकरण के लिए एक API कुंजी और प्रारंभिक क्रियाएँ आवश्यक होती हैं।
- जावास्क्रिप्ट HTML तत्वों तक गतिशील रूप से पहुँच सकता है, उन्हें संशोधित और स्टाइल कर सकता है।
- जावास्क्रिप्ट में इवेंट हैंडलिंग इंटरएक्टिव वेब अनुप्रयोगों को सक्षम करती है।
- HTML में हेरफेर के दौरान तत्वों को जोड़ा, हटाया जा सकता है और उनके डेटा गुणों तक पहुँच प्राप्त की जा सकती है।

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. जावास्क्रिप्ट में हेरफेर के लिए आप छवि को कैसे लोड कर सकते हैं?
   (क) <img> तत्व का उपयोग करके
   (ख) document.createElement विधि का उपयोग करके
   (ग) <canvas> तत्व का उपयोग करके
   (घ) navigator.geolocation ऑब्जेक्ट का उपयोग करके

2. जावास्क्रिप्ट में छवि के व्यक्तिगत पिक्सेल तक पहुँचने और उन्हें संशोधित करने के लिए कौन-सी विधि उपयोग की जाती है?
   (क) getImageData
   (ख) drawImage
   (ग) putImageData
   (घ) createElement

3. जावास्क्रिप्ट में परिवर्तित छवि को सहेजने के लिए, आप कैनवास की सामग्री को डेटा URL में परिवर्तित कर सकते हैं—
   (क) getContext('2d')
   (ख) toDataURL()
   (ग) img.src
   (घ) getImageData

4. जावास्क्रिप्ट में भू-स्थान का एक उद्देश्य क्या है?
   (क) छवियों में हेरफेर करना
   (ख) कस्टम फ़िल्टर बनाना
   (ग) प्रयोक्ता की भौगोलिक स्थिति तक पहुँचना
   (घ) HTML तत्व बनाना

5. navigator.geolocation के साथ प्रयोक्ता के स्थान का अनुरोध करते समय, सफल स्थान प्राप्ति को संभालने के लिए कौन-सी विधि उपयोग होती है?
   (क) onerror
   (ख) getCurrentPosition
   (ग) addEventListener
   (घ) getElementsByClassName

6. कौन-सी जावास्क्रिप्ट विधि HTML तत्व की शैली को बदलने की अनुमति देती है?
   (क) innerHTML
   (ख) appendChild
   (ग) style
   (घ) textContent

7. HTML में किसी छवि तत्व का स्रोत URL निर्दिष्ट करने के लिए कौन-सा गुण (attribute) उपयोग किया जाता है?
   (क) alt
   (ख) src
   (ग) class
   (घ) href

8. HTML तत्व में एक class जोड़ने के लिए जावास्क्रिप्ट में आप किसका उपयोग कर सकते हैं?
   (क) classList.add()
   (ख) style.backgroundColor
   (ग) innerHTML
   (घ) getElementById

9. जावास्क्रिप्ट में छवि हेरफेर में <canvas> तत्व का उद्देश्य क्या है?
   (क) एक इमेज रोलओवर प्रभाव बनाना
   (ख) डेटा URL बनाना
   (ग) छवियों को लोड और संशोधित करना
   (घ) भू-स्थान डेटा तक पहुँचना

10. <img> तत्व में कौन-सा गुण (attribute) छवि के लिए वैकल्पिक टेक्स्ट निर्दिष्ट करता है?
    (क) src
    (ख) alt
    (ग) title
    (घ) class

ख. रिक्त स्थान भरिए

1. किसी _____ को हेरफेर करने के लिए, पहले उसे एक छवि ऑब्जेक्ट का उपयोग करके लोड करें।

2. लोड की गई छवि को कैनवास पर प्रदर्शित करने के लिए, आप 2D ड्राइंग संदर्भ की _____ विधि का उपयोग कर सकते हैं।
3. आप _____ और putImageData विधियों का उपयोग करके छवि के व्यक्तिगत पिक्सेल तक पहुँच सकते हैं और उन्हें संशोधित कर सकते हैं।
4. संशोधित छवि को सहेजने के लिए, आप _____ विधि का उपयोग करके कैनवास की सामग्री को डेटा URL में परिवर्तित कर सकते हैं।
5. जावास्क्रिप्ट में _____ प्रयोक्ता की भौगोलिक स्थिति तक उनके डिवाइस के GPS, IP पता या अन्य स्रोतों के माध्यम से पहुँचने की अनुमति देता है।
6. आप navigator.geolocation ऑब्जेक्ट का उपयोग करके प्रयोक्ता के _____ में भू-स्थान के समर्थन की जाँच कर सकते हैं।
7. जिस फ़ंक्शन द्वारा प्रयोक्ता की _____ को सफलतापूर्वक प्राप्त किया जाता है, उसे getCurrentPosition कहा जाता है।
8. किसी प्रयोक्ता की स्थिति को लगातार _____ करने के लिए, आप watchPosition विधि का उपयोग कर सकते हैं।
9. जावास्क्रिप्ट में, आप style प्रॉपर्टी का उपयोग करके किसी तत्व की शैली गुणों तक पहुँच सकते हैं और उन्हें _____ कर सकते हैं।
10. आप removeChild विधि का उपयोग करके HTML तत्व को _____ से हटा सकते हैं।

ग. सही या गलत बताइए

1. जावास्क्रिप्ट में छवि हेरफेर मुख्यतः HTML के <img> तत्व का उपयोग करके की जाती है।
2. कैनवास के लिए डेटा URL बनाने के लिए आप getElementById विधि का उपयोग कर सकते हैं।
3. जावास्क्रिप्ट में भू-स्थान प्रयोक्ता की भौगोलिक स्थिति तक उनके डिवाइस के GPS, IP पता या अन्य स्थान डेटा स्रोतों के माध्यम से पहुँचने की अनुमति देता है।
4. भू-स्थान विकल्पों में timeout प्रॉपर्टी स्थान डेटा की अधिकतम आयु को निर्दिष्ट करती है।
5. जावास्क्रिप्ट में HTML तत्व की शैली को बदलने के लिए आप classList प्रॉपर्टी का उपयोग कर सकते हैं।
6. आप createElement विधि का उपयोग करके HTML तत्व को DOM से हटा सकते हैं।
7. सामग्री में हेरफेर करते समय, textContent प्रॉपर्टी सादा टेक्स्ट (plain text) सेट करने की अनुमति देती है।
8. भू-स्थान में watchPosition विधि आपको एक बार के लिए स्थान प्राप्त करने देती है।
9. HTML तत्व की सामग्री को बदलने के लिए आप appendChild विधि का उपयोग कर सकते हैं।
10. <img> तत्व में alt गुण छवि के स्रोत URL को निर्दिष्ट करता है।

घ. लघु उत्तर प्रश्न

1. HTML में <canvas> तत्व का उद्देश्य क्या है?
2. जावास्क्रिप्ट में छवि को हेरफेर के लिए कैसे लोड किया जाता है?
3. जावास्क्रिप्ट में किसी छवि के व्यक्तिगत पिक्सेल तक पहुँचने के लिए कौन-सी विधि उपयोग होती है?
4. <img> तत्व में alt गुण का उद्देश्य क्या है?
5. JavaScript का उपयोग करके आप HTML तत्व में class कैसे जोड़ सकते हैं?
6. भू-स्थान में watchPosition विधि क्या करती है?
7. जावास्क्रिप्ट में किसी HTML तत्व की सामग्री को बदलने के लिए कौन-सी प्रॉपर्टी उपयोग की जाती है?
8. भू-स्थान विकल्पों में timeout प्रॉपर्टी का क्या कार्य है?
9. जावास्क्रिप्ट में आप नया HTML तत्व कैसे बना सकते हैं?
10. जावास्क्रिप्ट में संशोधित छवि को सेव करने के लिए कौन-सी विधि का उपयोग किया जा सकता है?

सत्र 18
HTML5 कैनवस (Canvas)

जगदीश नाम के रचनात्मक बालक ने "HTML5 कैनवस" की खोज की — कंप्यूटर पर एक जादुई चित्रपट, जिस पर वह सीधे स्क्रीन पर चित्र बना सकता था, रंग भर सकता था और सुंदर कलाकृतियाँ रच सकता था। HTML5 कैनवस की सहायता से वह आकार बना सकता था, रंग जोड़ सकता था, एनीमेशन बना सकता था और यहाँ तक कि सरल खेल (games) भी डिज़ाइन कर सकता था, जिससे वह एक डिजिटल कलाकार बन गया। यह उसके कंप्यूटर पर रचनात्मकता की एक अनंत दुनिया की तरह था, जिसने इंटरनेट को और अधिक रंगीन और मनोरंजक बना दिया। जैसा कि चित्र 18.1 में दिखाया गया है।

चित्र 18.1— HTML5 कैनवस का उपयोग करता हुआ जो

इस अध्याय में, आप जावास्क्रिप्ट में HTML5 कैनवस टैग, आकृतियाँ बनाना, ग्रेडिएंट्स (Gradients) और चित्रों का उपयोग करना सीखेंगे।

18.1 जावास्क्रिप्ट में HTML5 कैनवस टैग

HTML5 का <canvas> एलिमेंट आपको जावास्क्रिप्ट का उपयोग करके किसी वेब पेज पर सीधे ग्राफिक्स, एनीमेशन और इंटरएक्टिव सामग्री (interactive content) बनाने की अनुमति देता है।

> **क्या आप जानते हैं?**
> <canvas> टैग केवल ग्राफिक्स के लिए एक कंटेनर होता है।

HTML5 कैनवस टैग को जावास्क्रिप्ट में उपयोग करने के लिए चरणबद्ध मार्गदर्शिका इस प्रकार है—

HTML फ़ाइल बनाएँ — शुरुआत के लिए एक HTML फ़ाइल बनाएँ, उदाहरण के लिए canvas.html, और उसे अपने पसंदीदा टेक्स्ट एडिटर (text editor) में खोलें।

<canvas> एलिमेंट जोड़ें — HTML फ़ाइल के अंदर वहाँ <canvas> एलिमेंट जोड़ें, जहाँ आप ग्राफिक्स बनाना चाहते हैं। इसे एक id एट्रिब्यूट दें जिससे इसे जावास्क्रिप्ट के माध्यम से आसानी से एक्सेस किया जा सके, और साथ ही width और height एट्रिब्यूट का उपयोग करके कैनवस का आकार परिभाषित करें, जैसा कि चित्र 18.2 में दिखाया गया है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Example</title>
</head>
<body>
    <canvas id="myCanvas" width="400" height="200"></canvas>
</body>
</html>
```

चित्र 18.2— जावास्क्रिप्ट में <canvas> टैग का उपयोग करना

जावास्क्रिप्ट में कैनवस तक पहुँचें — कैनवस पर कार्य करने के लिए आपको इसकी id के माध्यम से इसका संदर्भ (reference) प्राप्त करना होगा। निम्न कोड को <script> टैग के अंदर, </body> टैग से ठीक पहले, या किसी बाह्य जावास्क्रिप्ट फ़ाइल में रखें जिसे आपने HTML से जोड़ा हो।

javascript

```
const canvas = document.getElementById('myCanvas');
const context = canvas.getContext('2d');
```

getContext('2d') विधि का उपयोग 2D रेंडरिंग कॉन्टेक्स्ट प्राप्त करने के लिए किया जाता है, जो कैनवस पर चित्र बनाने के लिए विधियाँ प्रदान करता है।

कैनवस पर चित्र बनाना — अब जब आपके पास कैनवस और उसका कॉन्टेक्स्ट है, तो आप context द्वारा प्रदान की गई विभिन्न ड्रॉइंग विधियों का उपयोग कर सकते हैं। उदाहरण के लिए, आप एक आयत इस प्रकार बना सकते हैं —

javascript

```
context.fillStyle = 'blue'; // रंग नीला सेट करें
context.fillRect(50, 50, 100, 80); // भरी हुई आयत बनाएं
```

यह कोड भरने का रंग नीला निर्धारित करता है और कैनवस पर एक भरी हुई आयत बनाता है।

अधिक ग्राफिक्स जोड़ना — आप अधिक ग्राफिक्स, आकृतियाँ, टेक्स्ट, चित्र और यहाँ तक कि एनीमेशन भी बना सकते हैं, context की ड्रॉइंग विधियों को बार-बार उपयोग करके। जैसे fillRect, strokeRect, fillText, drawImage आदि विधियों का उपयोग करके अपने इच्छित दृश्य बनाएँ।

इंटरएक्टिविटी — अपने कैनवस को इंटरएक्टिव बनाने के लिए, प्रयोक्ता क्रियाओं पर प्रतिक्रिया देने हेतु ईवेंट लिस्नर्स (event listeners) जोड़ें। उदाहरण के लिए, आप canvas.addEventListener विधि का उपयोग करके माउस क्लिक या कुंजी दबाने की घटनाओं को कैप्चर कर सकते हैं और उसके अनुसार कैनवस को अपडेट कर सकते हैं।

HTML पृष्ठ चलाएँ — अपनी HTML फ़ाइल को वेब ब्राउज़र में खोलें और उस पर बनाए गए ग्राफिक्स को देखें। उदाहरण— canvas.html खोलें।

उन्नत सुविधाओं की खोज करें — <canvas> एलिमेंट और 2D कॉन्टेक्स्ट ग्राफिक्स बनाने के लिए शक्तिशाली उपकरण प्रदान करते हैं। जब आप कैनवस प्रोग्रामिंग में पारंगत हो जाएँगे, तो आप एनीमेशन, ट्रांसफॉर्मेशन, ग्रेडिएंट्स और पिक्सल हेरफेर जैसे उन्नत विषयों का अन्वेषण कर सकते हैं।

ध्यान दें कि HTML5 <canvas> 2D ग्राफिक्स के लिए उपयुक्त है। अधिक उन्नत 3D ग्राफिक्स के लिए, WebGL एक लोकप्रिय तकनीक है जो HTML5 पर आधारित है और वेब एप्लिकेशन के लिए हार्डवेयर-संवर्धित 3D रेंडरिंग क्षमताएँ प्रदान करती है।

प्रयोगात्मक अभ्यास 18.1
निम्नलिखित के लिए एक जावास्क्रिप्ट कोड लिखिए—
• <canvas> एलिमेंट जोड़ना
• जावास्क्रिप्ट में कैनवस तक पहुँच
• कैनवस पर चित्र बनाना

18.2 आकृतियाँ बनाना (Drawing Shapes)

18.2.1 आकृतियाँ बनाना – आयतें (Rectangles)

- HTML5 कैनवस पर आयत बनाने के लिए, आप 2D रेंडरिंग कॉन्टेक्स्ट की fillRect() और strokeRect() विधियों का उपयोग कर सकते हैं।
- fillRect(x, y, width, height) — यह निर्दिष्ट चौड़ाई और ऊँचाई के साथ (x, y) स्थान से शुरू होकर एक भरी हुई आयत बनाता है।
- strokeRect(x, y, width, height) — यह दिए गए मापदंडों के साथ आयत की रूपरेखा (स्ट्रोक) बनाता है।

18.2.2 आकृतियाँ बनाना – वृत्त और चाप (Circles and Arcs)

- वृत्त और चाप बनाने के लिए आप arc() विधि का उपयोग कर सकते हैं।
- arc(x, y, radius, startAngle, endAngle, anticlockwise) — यह (x, y) केन्द्र बिंदु के साथ निर्दिष्ट त्रिज्या का एक वृत्ताकार चाप बनाता है। startAngle और endAngle चाप की शुरुआत और अंत को परिभाषित करते हैं, तथा anticlockwise यह निर्धारित करता है कि चाप को वामावर्त (anticlockwise) या दक्षिणावर्त (clockwise) दिशा में खींचा जाएगा।

18.2.3 पथ और रेखाएँ बनाना (Drawing Paths and Lines)

- कैनवस API आपको कस्टम पथ बनाने और रेखाएँ खींचने की अनुमति देता है, इसके लिए आप beginPath(), moveTo(), lineTo(), और closePath() विधियों का उपयोग कर सकते हैं।
- beginPath() — एक नया पथ शुरू करता है।
- moveTo(x, y) — (x, y) स्थान पर बिना रेखा खींचे ड्राइंग स्थिति को ले जाता है।
- lineTo(x, y) — वर्तमान स्थिति से (x, y) तक सीधी रेखा खींचता है।
- closePath() — वर्तमान स्थिति को प्रारंभिक स्थिति से जोड़कर पथ को बंद करता है।
- इन विधियों का उपयोग करके आप कई रेखाओं को जोड़कर जटिल आकृतियाँ और पथ बना सकते हैं।

18.2.4 रेखा कैप (Line Caps) और रेखा जोड़ (Line Joins)—

- रेखा कैप और जोड़ वे गुण (properties) हैं जो रेखाओं के सिरों और उनके प्रतिछेदन (intersections) की उपस्थिति को प्रभावित करते हैं।
- **रेखा कैप (Line Caps)—**
  - context.lineCap — यह गुण "butt", "round", या "square" में से किसी एक पर सेट किया जा सकता है। यह निर्धारित करता है कि रेखाओं के सिरे कैसे दिखाई देंगे।
  - "butt" — डिफ़ॉल्ट मान। रेखा का अंत बिना किसी अतिरिक्त सजावट के होता है।
  - "round" — रेखा के सिरे को अर्द्धवृत्तीय (semicircular) आकार में गोल कर देता है।
  - "square" — रेखा के सिरे को चौकोर (square) आकार में समतल कर देता है।

- **रेखा जोड़ (Line Joins)—**
  - context.lineJoin — यह गुण "round", "bevel", या "miter" में से किसी एक पर सेट किया जा सकता है। यह निर्धारित करता है कि कोणों पर मिलती हुई रेखाएँ किस प्रकार जोड़ी जाएँगी।
  - "round" — रेखाओं को गोल कोने के साथ जोड़ता है।
  - "bevel" — रेखाओं को समतल (beveled) कोने के साथ जोड़ता है।
  - "miter" — कोने पर विस्तारित नोक (miter) के साथ रेखाओं को जोड़ता है।
- context.miterLimit — यह गुण उस सीमा को निर्धारित करता है जिसके बाद miter जोड़ काट दिए जाते हैं। यह केवल तब लागू होता है जब lineJoin का मान "miter" हो।

ये अवधारणाएँ HTML5 कैनवस (Canvas) पर विभिन्न आकृतियों और ग्राफिकल तत्वों को बनाने के लिए मौलिक हैं। इन विधियों और गुणों को संयोजित करके आप जटिल रेखाचित्र और दृश्य प्रभाव बना सकते हैं, जैसा कि **चित्र 18.3** और **चित्र 18.4** में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Example</title>
</head>
<body>
    <canvas id="myCanvas" width="400" height="200"></canvas>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Drawing Example</title>
    <style>
        canvas {
            border: 1px solid ■black;
        }
    </style>
</head>
<body>

</body>
</html>
```

चित्र 18.3 — HTML5 में आकृतियाँ बनाना

अभ्यास 18.2

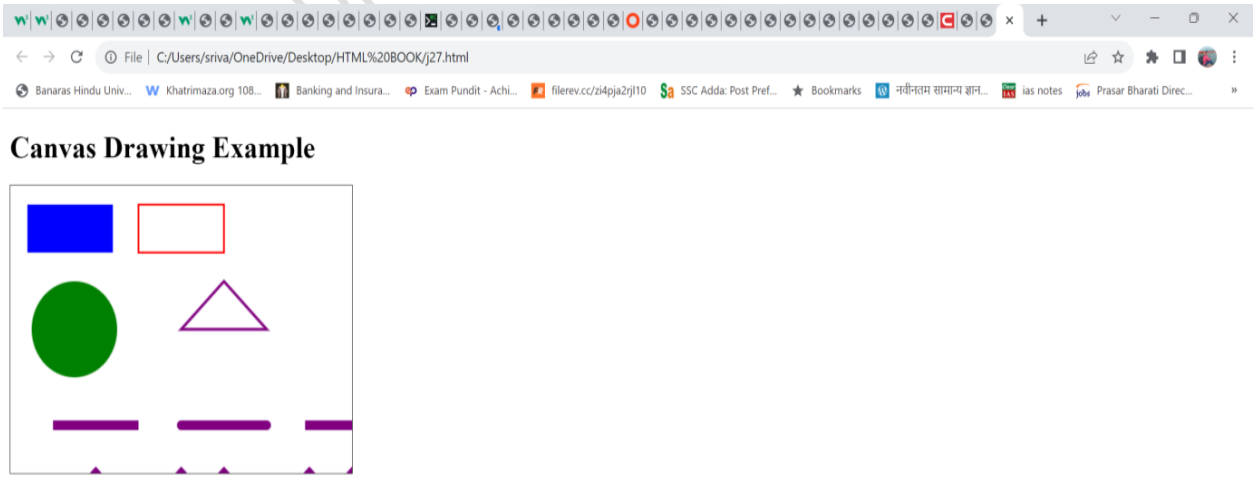
- वृत्त और आर्क (arc) बनाने की विधि का नाम लिखिए।
- कस्टम पथ (custom paths) बनाने और रेखाएँ खींचने की विधि का नाम लिखिए।
- Line Caps में "butt", "round", या "square" सेट करने वाले गुण का नाम लिखिए।

```
<body>
    <h1>Canvas Drawing Example</h1>
    <canvas id="myCanvas" width="400" height="300"></canvas>

    <script>
        const canvas = document.getElementById('myCanvas');
        const ctx = canvas.getContext('2d');

        // Drawing a filled rectangle
        ctx.fillStyle = 'blue';
        ctx.fillRect(20, 20, 100, 50);

        // Drawing the outline of a rectangle
        ctx.strokeStyle = 'red';
        ctx.lineWidth = 2;
        ctx.strokeRect(150, 20, 100, 50);

        // Drawing a circle (arc)
        ctx.beginPath();
        ctx.arc(75, 150, 50, 0, Math.PI * 2, false); // Full circle
        ctx.fillStyle = 'green';
        ctx.fill();

        // Drawing a path with lines
        ctx.beginPath();
        ctx.moveTo(200, 150); // Starting point
        ctx.lineTo(250, 100); // Line to point
        ctx.lineTo(300, 150); // Line to point
        ctx.closePath(); // Close the path
        ctx.strokeStyle = 'purple';
        ctx.lineWidth = 3;
        ctx.stroke();

        // Setting line caps and line joins
        ctx.lineWidth = 10;

        // Line with butt cap
        ctx.lineCap = 'butt';
        ctx.beginPath();
        ctx.moveTo(50, 250);
        ctx.lineTo(150, 250);
        ctx.stroke();

        // Line with round cap
        ctx.lineCap = 'round';
        ctx.beginPath();
        ctx.moveTo(200, 250);
        ctx.lineTo(300, 250);
        ctx.stroke();
```

```
        // Line with square cap
        ctx.lineCap = 'square';
        ctx.beginPath();
        ctx.moveTo(350, 250);
        ctx.lineTo(450, 250);
        ctx.stroke();

        // Line joins
        ctx.lineJoin = 'round';
        ctx.beginPath();
        ctx.moveTo(100, 300);
        ctx.lineTo(150, 350);
        ctx.lineTo(200, 300);
        ctx.stroke();

        ctx.lineJoin = 'bevel';
        ctx.beginPath();
        ctx.moveTo(250, 300);
        ctx.lineTo(300, 350);
        ctx.lineTo(350, 300);
        ctx.stroke();

        ctx.lineJoin = 'miter';
        ctx.miterLimit = 2; // Miter limit for sharp miter joins
        ctx.beginPath();
        ctx.moveTo(400, 300);
        ctx.lineTo(450, 350);
        ctx.lineTo(500, 300);
        ctx.stroke();
    </script>
</body>
</html>
```

चित्र 18.4 — HTML5 में जटिल रेखाचित्र और दृश्य प्रभाव बनाना

आउटपुट



चित्र 18.5 — कैनवस रेखाचित्र उदाहरण का आउटपुट

कोड विवरण —

भरा हुआ आयत (Filled Rectangle) बनाना— fillRect विधि का उपयोग (20, 20) निर्देशांक पर 100 पिक्सेल चौड़ाई और 50 पिक्सेल ऊँचाई वाले नीले रंग का भरा हुआ आयत बनाने के लिए किया गया है।

आयत की रूपरेखा (Outline) बनाना— strokeRect विधि का उपयोग (150, 20) निर्देशांक पर उसी चौड़ाई और ऊँचाई के साथ लाल रंग का बॉर्डर (सीमा रेखा) बनाने के लिए किया गया है। इसकी लाइन चौड़ाई 2 पिक्सेल है।

वृत्त (Arc) बनाना— arc विधि का उपयोग (75, 150) केंद्र पर 50 पिक्सेल त्रिज्या वाला हरा रंग का वृत्त (arc) बनाने के लिए किया गया है। यह 0 रेडियन से Math.PI * 2 रेडियन (360 डिग्री) तक का पूरा वृत्त बनाता है। fill विधि का उपयोग इस वृत्त को भरने के लिए किया गया है।

रेखाओं से पथ (Path) बनाना— beginPath, moveTo, lineTo और closePath विधियों का उपयोग रेखाओं से जुड़ा पथ (path) बनाने के लिए किया गया है। यह (200, 150) से शुरू होकर (250, 100) फिर (300, 150) तक जाता है, जिससे एक त्रिभुज बनता है। stroke विधि का उपयोग इसे बैंगनी रंग और 3 पिक्सेल की लाइन चौड़ाई के साथ आउटलाइन करने के लिए किया गया है।

लाइन कैप और लाइन जोड़ सेट करना—

- कोड में lineCap गुण का उपयोग करके विभिन्न रेखा सिरों की शैलियाँ (butt, round, और square) प्रदर्शित की गई हैं। तीन रेखाएँ विभिन्न लाइन कैप्स के साथ खींची गई हैं।
- साथ ही, lineJoin गुण का उपयोग करके विभिन्न रेखा जोड़ों की शैलियाँ (round, bevel, और miter) भी प्रदर्शित की गई हैं। तीन पथ बनाए गए हैं जिनमें तीव्र miter जोड़, समतल bevel जोड़, और गोल round जोड़ दिखाए गए हैं। miterLimit गुण miter जोड़ों की तीव्रता को नियंत्रित करता है।

आप इन सभी रेखाचित्र कार्यों के परिणाम HTML5 कैनवस तत्व पर देख सकते हैं। कोड के प्रत्येक भाग की व्याख्या इस बात को स्पष्ट करती है कि वह कैनवस पर विभिन्न चित्रकारी प्रभावों को कैसे प्राप्त करता है।

18.3 ग्रेडिएंट्स (Gradients)

HTML5 कैनवस में ग्रेडिएंट्स आकृति या पथ के अंदर रंग या शैली के मृदु संक्रमण (smooth transitions) बनाने की अनुमति देते हैं। ग्रेडिएंट्स रैखिक (linear) या रेडियल (radial) हो सकते हैं और सामान्यतः चित्रों में गहराई और दृश्य आकर्षण जोड़ने के लिए उपयोग किए जाते हैं। कैनवस में ग्रेडिएंट्स के दो मुख्य प्रकारों का वर्णन नीचे दिया गया है—

18.3.1 रैखिक ग्रेडिएंट्स (Linear Gradients)

रैखिक ग्रेडिएंट रंगों या शैलियों का एक सीधी रेखा के साथ मृदु संक्रमण होता है।

- रैखिक ग्रेडिएंट बनाने के लिए आपको कम से कम दो रंग स्टॉप्स (color stops — विशिष्ट रंगों वाले बिंदु) और ग्रेडिएंट रेखा के प्रारंभ तथा अंत बिंदु परिभाषित करने होते हैं।
- createLinearGradient विधि का उपयोग एक रैखिक ग्रेडिएंट ऑब्जेक्ट बनाने के लिए किया जाता है, इसके बाद रंग स्टॉप्स और ग्रेडिएंट दिशा को सेट किया जाता है।
- रंग स्टॉप्स को ग्रेडिएंट ऑब्जेक्ट की addColorStop विधि से परिभाषित किया जाता है। प्रत्येक रंग स्टॉप में एक स्थिति (0 और 1 के बीच का मान) और एक रंग मान होता है।
- एक बार ग्रेडिएंट परिभाषित हो जाने के बाद, आप इसे आकृतियाँ बनाते समय fill या stroke शैली के रूप में उपयोग कर सकते हैं।

18.3.2 रेडियल ग्रेडिएंट्स (Radial Gradients)—

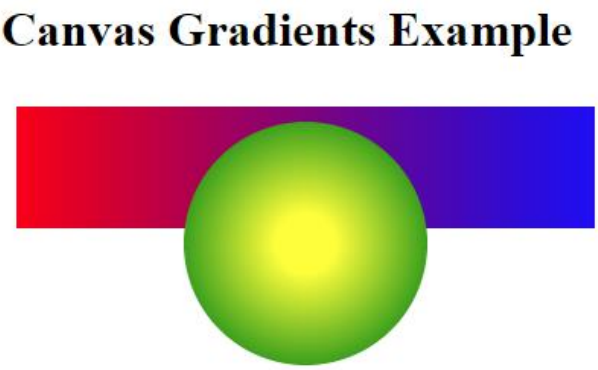
- रेडियल ग्रेडिएंट एक वृत्त के केंद्र से बाहर की ओर सभी दिशाओं में रंगों या शैलियों का एक सुगम बदलाव (transition) होता है।

- रेडियल ग्रेडिएंट बनाने के लिए, आपको कम से कम दो रंग-स्थान (color stops) तथा भीतरी और बाहरी वृत्तों के केंद्र व त्रिज्या को परिभाषित करना होता है।
- लीनियर ग्रेडिएंट्स (linear gradients) की तरह ही, आप रेडियल ग्रेडिएंट ऑब्जेक्ट बनाने के लिए createRadialGradient विधि का प्रयोग करते हैं, तथा रंग-स्थान और वृत्तों की स्थितियाँ और त्रिज्याएँ सेट करते हैं।
- रेडियल ग्रेडिएंट्स का उपयोग कोमल चमक, चमकदार सतह, या रेडियल पृष्ठभूमि जैसे प्रभाव उत्पन्न करने के लिए किया जा सकता है, जैसा कि चित्र 18.6 में दर्शाया गया है।

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Canvas Gradients Example</title>
</head>
<body>
    <h1>Canvas Gradients Example</h1>
    <!-- Create a canvas element -->
    <canvas id="myCanvas" width="400" height="200"></canvas>
    <script>
        // Get the canvas element and its 2D rendering context
        const canvas = document.getElementById('myCanvas');
        const ctx = canvas.getContext('2d');
        // Create a linear gradient (left to right)
        const linearGradient = ctx.createLinearGradient(0, 0, 400, 0); // (x0, y0, x1, y1)
        // Add color stops to the linear gradient
        linearGradient.addColorStop(0, 'red');    // Start color at the left
        linearGradient.addColorStop(1, 'blue');   // End color at the right
        // Fill a rectangle with the linear gradient
        ctx.fillStyle = linearGradient;
        ctx.fillRect(10, 10, 380, 80);
        // Create a radial gradient
        const radialGradient = ctx.createRadialGradient(200, 100, 20, 200, 100, 100);
        // (x0, y0, r0, x1, y1, r1)
        // Add color stops to the radial gradient
        radialGradient.addColorStop(0, 'yellow');  // Inner circle color
        radialGradient.addColorStop(1, 'green');   // Outer circle color
        // Fill a circle with the radial gradient
        ctx.fillStyle = radialGradient;
        ctx.beginPath();
        ctx.arc(200, 100, 80, 0, Math.PI * 2);
        ctx.fill();
    </script>
</body>
</html>
```

चित्र 18.6— कैनवास रेडियल ग्रेडिएंट्स का उदाहरण

आउटपुट



चित्र 18.7— कैनवास रेडियल ग्रेडिएंट्स उदाहरण का आउटपुट

कोड व्याख्या

हम HTML5 डॉक्यूमेंट संरचना से प्रारंभ करते हैं और "myCanvas" आईडी के साथ एक कैनवास एलिमेंट बनाते हैं तथा इसकी चौड़ाई और ऊँचाई सेट करते हैं।

जावास्क्रिप्ट अनुभाग में—

- हम getElementById और getContext का उपयोग करके कैनवास एलिमेंट और इसका 2D रेंडरिंग कॉन्टेक्स्ट (ctx) प्राप्त करते हैं।
- हम ctx.createLinearGradient का उपयोग करके एक लीनियर ग्रेडिएंट (linearGradient) बनाते हैं, जिसकी निर्देशांक (0, 0) (प्रारंभ) से (400, 0) (अंत) तक होती हैं।
- हम linearGradient.addColorStop का उपयोग करके लीनियर ग्रेडिएंट में रंग जोड़ते हैं। इस मामले में, हम बाएँ (0) से लाल रंग से शुरू करते हैं और दाएँ (1) तक नीले रंग में संक्रमण करते हैं।
- हम ctx.fillStyle के माध्यम से संदर्भ की फिल शैली को लीनियर ग्रेडिएंट पर सेट करते हैं।
- हम fillRect का उपयोग करके एक आयत को लीनियर ग्रेडिएंट से भरते हैं।
- सके बाद, हम ctx.createRadialGradient का उपयोग करके एक रेडियल ग्रेडिएंट (radialGradient) बनाते हैं जिसमें विशिष्ट निर्देशांक और त्रिज्याएँ होती हैं।
- हम रेडियल ग्रेडिएंट में रंग-स्थान जोड़ते हैं, जिसमें केंद्र (0) पर पीला और बाहरी वृत्त (1) पर हरा होता है।
- हम ctx.fillStyle को रेडियल ग्रेडिएंट पर सेट करते हैं।
- ctx.beginPath का उपयोग करके एक वृत्त खींचना प्रारंभ करते हैं।
- हम ctx.arc के माध्यम से (200, 100) निर्देशांकों के केंद्र और 80 की त्रिज्या के साथ एक वृत्त को परिभाषित करते हैं।

अंततः, हम रेडियल ग्रेडिएंट से वृत्त को भरते हैं, जिससे कैनवास पर एक सौंदर्यपरक ग्रेडिएंट भरा हुआ वृत्त बनता है।

18.4 कैनवास टैग के साथ चित्रों का उपयोग करना (Using Images with the Canvas Tag)—

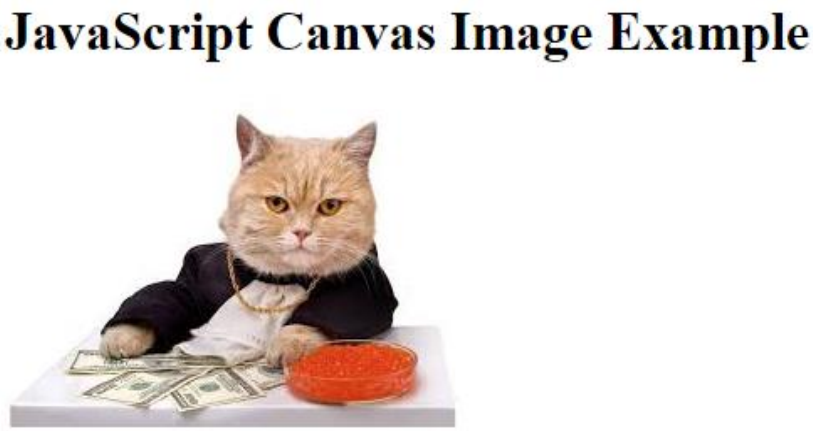
HTML5 का <canvas> टैग आपको एक वेब पृष्ठ पर सीधे ग्राफ़िक्स, जिनमें चित्र भी शामिल हैं, को चित्रित करने और हेरफेर करने की अनुमति देता है, जैसा कि चित्र 18.8 में दर्शाया गया है।

आप कैनवास के अंदर चित्रों को लोड, प्रदर्शित और इंटरैक्ट कर सकते हैं, जिससे यह इंटरएक्टिव वेब अनुप्रयोगों के निर्माण के लिए एक बहुउपयोगी उपकरण बन जाता है।

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Canvas Image Example</title>
</head>
<body>
    <h1>JavaScript Canvas Image Example</h1>
    <canvas id="canvas" width="400" height="200"></canvas>
    <script>
        const canvas = document.getElementById("canvas")
        const ctx = canvas.getContext("2d")
        // Create an image object
        const img = new Image()
        img.src = 'sample-image.jpg'
        // Draw the image on the canvas when it's loaded
        img.onload = function() {
            ctx.drawImage(img, 0, 0)
        }
    </script>
</body>
</html>
```

चित्र 18.8— जावास्क्रिप्ट कैनवास चित्र का उदाहरण

आउटपुट



चित्र 18.9— जावास्क्रिप्ट कैनवास चित्र उदाहरण का आउटपुट

18.4.2 चित्र पैटर्न्स (Image Patterns)—

चित्र पैटर्न में कैनवास के अंदर एक दोहराया जाने वाला पैटर्न या बनावट के रूप में एक चित्र का उपयोग किया जाता है। आप createPattern विधि का उपयोग करके एक पैटर्न बना सकते हैं, जो एक चित्र को इनपुट के रूप में लेती है और एक पैटर्न ऑब्जेक्ट लौटाती है, जैसा कि चित्र 18.10 में दिखाया गया है।

यह पैटर्न फिर कैनवास पर आकृतियाँ या पथ खींचते समय एक फिल स्टाइल के रूप में उपयोग किया जा सकता है।

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Canvas Image Pattern Example</title>
</head>
<body>
    <canvas id="canvas" width="400" height="200"></canvas>
    <script>
        const canvas = document.getElementById('canvas');
        const ctx = canvas.getContext('2d');

        // Create an image pattern
        const img = new Image();
        img.src = 'lamp.jpg';

        img.onload = function() {
            const pattern = ctx.createPattern(img, 'repeat');
            ctx.fillStyle = pattern;
            ctx.fillRect(0, 0, canvas.width, canvas.height);

            // Draw a circle with a different style
            ctx.beginPath();
            ctx.arc(canvas.width / 2, canvas.height / 2, 40, 0, Math.PI * 2);
            ctx.strokeStyle = 'red'; // Set the stroke color to red
            ctx.lineWidth = 5; // Set the line width to 5 pixels
            ctx.stroke(); // Draw the circle border
        };
    </script>
</body>
</html>
```

चित्र 18.10— इनपुट के रूप में चित्र और परिणामस्वरूप एक पैटर्न ऑब्जेक्ट

आउटपुट

चित्र 18.11— इनपुट चित्र का आउटपुट एक पैटर्न ऑब्जेक्ट के रूप में

अपनी प्रगति जाँचें

क. बहुविकल्पीय प्रश्न

1. HTML5 के <canvas> एलिमेंट का उद्देश्य क्या है?
   (क) वीडियो चलाना
   (ख) ग्राफ़िक्स और ऐनीमेशन बनाना
   (ग) फॉर्म बनाना
   (घ) टेक्स्ट प्रदर्शित करना

2. जावास्क्रिप्ट में कैनवास के लिए 2D रेंडरिंग कॉन्टेक्स्ट प्राप्त करने के लिए आप क्या उपयोग करेंगे?
   (क) document.getContext('canvas')
   (ख) document.canvas.get2DContext()
   (ग) document.querySelector('<canvas>')
   (घ) canvas.getContext('2d')

3. HTML5 कैनवास पर एक भरा हुआ आयत खींचने के लिए कौन-सी विधि प्रयोग होती है?
   (क) drawRect
   (ख) fillRect
   (ग) strokeRect
   (घ) drawFilledRect

4. canvas प्रोग्रामिंग में createLinearGradient विधि का उद्देश्य क्या है?
   (क) एक दोहराया जाने वाला पैटर्न बनाना
   (ख) एक वृत्त खींचना
   (ग) एक लीनियर ग्रेडिएंट बनाना
   (घ) कैनवास को साफ़ करना

5. canvas चित्रण में रेखाओं के सिरों की शैली सेट करने के लिए कौन-सी प्रॉपर्टी प्रयुक्त होती है?
   (क) lineCap
   (ख) lineJoin
   (ग) lineStyle
   (घ) lineEnds

6. canvas ऐनीमेशन में टाइमर का प्रमुख कार्य क्या है?
   (क) वस्तुओं को कैनवास पर खींचना
   (ख) कीबोर्ड इनपुट को संभालना
   (ग) कैनवास को साफ़ करना
   (घ) ऐनीमेशन फ़्रेम के समय को नियंत्रित करना

7. जावास्क्रिप्ट में requestAnimationFrame का उपयोग करके एक सरल ऐनीमेशन कैसे बनाया जा सकता है?
   (क) clearRect विधि का उपयोग करके

(ख) माउस ईवेंट्स को संभालकर
(ग) एक लूप में requestAnimationFrame को कॉल करके
(घ) कैनवास का आकार बदलकर

8. कीबोर्ड इनपुट के साथ कैनवास ऐनीमेशन में keydown ईवेंट का उद्देश्य क्या है?
(क) ऐनीमेशन प्रारंभ करना
(ख) कुंजी दबाने को संभालना
(ग) कुंजी छोड़ने को संभालना
(घ) कैनवास को साफ़ करना

9. canvas आधारित ऐनीमेशन में माउस क्लिक को कैप्चर करने के लिए कौन-सा ईवेंट उपयोग किया जाता है?
(क) mousedown
(ख) mousemove
(ग) mouseup
(घ) keydown

10. canvas प्रोग्रामिंग में टच ईवेंट्स का मुख्य उद्देश्य क्या है?
(क) कीबोर्ड इनपुट को पहचानना
(ख) टच-सक्षम उपकरणों पर इंटरैक्शन को कैप्चर करना
(ग) कैनवास को साफ़ करना
(घ) जटिल आकृतियाँ बनाना

ख. रिक्त स्थान भरिए

1. HTML5 का <canvas> एलिमेंट ग्राफ़िक्स, ऐनीमेशन और इंटरैक्टिव सामग्री को सीधे एक वेब पृष्ठ पर _____ का उपयोग करके चित्रित करने की अनुमति देता है।
2. जावास्क्रिप्ट में कैनवास के साथ कार्य करने के लिए, आपको उसकी _____ के माध्यम से संदर्भ प्राप्त करना होता है।
3. canvas प्रोग्रामिंग में 2D रेंडरिंग कॉन्टेक्स्ट प्राप्त करने के लिए प्रयुक्त विधि है _____।
4. HTML5 कैनवास पर एक भरा हुआ आयत खींचने के लिए आप _____ विधि का उपयोग कर सकते हैं।
5. fillRect विधि निर्देशांक स्थिति, चौड़ाई और _____ के लिए पैरामीटर लेती है।
6. lineCap प्रॉपर्टी को “butt”, “round” या “square” सेट किया जा सकता है, जैसे कि चौकोर आकृति के लिए “_____”।
7. canvas API आपको _____ जैसी विधियों का उपयोग करके कस्टम पथ बनाने और रेखाएँ खींचने की अनुमति देती है।
8. canvas ड्राइंग में एक नया पथ प्रारंभ करने की विधि है _____।
9. पथ चित्रण स्थिति को बिना रेखा खींचे एक निर्दिष्ट बिंदु पर ले जाने की विधि है _____।
10. canvas पर चित्र खींचने के लिए प्रयुक्त विधि है _____।

ग. सही या गलत बताइए

1. HTML5 का <canvas> एलिमेंट मुख्यतः वेब पृष्ठों पर फॉर्म बनाने के लिए प्रयुक्त होता है।
2. getContext('2d') विधि जावास्क्रिप्ट में कैनवास के लिए 3D रेंडरिंग कॉन्टेक्स्ट प्राप्त करने के लिए प्रयुक्त होती है।

3. strokeRect विधि HTML5 कैनवास पर आयत का भरा हुआ क्षेत्र खींचने के लिए प्रयुक्त होती है।
4. arc विधि कैनवास पर सीधी रेखाएँ खींचने के लिए उपयोग की जाती है।
5. लाइन कैप्स और जॉइन्स वे प्रॉपर्टीज़ हैं जो कैनवास ड्राइंग में रेखाओं के सिरों और उनके प्रतिच्छेदों की उपस्थिति को प्रभावित करती हैं।
6. beginPath विधि कैनवास प्रोग्रामिंग में एक पथ को बंद करने के लिए प्रयुक्त होती है।
7. HTML5 कैनवास में रेडियल ग्रेडिएंट का उपयोग रंगों के सीधे रेखा में संक्रमण हेतु किया जाता है।
8. drawImage विधि कैनवास पर टेक्स्ट को खींचने के लिए प्रयुक्त होती है।
9. चित्र स्केलिंग का तात्पर्य कैनवास पर चित्र खींचते समय उसकी चौड़ाई और ऊँचाई को बदलना होता है।
10. सरल कैनवास ऐनीमेशन में, टाइमर का उपयोग ऐनीमेशन फ़्रेम के समय को नियंत्रित करने के लिए किया जाता है।

घ. लघु उत्तर प्रश्न

1. HTML5 के <canvas> एलिमेंट का उद्देश्य क्या है?
2. जावास्क्रिप्ट में <canvas> एलिमेंट के लिए 2D रेंडरिंग कॉन्टेक्स्ट कैसे प्राप्त करते हैं?
3. cavas पर भरा हुआ आयत खींचने के लिए कौन-सी विधि प्रयुक्त होती है?
4. <canvas> एलिमेंट का आकार परिभाषित करने के लिए कौन-कौन से गुण (attributes) उपयोग किए जाते हैं?
5. HTML5 कैनवास में लीनियर ग्रेडिएंट कैसे बनाया जाता है?
6. cavas ड्राइंग में lineCap प्रॉपर्टी का क्या उद्देश्य होता है?
7. प्रत्येक ऐनीमेशन फ़्रेम से पहले कैनवास को कैसे साफ़ किया जाता है?
8. canvas ऐनीमेशन में माउस इंटरैक्शन का पता लगाने के लिए कौन-कौन से ईवेंट लिसनर्स उपयोग किए जा सकते हैं?
9. HTML5 कैनवास में रेडियल ग्रेडिएंट कैसे बनाया जाता है?
10. canvas ड्राइंग में arc() विधि का क्या उद्देश्य है?

उत्तर कुंजी

मॉड्यूल 1. वेब विकास की मूल बातें (Web Development Essentials)

सत्र 1. सूचना प्रौद्योगिकी (IT) और सूचना प्रौद्योगिकी सक्षम सेवाएँ (ITeS)

क. बहुविकल्पीय प्रश्न

1. (ख) 2. (ग) 3. (ग) 4. (घ) 5. (ख) 6. (घ) 7. (ग) 8. (ग) 9. (घ) 10. (क)

ख. रिक्त स्थान भरिए

1. Information,
2. Business process
3. Storage devices
4. Devices, data
5. Computing
6. Automation
7. Cloud-based
8. Web
9. Video games
10. Blockchain

ग. सही या गलत बताइए

1. (सही)
2. (सही)
3. (ग़लत)
4. (ग़लत)
5. (सही)
6. (ग़लत)
7. (सही)
8. (ग़लत)
9. (ग़लत)
10. (ग़लत)

सत्र 2. वेब डिज़ाइन और विकास (Web Design and Development)

क. बहुविकल्पीय प्रश्न

1. (ग) 2. (ग) 3. (ख) 4. (क) 5. (ख) 6. (ग) 7. (क) 8. (घ) 9. (क) 10. (ग)

ख. रिक्त स्थान भरिए

1. Front-end
2. Website
3. Design
4. Stages
5. User interface
6. Store
7. Website
8. Static
9. Browsers
10. Web design

ग. सही या गलत बताइए

1. (ग़लत)
2. (सही)
3. (ग़लत)
4. (ग़लत)
5. (सही)
6. (सही)
7. (ग़लत)
8. (ग़लत)
9. (ग़लत)
10. (सही)

सत्र 3. वेब डिज़ाइन और विकास उपकरण (Web Design and Development Tools)

क. बहुविकल्पीय प्रश्न

1. (ग) 2. (ग) 3. (ग) 4. (ख) 5. (ग) 6. (ख) 7. (ग) 8. (ग) 9. (घ) 10. (ग)

ख. रिक्त स्थान भरिए

1. Structure
2. Responsive
3. Interactivity
4. Data security
5. Websites
6. HTML
7. Web
8. Javascript
9. High-level
10. SQL

ग. सही या गलत बताइए

1. (ग़लत)
2. (सही)
3. (ग़लत)
4. (ग़लत)
5. (सही)
6. (ग़लत)
7. (ग़लत)
8. (ग़लत)
9. (सही)
10. (सही)

सत्र 4. आईटी के अनुप्रयोग विकास मॉडल (Application Development Models of IT)

क. बहुविकल्पीय प्रश्न

1. (ख) 2. (ख) 3. (क) 4. (ख) 5. (ख) 6. (ग) 7. (ख) 8. (ख) 9. (क) 10. (ग)

ख. रिक्त स्थान भरिए

1. Cycles or iterations
2. Shared
3. Plans
4. Risk Management
5. Change
6. Verification
7. Repository
8. Workflow
9. Feedback
10. Lifecycle

ग. सही या गलत बताइए

1. (ग़लत)
2. (ग़लत)
3. (सही)
4. (ग़लत)
5. (ग़लत)
6. (ग़लत)
7. (ग़लत)
8. (सही)
9. (ग़लत)

10. (ग़लत)

सत्र 5. वेब डिज़ाइन विनिर्देश (Web Designing Specifications)

क. बहुविकल्पीय प्रश्न

1. (ख) 2. (ग) 3. (क) 4. (ख) 5. (ख) 6. (ख) 7. (ग) 8. (घ) 9. (ख) 10. (ख)

ख. रिक्त स्थान भरिए

1. Risks
2. End-user
3. Response
4. Blueprint
5. Operating
6. URS
7. Security
8. Purpose
9. Non-Functional
10. screen sizes, devices

ग. सही या गलत बताइए

1. (ग़लत)
2. (ग़लत)
3. (ग़लत)
4. (सही)
5. (सही)
6. (ग़लत)
7. (सही)
8. (ग़लत)
9. (ग़लत)
10. (ग़लत)

सत्र 6. प्रोग्रामिंग के लिए निम्न-स्तरीय और उच्च-स्तरीय डिज़ाइन (Low-level and High-level Design for Programming)

क. बहुविकल्पीय प्रश्न

1. (ख) 2. (ख) 3. (क) 4. (क) 5. (ख) 6. (ख) 7. (ग) 8. (ख) 9. (ग) 10. (ख)

ख. रिक्त स्थान भरिए

1. Conceptual
2. Methods
3. Redundancy
4. Abuse

5. Low-level
6. Requirements
7. Fair usage
8. Design
9. Data flow
10. Breaking

ग. सही या गलत बताइए

1. (सही)
2. (सही)
3. (ग़लत)
4. (सही)
5. (ग़लत)
6. (सही)
7. (सही)
8. (सही)
9. (सही)
10. (ग़लत)

सत्र 7. डिज़ाइन दोषों की जाँच और पहचान (Test and Identify Design Defects)

क. बहुविकल्पीय प्रश्न

1. (ग) 2. (क) 3. (ख) 4. (ग) 5. (ख) 6. (घ) 7. (घ) 8. (ग) 9. (ग) 10. (ख)

ख. रिक्त स्थान भरिए

1. Arithmetic Defects
2. Wrong
3. Style
4. Needs
5. Components
6. Unauthorized
7. Regression
8. Scalability
9. Browsers
10. Identification

ग. सही या गलत बताइए

1. (ग़लत)
2. (ग़लत)
3. (ग़लत)
4. (सही)

5. (सही)
6. (ग़लत)
7. (सही)
8. (ग़लत)
9. (सही)
10. (ग़लत)

सत्र 8. डिज़ाइन दोषों का समाधान (Resolve Design Defects)

क. बहुविकल्पीय प्रश्न

1. (ग) 2. (ख) 3. (ख) 4. (क) 5. (ग) 6. (ग) 7. (ग) 8. (ख) 9. (ग) 10. (ग)

ख. रिक्त स्थान भरिए

1. Product
2. Identify
3. First
4. Resolution
5. Functionality
6. Monitoring
7. Development
8. Experiment
9. Addressed
10. Revisions

ग. सही या गलत बताइए

1. (ग़लत)
2. (ग़लत)
3. (ग़लत)
4. (ग़लत)
5. (सही)
6. (ग़लत)
7. (सही)
8. (ग़लत)
9. (ग़लत)
10. (ग़लत)

मॉड्यूल 2. HTML और CSS

सत्र 9. HTML

क. बहुविकल्पीय प्रश्न (MCQs)

1. (क) 2. (क) 3. (ख) 4. (ग) 5. (ग) 6. (ख) 7. (ख) 8. (ग) 9. (ग) 10. (क)

ख. रिक्त स्थान भरिए

1. हाइपरटेक्स्ट मार्कअप
2. HTML5
3. कोणीय
4. <p>
5. एलिमेंट
6. हाइपरलिंक्स
7. <blockquote>
8.
9.
10. HTML

ग. सही या गलत

1. (सही) 2. (गलत) 3. (सही) 4. (गलत) 5. (सही) 6. (गलत) 7. (गलत) 8. (सही) 9. (गलत) 10. (सही)

सत्र 10. सूची की संकल्पना – अव्यवस्थित सूची, क्रमबद्ध सूची, परिभाषा सूची

क. बहुविकल्पीय प्रश्न (MCQs)

1. (क) 2. (ख) 3. (ख) 4. (घ) 5. (ग) 6. (ख) 7. (ग) 8. (ख) 9. (ख) 10. (ख)

ख. रिक्त स्थान भरिए

1.
2. क्रमबद्ध
3.
4. परिभाषा
5. <dl>
6. <img>
7. छवि
8. मैपिंग
9. हाइपरलिंक्स
10. <tbody>

ग. सही या गलत

1. (गलत) 2. (सही) 3. (गलत) 4. (गलत) 5. (गलत) 6. (सही) 7. (सही) 8. (सही) 9. (गलत) 10. (सही)

सत्र 11. CSS

क. बहुविकल्पीय प्रश्न (MCQs)

1. (ख) 2. (ख) 3. (ग) 4. (क) 5. (ख) 6. (घ) 7. (ख) 8. (ख) 9. (ग) 10. (ख)

ख. रिक्त स्थान भरिए

1. प्रस्तुतीकरण
2. तारक (*)
3. तत्व
4. text-align
5. मूल (Parent)
6. letter-spacing
7. अवयवी (Descendant)
8. उत्तरदायी (Responsive)
9. हेड
10. आंतरिक (Internal)

ग. सही या गलत

1. ( ) 2. ( ) 3. ( ) 4. ( ) 5. ( ) 6. ( ) 7. ( ) 8. ( ) 9. ( ) 10. ( )

सत्र 12. CSS बॉक्स मॉडल

क. बहुविकल्पीय प्रश्न (MCQs)

1. (ग) 2. (ग) 3. (क) 4. (क) 5. (ग) 6. (क) 7. (ख) 8. (ख) 9. (ग) 10. (क)

ख. रिक्त स्थान भरिए

1. CSS
2. अंतराल (Space)
3. आयाम (Dimensions)
4. max-width
5. स्थिति (Position)
6. float
7. छिपा हुआ (Hidden)
8. इनलाइन
9. ओवरफ्लो करता है (Overflows)
10. दायाँ (Right)

ग. सही या गलत

1. (गलत) 2. (सही) 3. (सही) 4. (गलत) 5. (गलत) 6. (गलत) 7. (गलत) 8. (गलत) 9. (सही) 10. (गलत)

मॉड्यूल 3. जावास्क्रिप्ट का उपयोग करते हुए वेब विकास

सत्र 13. जावास्क्रिप्ट

क. बहुविकल्पीय प्रश्न (MCQ)

1. (ग) 2. (ख) 3. (ख) 4. (ख) 5. (ख) 6. (क) 7. (घ) 8. (घ) 9. (क) 10. (ख)

ख. रिक्त स्थान भरिए

1. const
2. जावास्क्रिप्ट
3. Node.js
4. संवेदनशील (Sensitive)
5. alert()
6. ब्राउज़र
7. इतिहास
8. कुकीज़
9. स्थान (Location)
10. निष्पादन (Execution)

ग. सही या गलत

1. (सही) 2. (गलत) 3. (सही) 4. (गलत) 5. (सही) 6. (सही) 7. (गलत) 8. (सही) 9. (गलत) 10. (सही)

सत्र 14. सशर्त तर्क और प्रवाह नियंत्रण

क. बहुविकल्पीय प्रश्न (MCQ)

1. (ग) 2. (ख) 3. (क) 4. (क) 5. (ग) 6. (ग) 7. (क) 8. (ख) 9. (ग) 10. (ख)

ख. रिक्त स्थान भरिए

1. true
2. सशर्त (Conditional)
3. तुलना (Compare)
4. बूलियन (Boolean)
5. नेस्टेड (Nested)
6. switch
7. break
8. continue
9. पुनरावृत्ति करना (Iterate)
10. for...in

ग. सही या गलत

1. (सही) 2. (सही) 3. (गलत) 4. (सही) 5. (गलत) 6. (गलत) 7. (सही) 8. (सही) 9. (गलत) 10. (सही)

सत्र 15. ऐरे और फ़ंक्शन

क. बहुविकल्पीय प्रश्न (MCQ)

1. (ख) 2. (ख) 3. (क) 4. (ग) 5. (ख) 6. (ख) 7. (ग) 8. (ग) 9. (ग) 10. (ग)

ख. रिक्त स्थान भरिए

1. pop()
2. पुनः उपयोग योग्य (Reusable)
3. कोष्ठक (Parentheses)
4. return
5. नेटिव (Native)
6. Math.random()
7. String()
8. push()
9. shift()
10. function

ग. सही या गलत

1. (सही) 2. (गलत) 3. (सही) 4. (सही) 5. (गलत) 6. (सही) 7. (गलत) 8. (गलत) 9. (सही) 10. (सही)

सत्र 16. स्ट्रिंग हेरफेर (String Manipulation)

क. बहुविकल्पीय प्रश्न (MCQ)

1. (ख) 2. (क) 3. (ग) 4. (ख) 5. (घ) 6. (ग) 7. (ख) 8. (ख) 9. (ख) 10. (घ)

ख. रिक्त स्थान भरिए

1. संयोजन (Combining)
2. length()
3. toLowerCase()
4. indexOf()
5. substring()
6. सत्यापन (Validation)
7. split()
8. date()
9. isValidDate
10. charAt()

ग. सही या गलत

1. (गलत) 2. (गलत) 3. (गलत) 4. (गलत) 5. (सही) 6. (सही) 7. (सही) 8. (सही) 9. (गलत) 10. (गलत)

सत्र 17. जावास्क्रिप्ट द्वारा चित्रों में हेरफेर

क. बहुविकल्पीय प्रश्न (MCQ)

1. (ख) 2. (क) 3. (ख) 4. (ग) 5. (ख) 6. (ग) 7. (ख) 8. (क) 9. (ग) 10. (ख)

ख. रिक्त स्थान भरिए

1. छवि (Image)
2. drawImage
3. getImageData
4. toDataURL
5. जियोलोकेशन
6. ब्राउज़र
7. स्थान (Location)
8. ट्रैक करना (Track)
9. संशोधित करना (Modify)
10. DOM

ग. सही या गलत

1. (गलत) 2. (गलत) 3. (सही) 4. (गलत) 5. (गलत) 6. (गलत) 7. (सही) 8. (गलत) 9. (गलत) 10. (गलत)

सत्र 18. HTML5 कैनवास (Canvas)

क. बहुविकल्पीय प्रश्न (MCQ)

1. (ख) 2. (घ) 3. (ख) 4. (ग) 5. (क) 6. (घ) 7. (ग) 8. (ख) 9. (क) 10. (ख)

ख. रिक्त स्थान भरिए

1. जावास्क्रिप्ट
2. id
3. getContext('2d')
4. fillRect
5. x, y, चौड़ाई (width), और ऊँचाई (height)
6. "square"
7. beginPath, moveTo, और lineTo
8. beginPath

9. moveTo
10. drawImage

ग. सही या गलत

1. (गलत) 2. (गलत) 3. (गलत) 4. (गलत) 5. (सही) 6. (गलत) 7. (गलत) 8. (गलत) 9. (सही) 10. (सही)