

Foundations of Modern Web Development

Editors

Nisha, Ritika Mishra, Pertik Garg



Published By
Mantra Publication
(An International Publication)

P-ISBN: 978-81-68473-22-5
E-ISBN: 978-81-68473-23-2

Book Title

Foundations of Modern Web Development

Editors Details

1. Nisha

Assistant Professor, Swami Vivekanand Institute of Engineering & Technology,
Banur (Patiala), Punjab, 140601. (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>

2. Ritika Mishra

Assistant Professor, Swami Vivekanand Institute of Engineering & Technology,
Banur (Patiala), Punjab, 140601. (ritikamishra494@gmail.com)
<https://orcid.org/0009-0007-6552-279X>

3. Pertik Garg

Professor, Swami Vivekanand Institute of Engineering & Technology, Banur
(Patiala), Punjab, 140601 (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>



Published, marketed, and distributed by:

Mantra Publication

An International Publication

<https://mantrapublicationservices.com/books-publication/>
editorbooks@mantrapublicationservices.com

Whatsapp: +91 9236371090

P-ISBN: 978-81-68473-22-5

E-ISBN: 978-81-68473-23-2

Copyright ©2026 **Nisha, Ritika Mishra, Pertik Garg**

Citation:

Nisha, Mishra, R., & Garg, P. (Eds.). (2026). *Foundations of modern web development*. Mantra Publication.

This book is published online under a fully open-access model and is distributed in accordance with the Creative Commons Attribution–Non Commercial (CC BY-NC) license. Under the terms of this license, users are permitted to copy, share, and redistribute the content in any medium or format, provided appropriate credit is given to the author(s) and the original source, and the material is not used for commercial purposes. The publisher, editors, and authors disclaim all responsibility for any errors, omissions, or outcomes that may result from the use or interpretation of the information presented in this book. No warranties—whether express or implied—are made regarding the accuracy, completeness, or reliability of the content. While every reasonable effort has been taken to ensure the material is accurate and not misleading, the publisher and contributors do not guarantee that all information, particularly that originating from third parties, has been independently verified. The publisher maintains neutrality with respect to jurisdictional claims reflected in maps, figures, or institutional affiliations included in this publication. Additionally, diligent efforts have been made to identify and obtain permission from all copyright holders for reproduced material. The publisher extends apologies for any inadvertent omissions and requests that any unacknowledged copyright holders contact us so that appropriate corrections can be made in subsequent editions.

Acknowledgment

I would like to express my sincere gratitude to all those who contributed to the successful completion of this book, *Foundations of Modern Web Development*. I am deeply thankful to the editors, contributors, mentors, colleagues, and academic associates for their valuable guidance, encouragement, constructive suggestions, and continuous support throughout the preparation and publication of this scholarly work.

I extend my heartfelt appreciation to my mentors, faculty members, colleagues, collaborators, and academic peers for their insightful discussions, thoughtful suggestions, and critical feedback. Their knowledge, expertise, and encouragement have greatly enriched the content and strengthened the academic and practical perspectives presented in this book.

I would also like to acknowledge the invaluable contributions of researchers, educators, developers, and technology experts worldwide whose pioneering work in Web Technologies, HTML5, CSS3, JavaScript, Responsive Web Design, User Interface and User Experience Design, Web Accessibility, Frontend Development, Web Performance, Version Control, and Modern Web Applications has provided the foundation for the concepts and practices discussed in this volume. Their research, technical standards, documentation, and innovations have served as valuable sources of knowledge and inspiration.

I am sincerely grateful to Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala), Punjab, for providing an encouraging academic environment, institutional support, and continuous motivation that contributed significantly to the completion of this work. I also appreciate the efforts of all contributors, reviewers, technical experts, proofreaders, and members of the publishing team whose dedication and professionalism supported the development and completion of this book.

Special thanks are extended to the global web development and open-source community for continuously advancing web standards, development tools, frameworks, accessibility practices, and emerging technologies that have made the modern Web more powerful, accessible, responsive, and user-oriented.

Finally, I am profoundly grateful to my family and friends for their constant support, patience, understanding, and encouragement throughout the preparation of this book. Their unwavering motivation has been a source of strength and inspiration throughout this academic endeavor.

Editors

- **Nisha**
- **Ritika Mishra**
- **Pertik Garg**

Preface

The World Wide Web has become one of the most influential technological developments of the modern era. From information sharing and communication to education, e-commerce, healthcare, banking, entertainment, and digital governance, web technologies have transformed the way individuals, organizations, and communities interact with information and services. The continuous evolution of web standards, programming languages, development frameworks, and digital platforms has created a growing need for a strong understanding of both fundamental and contemporary web development practices.

The book *Foundations of Modern Web Development* has been developed with the objective of providing a structured and accessible introduction to the essential concepts, technologies, and practices involved in modern web development. It is intended primarily for students, beginners, educators, and readers who wish to establish a strong conceptual foundation before progressing to advanced web technologies and application development.

The book begins with an introduction to the Internet and the World Wide Web, explaining important concepts such as client-server architecture, web browsers, web servers, domain names, DNS, web hosting, and frontend and backend development. These concepts provide the foundation for understanding how web applications are created, deployed, and accessed.

The subsequent chapters introduce the three core technologies of frontend development—HTML, CSS, and JavaScript. HTML5 is discussed as the structural foundation of web pages, covering document structure, elements, hyperlinks, lists, tables, forms, multimedia, and semantic elements. CSS3 focuses on the presentation and visual design of web pages, including selectors, typography, the box model, layouts, Flexbox, Grid, responsive design, transitions, and animations. JavaScript introduces programming concepts that enable webpages to become dynamic and interactive, including variables, operators, conditional statements, loops, functions, events, arrays, objects, DOM manipulation, and form validation.

The final chapter focuses on modern frontend development practices, bringing together concepts such as responsive and mobile-first design, UI/UX principles, web accessibility, CSS frameworks, version control, Git and GitHub, performance optimization, browser compatibility, code quality, and frontend security. It also provides an overview of emerging technologies and trends, including Progressive Web Apps, artificial intelligence, component-based development, voice interfaces, serverless architectures, and WebAssembly.

Editors

- **Nisha**
- **Ritika Mishra**
- **Pertik Garg**

Chapter 1: Introduction to the Web and Internet Technologies

Ms. Nisha*, Er. Ritika Mishra¹, Dr. Pertik Garg²

***Assistant Professor**, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala), Punjab, 140601 (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>.

1. Assistant Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala), Punjab, 140601 (ritikamishra494@gmail.com) <https://orcid.org/0009-0007-6552-279X>.

2. Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala), Punjab, 140601. (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>

Abstract

The World Wide Web has transformed the way people communicate, access information, conduct business, deliver services, and learn in the digital era. The rapid growth of Internet connectivity and web technologies has enabled the development of interactive, dynamic, scalable, and user-friendly applications that can be accessed from virtually anywhere in the world.

This chapter presents a comprehensive introduction to the fundamental concepts of the Internet and the World Wide Web and establishes the foundational knowledge required for understanding modern web development. It explains the evolution of the Web from static information-based websites to interactive and intelligent web applications and highlights the distinction between the Internet as a global communication infrastructure and the Web as a service operating over it. The chapter further discusses client-server architecture and explains how web browsers, web servers, domain names, Domain Name System (DNS), and web hosting services work together to deliver web resources to users. In addition, it introduces the major aspects of frontend and backend development, including the roles of HTML, CSS, JavaScript, server-side technologies, and databases in developing functional web applications.

The chapter also emphasizes the growing importance of web technologies in areas such as education, e-commerce, healthcare, finance, communication, and digital services. By developing a clear understanding of these fundamental concepts and components, learners can build a strong foundation for exploring advanced web development frameworks, application architectures, web services, security mechanisms, and emerging technologies. Overall, the chapter provides a conceptual basis for understanding how modern web applications are designed, developed, deployed, and accessed over the Internet.

Keywords

Internet, World Wide Web, Web Browser, Web Server, Domain Name, Web Development.

Introduction

The Internet has become an integral part of modern life, connecting billions of people and devices across the globe. It serves as a vast network that enables communication, information sharing, online transactions, entertainment, and education. The rapid growth of digital technologies has significantly

increased the demand for web-based applications and services, making web development one of the most important fields in information technology.

The World Wide Web (WWW), commonly known as the Web, is a system of interconnected documents and resources that can be accessed through the Internet. Developed by Sir Tim Berners-Lee in 1989, the Web revolutionized information sharing by providing a standardized method for accessing and displaying content using web browsers. Today, websites and web applications support a wide range of activities, from social networking and e-commerce to cloud computing and online learning.

Web technologies consist of various tools, protocols, and programming languages that work together to create and deliver web content. These technologies enable users to access information through web browsers while allowing organizations to provide services through web servers. Understanding the fundamentals of web technologies is essential for anyone interested in developing websites and web applications [1, 2, 5].

Feature	Internet	World Wide Web
Definition	Global network of computers	Collection of web pages
Purpose	Data communication	Information sharing
Developed By	Various organizations	Tim Berners-Lee
Access Method	Various protocols	HTTP/HTTPS
Examples	Email, FTP, VoIP	Websites, Blogs

Table 1.1 Comparison between Internet and World Wide Web

1.1 Evolution of the Web

The Web has evolved significantly since its inception. The first generation, known as Web 1.0, primarily consisted of static web pages that provided information to users with limited interaction. Users could read content but had minimal opportunities to contribute or engage with websites.

The emergence of Web 2.0 introduced interactive and user-generated content. Social media platforms, blogs, online forums, and collaborative tools allowed users to create, share, and modify content. This era emphasized user participation and dynamic web applications.

Web 1.0 (1990–2004)



Static Websites
(Read Only)



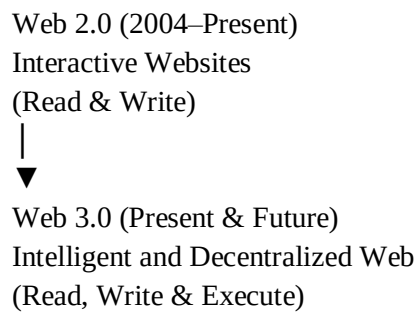


Figure 1.1 Evolution of Web Technologies

The current phase, often referred to as Web 3.0, focuses on intelligent systems, semantic technologies, decentralized applications, and enhanced user experiences. Artificial intelligence, machine learning, blockchain technology, and the Internet of Things (IoT) are shaping the future of web development and digital interaction [1, 2, 3].

1.2 Internet and World Wide Web

The Internet and the World Wide Web (WWW) are related but different concepts. The Internet is the foundation or infrastructure, while the Web is a service that runs on the Internet.

Internet

The Internet is a worldwide network that connects millions of computers, smartphones, servers, and other devices. It allows these devices to exchange data using standard communication protocols such as TCP/IP (Transmission Control Protocol/Internet Protocol).

The Internet provides the basic connectivity required for various services such as:

- Web browsing
- Email communication
- Online banking
- Video streaming
- Cloud services
- File sharing

In simple words, the Internet is like a global communication network that connects devices around the world.

World Wide Web (WWW)

The World Wide Web is a service that works on the Internet. It consists of interconnected web pages, websites, images, videos, and online applications that users access through web browsers such as Chrome, Edge, or Firefox.

The Web uses communication protocols:

- **HTTP (HyperText Transfer Protocol):** Used for transferring web pages between clients and servers.
- **HTTPS (HyperText Transfer Protocol Secure):** A secure version of HTTP that encrypts data communication.

For example, when a user opens a website, the browser sends a request to a web server, and the server returns the required web page [5, 6, 7].

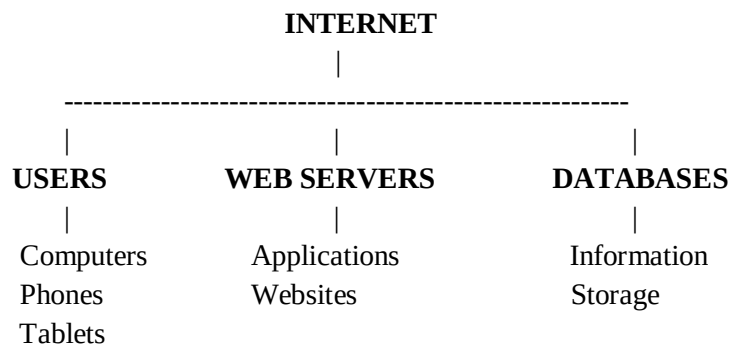


Figure 1.2 Basic Structure of the Internet

1. Users

Users access Internet services through devices such as:

- Computers
- Smartphones
- Tablets

2. Web Servers

- Web servers store and deliver websites and applications to users.
- When a user requests a webpage, the web server processes the request and sends the required content.

3. Databases

- Databases store information required by websites and applications.
- Examples include:
 - ✓ User accounts
 - ✓ Product details
 - ✓ Online records

4. Internet Connection

- The Internet acts as a communication medium connecting users, servers, and databases worldwide.

1.3 Client-Server Architecture

Client-server architecture is a model used in modern web applications where two main components, client and server, communicate with each other to provide services to users.

Client: The client is the user-side application, usually a web browser such as Chrome, Firefox, or Edge. It sends requests to the server and displays the received information to the user.

Server: The server is a powerful computer or system that receives client requests, processes them, manages databases, performs authentication, and sends responses back to the client.

Working of Client-Server Architecture:

1. The user interacts with a web browser (client).
2. The client sends a request to the server through the Internet.
3. The server processes the request using application logic and databases.
4. The server sends the required response back to the client.
5. The client displays the result to the user.

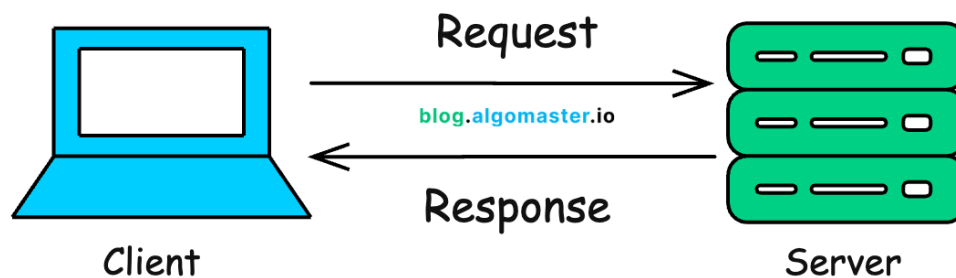


Figure 1.3 Client-Server Communication Model

Components of Client-Server Architecture:

1. Client

- Provides the user interface.
- Collects input from users.
- Sends requests to the server.
- Displays the received results.

Example: A user opens an online shopping website using a browser and searches for a product.

2. Server

- Handles client requests.
- Processes data and business rules.

- Stores and retrieves information from databases.
- Provides security and authentication services.

Example: The shopping website server checks product availability and returns product details.

Advantages of Client-Server Architecture:

- **Scalability:** New users and services can be added easily.
- **Maintainability:** Server-side updates can be managed centrally.
- **Security:** Important data and authentication are controlled by the server.
- **Efficient Resource Utilization:** Processing and storage are handled by dedicated servers.
- **Better Performance:** Multiple clients can access shared resources efficiently [8, 9].

1.4 Web Browsers and Web Servers

Web Browser

A web browser is a software application that allows users to access, view, and interact with websites on the Internet. It acts as a client in the client-server architecture.

When a user enters a website address (URL), the browser sends a request to the web server. The server responds by sending website files such as HTML, CSS, images, and JavaScript. The browser interprets these files and displays the webpage to the user [4].

Examples of popular web browsers:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

Functions of a Web Browser:

- Displays web pages and multimedia content.
- Interprets HTML to create page structure.
- Uses CSS for webpage design and formatting.
- Executes JavaScript to provide interactive features.
- Allows users to navigate between websites.

Web Server

A web server is a computer system or software application that stores website files and provides them to users when requested. It acts as a server in the client-server architecture.

A web server receives requests from browsers, processes them, accesses required files or databases, and sends responses back through the Internet.

Examples of Web Servers:

- Apache HTTP Server
- Nginx
- Microsoft IIS (Internet Information Services)

Functions of a Web Server:

- Stores website files such as HTML, CSS, and images.
- Handles user requests from browsers.
- Processes application logic and database operations.
- Generates responses and sends data to clients.
- Provides security and authentication services [3, 9, 12].

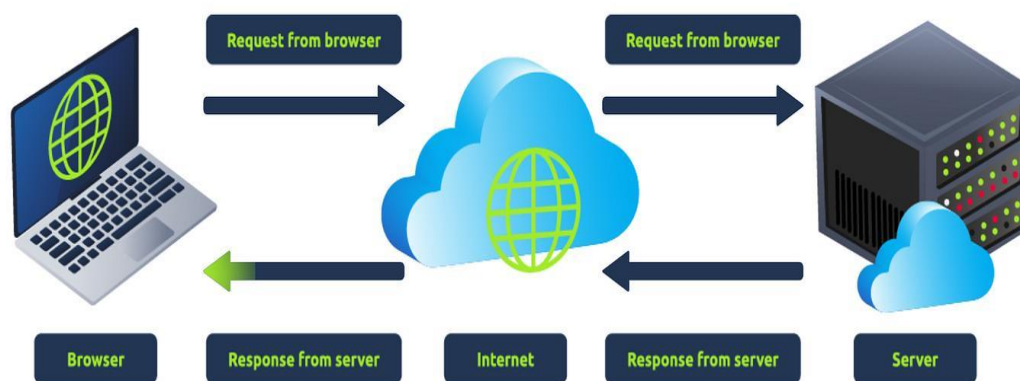


Figure 1.4 Relationship between Web Browser and Web Server

1.5 Domain Names and Web Hosting

A domain name is a human-readable address used to access a website on the Internet. It provides an easy way for users to locate websites without remembering complex numerical IP addresses.

For example, instead of entering an IP address such as 192.168.1.1, users can simply type a domain name like example.com to visit a website.

The Domain Name System (DNS) works like the Internet's address book. It converts domain names into their corresponding IP addresses so that browsers can locate the correct web server [10,11].

Examples of Domain Names:

- google.com
- wikipedia.org
- example.com
- sviet.ac.in

Functions of Domain Names:

- Provide a simple and memorable identity for websites.
- Help users easily access websites.
- Connect website names with their respective IP addresses.
- Improve brand recognition and online presence.

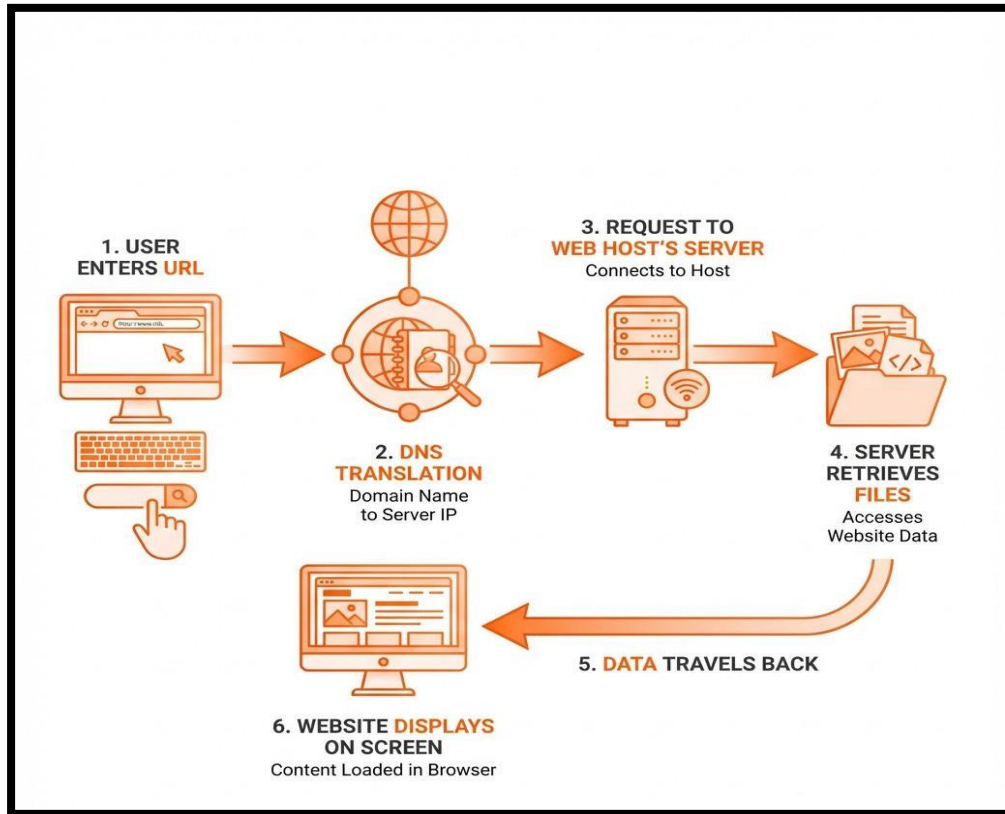


Figure 1.5 Relationship between Domain Names and Web Hosting

Web Hosting

Web hosting is a service that stores website files on powerful servers connected to the Internet. These servers make websites available to users whenever they request access through a web browser.

Hosting providers provide storage space, server resources, security, and network connectivity required for running websites.

Types of Web Hosting:**1. Shared Hosting**

- Multiple websites share resources on a single server.
- It is affordable and suitable for small websites.

2. Virtual Private Server (VPS) Hosting

- A physical server is divided into multiple virtual servers.
- Provides better performance and control than shared hosting.

3. Cloud Hosting

- Websites are hosted on multiple interconnected servers.
- Provides scalability and high availability.

4. Dedicated Hosting

- A complete server is dedicated to a single website or organization.
- Offers maximum performance, security, and customization.

1.6 Frontend and Backend Development

Web development is mainly divided into two major parts:

1. Frontend Development
2. Backend Development

Both parts work together to create complete and interactive web applications.

1. Frontend Development (Client-Side Development)

Frontend development refers to the part of a website or web application that users can see and interact with directly. It focuses on designing the user interface (UI) and improving the user experience (UX).

The frontend is responsible for:

- Displaying web pages in a browser
- Creating layouts, colors, buttons, menus, and forms
- Handling user interactions such as clicks, typing, and navigation
- Making websites responsive for different devices

Common Frontend Technologies:

1. HTML (HyperText Markup Language)

- HTML provides the basic structure of a web page.
- It defines elements such as headings, paragraphs, images, links, and forms.

2. CSS (Cascading Style Sheets)

- CSS is used to design and style web pages.

- It controls colors, fonts, spacing, layouts, and animations.

3. JavaScript

- JavaScript adds interactivity and dynamic behavior to websites.
- It is used for features such as image sliders, form validation, and interactive menus [12, 13].

2. Backend Development (Server-Side Development)

Backend development refers to the server-side part of a web application. It manages the processing of data, business rules, security, and communication between the frontend and databases.

The backend is responsible for:

- Processing user requests
- Storing and retrieving data
- User authentication and security
- Managing application logic
- Connecting with databases

Common Backend Technologies:

1. PHP

- A server-side scripting language used for creating dynamic websites.
- Commonly used with databases like MySQL.

2. Python

- A popular programming language used for web development frameworks such as Django and Flask.
- Provides simple and efficient backend development.

3. Java

- Used for developing large-scale and enterprise-level web applications.
- Provides high security and reliability.

4. Node.js

- Allows JavaScript to run on the server side.
- Used for building fast and real-time applications [14, 15].

Database Technologies:

Databases store and manage application data.

1. MySQL

- A relational database management system.
- Stores data in tables with rows and columns.

2. MongoDB

- A NoSQL database that stores data in flexible document formats.

Frontend Development	Backend Development
User Interface	Server Logic
HTML, CSS, JavaScript	PHP, Python, Java
Runs in Browser	Runs on Server
Visible to Users	Hidden from Users
Focus on User Experience	Focus on Data Processing

Table 1.2 Frontend and Backend Development Comparison

1.7 Importance of Web Technologies

Web technologies play a crucial role in modern society. They support online education, e-commerce, digital marketing, healthcare systems, financial services, and government operations. Organizations rely on websites and web applications to reach customers, provide services, and improve operational efficiency.

Advancements in web technologies continue to drive innovation and digital transformation. As businesses increasingly adopt online platforms, the demand for skilled web developers and designers continues to grow.

Conclusion

The Internet and the World Wide Web have become fundamental components of modern digital communication, information sharing, and online services. This chapter introduced the basic concepts of Internet and Web technologies, including the evolution of the Web, client-server architecture, web browsers, web servers, domain names, DNS, web hosting, and frontend and backend development. Understanding how these components work together provides a strong foundation for designing and developing modern web applications. With the continuous advancement of technologies such as cloud computing, artificial intelligence, the Internet of Things, and distributed systems, web development is becoming increasingly dynamic, intelligent, and user-oriented. Therefore, a sound understanding of fundamental web concepts and technologies is essential for learners to effectively explore advanced web development frameworks, tools, security mechanisms, and emerging digital applications.

References

1. T. Berners-Lee, "Information Management: A Proposal," CERN, Geneva, Switzerland, 1989.
2. T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The World Wide Web," *Communications of the ACM*, vol. 37, no. 8, pp. 76–82, 1994.
3. World Wide Web Consortium (W3C), "Architecture of the World Wide Web, Volume One," W3C Recommendation, 2004.
4. T. Berners-Lee, "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web," RFC 1630, Internet Engineering Task Force (IETF), 1994.
5. J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. Boston, MA, USA: Pearson, 2021.
6. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 6th ed. Boston, MA, USA: Pearson Education, 2011.
7. D. E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, 6th ed. Boston, MA, USA: Pearson Education, 2013.
8. R. Fielding, "Architectural Styles and the Design of Network-Based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
9. R. T. Fielding et al., "Hypertext Transfer Protocol—HTTP/1.1," RFC 2616, Internet Engineering Task Force (IETF), 1999.
10. P. Mockapetris, "Domain Names—Concepts and Facilities," RFC 1034, Internet Engineering Task Force (IETF), 1987.
11. P. Mockapetris, "Domain Names—Implementation and Specification," RFC 1035, Internet Engineering Task Force (IETF), 1987.
12. J. Duckett, *HTML and CSS: Design and Build Websites*. Indianapolis, IN, USA: Wiley, 2011.
13. D. Flanagan, *JavaScript: The Definitive Guide*, 7th ed. Sebastopol, CA, USA: O'Reilly Media, 2020.
14. R. Nixon, *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*, 6th ed. Sebastopol, CA, USA: O'Reilly Media, 2021.
15. K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.

Chapter 2

HTML5 Fundamentals and Web Page Structure

Ms. Nisha*, Er. Ritika Mishra¹, Dr. Pertik Garg²

***Assistant Professor**, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>.

1. Assistant Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (ritikamishra494@gmail.com) <https://orcid.org/0009-0007-6552-279X>.

2. Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601. (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>

Abstract

HTML5 is the modern standard markup language used for structuring and presenting content on the World Wide Web. It provides a well-defined framework for creating web pages and web applications that can be accessed across different browsers, devices, and platforms. HTML5 introduced several important features that simplify web development, including semantic elements, native multimedia support, enhanced forms, improved accessibility, and support for modern web applications.

This chapter provides a comprehensive introduction to the fundamental concepts of HTML5 and explains how different elements contribute to the structure and functionality of a web page. It discusses the basic HTML5 document structure, commonly used tags and elements, text formatting, hyperlinks, lists, tables, forms, multimedia elements, and semantic HTML5 elements. Particular emphasis is placed on the use of semantic tags such as `<header>`, `<nav>`, `<section>`, `<article>`, `<aside>`, and `<footer>`, which help organize webpage content and improve readability, accessibility, and search engine understanding.

The chapter also explains the role of forms and modern input controls in collecting user information and highlights the importance of native audio and video elements in delivering multimedia content without relying on external plug-ins. Through these concepts, learners can understand how HTML5 provides the structural foundation for modern web development. The knowledge presented in this chapter establishes a strong basis for further learning of CSS, JavaScript, responsive web design, accessibility, and interactive web application development.

Keywords

HTML5, Web Pages, Hyperlinks, Forms, Multimedia, Semantic Elements

Introduction

The World Wide Web consists of billions of web pages that provide information and services to users across the globe. At the core of every web page lies HTML (HyperText Markup Language), the

standard markup language used to create and structure content on the web. HTML defines the organization of text, images, multimedia, forms, and other elements that appear in a web browser.

HTML5 is the fifth and most advanced version of HTML. It was developed to address the growing needs of modern web applications by introducing improved support for multimedia, graphics, accessibility, and semantic content. HTML5 eliminates the need for many third-party plugins and provides developers with powerful features to create rich and interactive websites.

Understanding HTML5 is the first step toward becoming a web developer. It forms the foundation upon which other technologies such as CSS and JavaScript are built. This chapter introduces the key concepts and components of HTML5 and explains how web pages are structured and displayed in browsers.

2.1 Overview of HTML5

HTML5 is a markup language used to define the structure and content of web pages. Unlike programming languages, HTML uses tags and elements to describe different parts of a document.

Some key features of HTML5 include:

- Simplified document structure
- Native support for audio and video
- Improved form controls
- Semantic elements for better content organization
- Enhanced accessibility
- Better compatibility across devices and browsers
- Support for modern web applications

HTML5 has become the standard technology for web development due to its flexibility, efficiency, and compatibility with emerging web technologies [1, 2].

2.2 Structure of an HTML5 Document

Every HTML document follows a standard structure that helps browsers interpret and display content correctly.

The basic structure of an HTML5 document consists of:

- Document Type Declaration
- HTML Element
- Head Section
- Body Section

The document type declaration informs the browser that the document uses HTML5 standards.

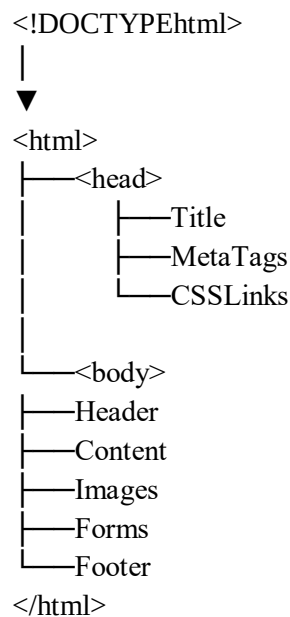


Figure 2.1 Basic Structure of an HTML5 Document

The head section contains metadata such as the page title, character encoding, and links to external resources.

The body section contains the visible content of the webpage, including text, images, forms, and multimedia elements.

A properly structured HTML document improves readability, maintainability, and search engine optimization [1, 2].

2.3 HTML Elements and Tags

HTML documents are composed of elements represented by tags. Most elements consist of an opening tag, content, and a closing tag.

Examples of common HTML elements include:

- Heading elements
- Paragraph elements
- Hyperlink elements
- Image elements
- List elements
- Table elements

Tags help browsers understand the purpose of content and render it appropriately. Proper use of tags contributes to better website structure and accessibility [1, 2].

Tag	Purpose
<html>	Root element
<head>	Metadata section
<title>	Page title
<body>	Visible content
<h1> to <h6>	Headings
<p>	Paragraph
<a>	Hyperlink
	Image
<table>	Table
<form>	Form

Table 2.1 frequently used HTML5 Tags

2.4 Text Formatting in HTML5

HTML5 provides several elements for formatting text and improving content presentation.

Common text formatting elements include:

- Headings
- Paragraphs
- Line breaks
- Bold text
- Italic text
- Underlined text
- Superscript and subscript text

Headings play an important role in organizing webpage content. They create a hierarchical structure that enhances readability and supports search engine indexing.

Paragraph elements allow developers to present textual content in a clear and organized manner [1, 2].

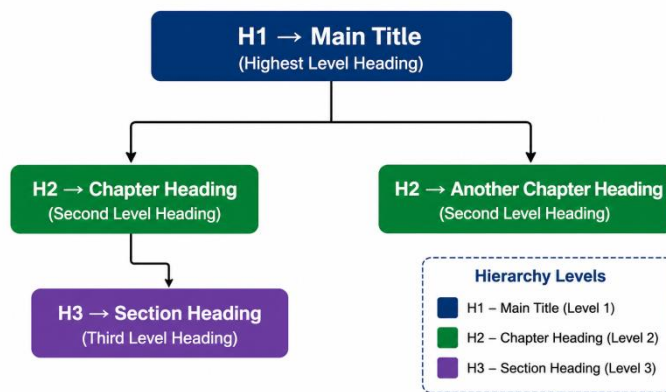


Figure 2.2 HTML Heading Hierarchy

2.5 Hyperlinks and Navigation

Hyperlinks are one of the most important features of the Web. They enable users to navigate between web pages and access resources across the Internet.

A hyperlink can connect:

- One webpage to another
- A webpage to a downloadable file
- A webpage to an email address
- Different sections within the same webpage

Effective navigation improves user experience and helps visitors access information quickly and efficiently.

Modern websites often use navigation menus, breadcrumbs, and internal links to enhance usability and content accessibility [2, 4].

2.6 Lists in HTML5

Lists are used to organize related information in a structured format. HTML5 supports different types of lists.

Ordered Lists

Ordered lists display items in a numbered sequence. They are useful for instructions, procedures, and rankings.

List Type	Tag Used	Example
Ordered List		1. Item One 2. Item Two 3. Item Three
Unordered List		• Item One • Item Two • Item Three
Description List	<dl>	Term 1 – Definition 1 Term 2 – Definition 2

Table 2.2 Types of HTML Lists

Unordered Lists

Unordered lists display items using bullet points. They are commonly used for menus, feature lists, and categories.

Description Lists

Description lists present terms and their corresponding descriptions, making them suitable for glossaries and definitions.

Lists improve readability and allow users to quickly scan webpage content [1, 2].

2.7 Tables in HTML5

Tables are used to organize data into rows and columns. They provide an effective method for presenting structured information such as schedules, reports, and statistical data.

A table typically contains:

- Rows
- Columns
- Headers
- Data cells

Student Marks Table	
Student	Marks
Ravi	85
Neha	92
Aman	78

Table 2.3 Structure of an HTML Table

Well-designed tables enhance data presentation and improve user understanding. HTML5 also supports table captions and grouping elements that improve accessibility and organization [1, 2, 5].

2.8 Forms and User Input

Forms are interactive elements that allow users to submit information to a website.

Common form controls include:

- Text fields
- Password fields
- Radio buttons
- Checkboxes
- Drop-down menus
- Text areas
- Submit buttons

HTML5 introduces advanced form features such as:

- Email validation
- Date pickers
- Number fields
- Range sliders
- Search boxes



The image shows a sample HTML form titled "Student Registration Form". The form is enclosed in a light green border. It contains the following elements: a "Name" label followed by a text input field; an "Email" label followed by a text input field; a "Course" label followed by a dropdown menu with the text "Select Course" and a downward arrow; a "Gender" label followed by two radio buttons labeled "Male" and "Female"; and a green "Submit" button at the bottom.

Figure 2.3 Samle HTML Form

These enhancements improve user experience and reduce the need for complex validation scripts.

Forms play a vital role in applications such as registration systems, login portals, surveys, and online transactions [1, 2, 6].

2.9 Multimedia Elements in HTML5

One of the most significant improvements in HTML5 is native multimedia support.

HTML5 provides built-in elements for:**Audio Content**

Developers can embed music, podcasts, and sound effects directly into webpages without relying on external plugins.

Video Content

HTML5 supports video playback using standard browser functionality, enabling seamless delivery of educational content, advertisements, and entertainment media.

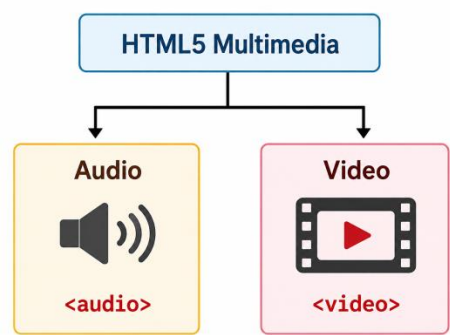


Figure 2.4 HTML5 Multimedia Support

Graphics and Visual Content

HTML5 introduces advanced graphical capabilities that support animations, interactive content, and dynamic visualizations.

The integration of multimedia elements enhances user engagement and improves content delivery [1, 2].

2.10 Semantic HTML5 Elements

Semantic elements describe the meaning and purpose of content within a webpage.

Examples of semantic elements include:

- Header
- Navigation
- Section
- Article
- Aside
- Footer

Tag	Purpose
<audio>	Play audio files
<video>	Play video files
<source>	Specify media source
<track>	Add subtitles/captions

Table 2.4 Multimedia Elements in HTML

These elements improve:

- Website organization
- Accessibility
- Search engine optimization
- Code readability

Semantic HTML enables browsers and search engines to better understand webpage content, resulting in improved indexing and user experiences [1, 2, 3].

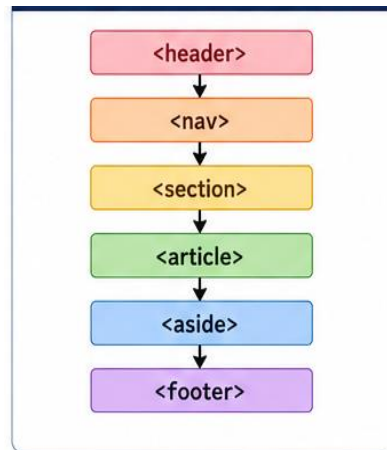


Figure 2.5 Sematic Structure of an HTML5 Webpage

2.11 Importance of HTML5 in Modern Web Development

HTML5 serves as the backbone of modern web applications. It provides a standardized framework for content creation and ensures compatibility across different browsers and devices.

The advantages of HTML5 include:

- Cross-platform compatibility
- Mobile responsiveness
- Improved performance
- Enhanced accessibility
- Better multimedia support
- Simplified code structure

Organizations worldwide rely on HTML5 to build websites, e-commerce platforms, learning management systems, social networking applications, and enterprise solutions [1-3, 7].

Conclusion

HTML5 provides the fundamental structure for developing modern, accessible, and interactive web pages. This chapter covered the essential concepts of HTML5, including document structure, elements and tags, text formatting, hyperlinks, lists, tables, forms, multimedia, and semantic elements. Features such as native audio and video support, enhanced form controls, and semantic tags make web development more efficient, organized, and user-friendly. A clear understanding of HTML5 forms the foundation for learning CSS, JavaScript, responsive design, accessibility, and modern web application

development. Proper use of HTML5 elements also improves the readability, maintainability, and accessibility of web pages. Semantic elements help browsers, search engines, and assistive technologies understand the purpose and organization of webpage content. The use of forms, tables, hyperlinks, and multimedia further enables developers to create functional and engaging websites. Therefore, mastering HTML5 is an essential step toward developing well-structured, responsive, and user-oriented web applications.

References

1. WHATWG. HTML: Living Standard. Web Hypertext Application Technology Working Group. Updated July 2026.
2. Mozilla Developer Network (MDN). Structuring content with HTML. MDN Web Docs.
3. World Wide Web Consortium (W3C). Web Content Accessibility Guidelines (WCAG) 2.2. W3C Recommendation.
4. Mozilla Developer Network (MDN). Creating links. MDN Web Docs.
5. Mozilla Developer Network (MDN). HTML table basics and table accessibility. MDN Web Docs.
6. Mozilla Developer Network (MDN). Forms and buttons in HTML. MDN Web Docs.
7. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis: Wiley; 2011. ISBN: 978-1-118-00818-8.

Chapter 3

Styling Web Pages with CSS3

Ms. Nisha*, Er. Ritika Mishra¹, Dr. Pertik Garg²

***Assistant Professor**, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>.

1. Assistant Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (ritikamishra494@gmail.com) <https://orcid.org/0009-0007-6552-279X>.

2. Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601. (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>

Abstract

CSS3 (Cascading Style Sheets) is a core technology of modern web development that defines the presentation, layout, and visual appearance of web pages while separating design from HTML content. It enables developers to create attractive, maintainable, and responsive websites through advanced styling features such as selectors, typography, colors, backgrounds, borders, transitions, animations, Flexbox, CSS Grid, and media queries. Modern CSS emphasizes modular specifications and responsive design techniques that allow web pages to adapt seamlessly across desktops, tablets, and smartphones, improving usability and accessibility.

This chapter introduces the fundamental concepts of CSS3, including methods of applying styles, CSS selectors, the box model, layout techniques, responsive web design, and visual effects. Practical examples are provided to strengthen conceptual understanding and demonstrate real-world implementation. The chapter also discusses best practices for writing organized, reusable, and efficient stylesheets that enhance website performance and maintainability. Overall, it equips learners with the knowledge and practical skills required to develop visually appealing, user-friendly, and professional web applications using modern CSS techniques.

Keywords

CSS3, Stylesheets, Selectors, Flexbox, Grid Layout, Responsive Design.

Introduction

Modern websites are expected to be visually appealing, responsive, and user-friendly. While HTML provides the content and structure of a webpage, CSS3 controls its appearance and presentation. CSS stands for Cascading Style Sheets and is used to define how HTML elements should be displayed on screen, paper, or other media.

Before CSS was introduced, web designers had limited control over the appearance of web pages. CSS separated content from presentation, making websites easier to design, maintain, and update.

CSS3 is the latest major version of CSS and provides advanced features such as animations, transitions, gradients, shadows, flexible layouts, and responsive design capabilities.

CSS3 enables developers to create professional-looking websites without relying heavily on images or external design tools. It plays a crucial role in modern web development and is one of the core technologies of the World Wide Web.

3.1 What is CSS3?

CSS3 (Cascading Style Sheets Level 3) is a stylesheet language used to define **the** visual appearance, presentation, and layout of HTML documents. While HTML is mainly responsible for creating and organizing the content of a webpage, CSS3 determines how that content looks and how it is positioned on the screen.

CSS3 allows web developers to separate content from presentation, making websites easier to design, maintain, and update. Instead of adding formatting instructions to every HTML element, developers can create reusable CSS rules and apply them to multiple elements [1, 2].

What Can CSS3 Control?

CSS3 provides a wide range of features for designing modern websites, including:

- **Text Styles:** Font family, font size, font weight, alignment, line height, and text decoration.
- **Colors:** Text colors, background colors, gradients, and transparency.
- **Layouts:** Positioning and arrangement of webpage elements using techniques such as Flexbox and CSS Grid.
- **Spacing:** Margins, padding, and gaps between elements.
- **Borders:** Border width, style, color, radius, and rounded corners.
- **Backgrounds:** Background colors, images, gradients, sizes, and positions.
- **Animations and Transitions:** Smooth visual effects and movement of webpage elements.
- **Responsive Design:** Adaptation of webpages to different screen sizes, including desktops, tablets, and smartphones.
- **Shadows and Effects:** Box shadows, text shadows, opacity, filters, and other visual effects.

How CSS3 Works

CSS works by selecting HTML elements and applying a set of style rules to them. A CSS rule generally consists of a selector, followed by one or more declarations.

Example

HTML Code:

```
<h1>Welcome to CSS3</h1>
<p>This is a paragraph.</p>
```

CSS Code:

```
h1
{
  color: blue;
}

p
{
  color: green;
}
```

Output:

The heading appears in blue, and the paragraph appears in green.

3.2 Advantages of CSS3

CSS3 provides several important advantages in modern web development:

1. **Separation of Content and Design:** HTML handles the structure while CSS handles presentation.
2. **Consistency:** A single stylesheet can be applied to multiple webpages, ensuring a consistent design.
3. **Reduced Code:** Reusable CSS rules reduce repetitive formatting code in HTML.
4. **Better Maintainability:** Changes to the design can be made by modifying the CSS instead of individual HTML elements.
5. **Responsive Websites:** CSS3 makes it possible to create layouts that adapt to different devices and screen sizes.
6. **Rich Visual Effects:** Developers can create animations, transitions, gradients, shadows, and other effects without relying heavily on external tools.
7. **Improved User Experience:** Well-designed layouts, readable typography, and responsive interfaces make websites easier to use [1, 2].

3.3 Ways to Apply CSS

CSS can be applied to HTML documents in three main ways: Inline CSS, Internal CSS, and External CSS. All three methods allow developers to control the appearance and presentation of HTML elements. The choice of method depends on the size of the website, the level of customization required, and the need for code reusability [3].

1. Inline CSS

Inline CSS is used to apply styling directly to an individual HTML element. The CSS declarations are written inside the style attribute of the HTML element.

Example:

```
<p style="color:red;">Hello World</p>
```

In this example, the style attribute applies the color: red property only to the particular <p>element.

Advantages of Inline CSS:

- Easy to apply for a single element.
- Useful for quick testing and small modifications.
- Does not require a separate CSS section or file.
- Useful when a particular element needs a unique style.

Disadvantages of Inline CSS:

- Difficult to maintain in large websites.
- Produces repetitive code.
- Cannot be easily reused across multiple elements.
- Makes HTML documents less organized.
- Mixing content and presentation can make the code difficult to read.

Therefore, inline CSS is generally suitable for small, specific styling requirements rather than complete website design.

2. Internal CSS

Internal CSS, also called Embedded CSS, is written inside a <style> element within the <head> section of an HTML document. It allows developers to define styles that can be applied to multiple elements on the same webpage.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Internal CSS Example</title>

<style>
  p {
    color: blue;
    font-size: 18px;
  }

  h1 {
    text-align: center;
    color: darkblue;
  }
</style>
```

```
</head>

<body>

<h1>Welcome to CSS</h1>
<p>This paragraph uses internal CSS.</p>

</body>
</html>
```

In this example, the `<style>` element contains CSS rules for both the `<h1>` and `<p>` elements. These styles are applicable to elements on that particular HTML page.

Advantages of Internal CSS:

- Suitable for designing a single webpage.
- CSS rules can be applied to multiple elements on the same page.
- Keeps the styling separate from individual HTML elements.
- Easier to manage than multiple inline styles.
- Useful for webpage-specific designs.

Disadvantages of Internal CSS:

- The styles cannot be directly reused across multiple HTML pages.
- Large internal stylesheets can increase the size of HTML documents.
- Maintaining the same design across many pages becomes difficult.
- Repeated CSS rules may be required for different webpages.

Internal CSS is therefore useful for single-page websites, demonstrations, prototypes, and page-specific styling.

3. External CSS

External CSS is the most commonly preferred method for designing multi-page websites. In this approach, CSS rules are written in a separate file with the `.css` extension. The CSS file is then connected to the HTML document using the `<link>` element.

HTML:

```
<!DOCTYPE html>
<html>
<head>
<title>External CSS Example</title>
<link rel="stylesheet" href="style.css">
</head>
```

```
<body>

<h1>Welcome to My Website</h1>
<p>This webpage uses external CSS.</p>

</body>
</html>
```

CSS File (style.css):

```
h1 {
  color: darkblue;
  text-align: center;
}

p {
  color: green;
  font-size: 18px;
}
```

The statement:

```
<link rel="stylesheet" href="style.css">
```

Advantages of External CSS:

- **Reusability:** One CSS file can be used by many HTML pages.
- **Maintainability:** Changes can be made in one stylesheet and reflected across multiple pages.
- **Consistency:** Helps maintain a uniform appearance throughout a website.
- **Cleaner HTML:** Presentation rules are separated from the webpage structure.
- **Better Organization:** HTML and CSS code are stored separately.
- **Efficient Development:** Developers can manage the design of large websites more easily.
- **Caching Benefits:** Browsers may cache external stylesheets, which can improve loading efficiency when users visit multiple pages using the same stylesheet.

Disadvantages of External CSS:

- Requires an additional CSS file.
- The webpage depends on the stylesheet being correctly linked.
- An incorrect file path can prevent the styles from being applied.
- Very small webpages may not require a separate stylesheet.

3.4 CSS Selectors

CSS selectors are one of the most important concepts in CSS. A selector identifies the HTML element or elements to which a particular set of CSS rules should be applied. Selectors allow developers to control the appearance, layout, spacing, colors, fonts, and other visual properties of web pages.

The general syntax of a CSS rule is:

```
selector {  
    property: value;  
}
```

For example:

```
p {  
    color: blue;  
    font-size: 16px;  
}
```

Here, p is the selector, color and font-size are properties, and blue and 16px are their corresponding values.

CSS provides several types of selectors. The commonly used selectors are described below.

1. Element Selector

The element selector selects all HTML elements of a particular type. It uses the HTML element name as the selector.

Example:

```
h1 {  
    color: blue;  
    font-size: 32px;  
}
```

HTML:

```
<h1>Welcome to My Website</h1>  
<h1>About Us</h1>
```

This rule applies the specified styles to all <h1> elements on the webpage.

Advantages:

- Simple and easy to use.

- Useful when the same style is required for all elements of a particular type.
- Reduces repetition in CSS code.

2. Class Selector

The class selector is used to apply the same style to one or more HTML elements. A class selector is represented by a dot (.) followed by the class name.

Example:

```
.title {  
  
    color: red;  
  
    font-size: 28px;  
  
    text-align: center;  
  
}
```

HTML:

```
<h1 class="title">Welcome</h1>  
<p class="title">Introduction to CSS</p>
```

Both elements use the title class, so the CSS styles are applied to both.

A major advantage of class selectors is that the same class can be assigned to multiple HTML elements.

3. ID Selector

The ID selector is used to style a specific HTML element. It is represented by a hash symbol (#) followed by the ID name.

Example:

```
#header {  
  
    background-color: lightgray;  
  
    padding: 20px;  
  
    text-align: center;  
  
}
```

HTML:

```
<div id="header">Website Header</div>
```

The style is applied to the element whose id is header.

An ID should generally be unique within an HTML document, meaning that the same ID should not normally be assigned to multiple elements.

4. Universal Selector

The universal selector selects all HTML elements on a webpage. It is represented by an asterisk (*).

Example:

```
* {  
  margin: 0;  
  padding: 0;  
}
```

This rule removes the default margin and padding from all HTML elements.

The universal selector is commonly used when developers want to establish a consistent starting point for page styling.

5. Grouping Selector

A grouping selector allows multiple selectors to share the same CSS declarations. The selectors are separated by commas.

Example:

```
h1, h2, h3 {  
  color: navy;  
  font-family: Arial, sans-serif;  
}
```

This rule applies the same styles to <h1>, <h2>, and <h3> elements.

6. Descendant Selector

A descendant selector selects elements that are contained inside another element. A space is used between the selectors.

Example:

```
nava {  
  color: white;  
  text-decoration: none;  
}
```

HTML:

```
<nav>
```

```
<a href="#">Home</a>
<a href="#">About</a>
</nav>
```

Here, `nav a` selects all `<a>` elements that occur inside the `<nav>` element.

This selector is useful for styling elements according to their location within the HTML structure.

7. Child Selector

The child selector selects elements that are direct children of another element. It is represented by the greater-than symbol (`>`).

Example:

```
nav>a {
  color: green;
}
```

This selects only `<a>` elements that are direct children of `<nav>`.

8. Attribute Selector

An attribute selector selects HTML elements based on their attributes.

Example:

```
input[type="text"] {
  border: 1pxsolidgray;
  padding: 8px;
}
```

HTML:

```
<inputtype="text">
<inputtype="password">
```

The CSS rule applies only to the input element whose type attributes has the value "text".

Attribute selectors are especially useful for styling different types of form controls.

9. Pseudo-Class Selector

A pseudo-class selector defines a special state of an HTML element. It begins with a colon (`:`).

A common example is the: hover pseudo-class.

```
a:hover {  
  color: red;  
}
```

This rule changes the color of a link when the user moves the mouse pointer over it.

10. Pseudo-Element Selector

A pseudo-element is used to style a specific part of an element. It is generally represented using two colons (::).

Example:

```
p::first-letter {  
  font-size: 30px;  
  font-weight: bold;  
}
```

This rule styles only the first letter of each paragraph [5, 6].

3.5 Colors and Backgrounds

Colors and backgrounds play an important role in web design. They improve the visual appearance of a webpage, create visual hierarchy, highlight important information, and contribute to a consistent website theme. CSS3 provides several methods for specifying colors and many properties for controlling backgrounds.

3.5.1 Color Names

CSS supports a large number of predefined color names. These names provide a simple way to specify colors.

Example:

```
p  
{  
  color: red;  
}
```

In this example, the text of all <p> elements will appear in red.

3.5.2 RGB Values

RGB stands for Red, Green, and Blue. In the RGB color model, a color is created by combining different intensities of these three primary colors.

The syntax is:

```
rgb(red, green, blue)
```

Each value normally ranges from 0 to 255.

Example:

```
p
{
  color: rgb(255,0,0);
}
```

The value `rgb(255, 0, 0)` represents red because the red component is at its maximum while green and blue are set to zero.

3.5.3 Hexadecimal Values

Hexadecimal colors are represented using a hash symbol (#) followed by six hexadecimal characters.

Example:

```
p
{
  color: #ff0000;
}
```

The value `#ff0000` represents red.

3.5.4 Background Color

The `background-color` property specifies the background color of an element.

Example:

```
body
{
  background-color: lightblue;
}
```

This sets the background color of the entire webpage to light blue.

3.5.5 Background Image

The `background-image` property is used to display an image behind the content of an element.

Example:

```
body
{
  background-image: url("background.jpg");
}
```

Background properties help create visually attractive websites [7, 8].

3.6 Typography and Text Styling

Typography refers to the way text is designed, arranged, and presented on a webpage. It is an important part of web design because well-formatted text improves readability, visual hierarchy, accessibility, and overall aesthetics. CSS provides several properties that allow developers to control the appearance of text, including its font, size, weight, alignment, color, spacing, and decoration.

3.6.1 Font Family

The font-family property specifies the typeface used to display text.

Example:

```
p
{
  font-family: Arial, sans-serif;
}
```

In this example, the browser first tries to use the Arial font. If Arial is not available, it uses a suitable sans-serif font.

3.6.2 Font Size

The font-size property controls the size of text.

Example:

```
p
{
  font-size: 18px;
}
```

This sets the paragraph text to a size of 18 pixels.

3.6.3 Font Weight

The font-weight property determines how thick or bold the text appears.

Example:

```
p
{
  font-weight: bold;
}
```

The paragraph text will appear bold.

3.6.4 Text Alignment

The text-align property controls the horizontal alignment of text within its containing element.

Example:

```
h1
```

```
{  
  text-align: center;  
}
```

Effective typography enhances user experience and content presentation [9, 10].

3.7 The CSS Box Model

The CSS Box Model is one of the fundamental concepts in web design and layout. It describes how the browser calculates the size of an HTML element and the space surrounding it.

In CSS, every HTML element is treated as a rectangular box. Whether the element is a heading, paragraph, image, button, `<div>`, or form control, it can be understood in terms of the box model.

The box model consists of:

1. Content

The content is the actual information contained inside an HTML element. It may include text, images, videos, or other HTML elements.

For example:

```
<div>  
  Welcome to Web Development  
</div>
```

The text "Welcome to Web Development" is the content of the `<div>`.

The width and height properties can be used to control the dimensions of the content area.

Example:

```
div {  
  width: 300px;  
  height: 100px;  
}
```

Here, the content area has a width of 300 pixels and a height of 100 pixels, subject to the box-sizing rules in use.

2. Padding

Padding is the space between the content and the border of an element. It creates internal space and prevents the content from touching the border.

Example:

```
div {  
  padding: 20px;  
}
```

This adds 20 pixels of padding on all four sides of the element.

Padding can also be specified separately:

```
div {  
  padding-top: 10px;  
  padding-right: 20px;  
  padding-bottom: 10px;  
  padding-left: 20px;  
}
```

3. Border

The border surrounds the content and padding. It provides a visible boundary around an element.

Example:

```
div {  
  border: 2pxsolidblack;  
}
```

This creates a 2-pixel-wide solid black border.

A border consists of three main properties:

```
div {  
  border-width: 2px;  
  border-style: solid;  
  border-color: black;  
}
```

4. Margin

Margin is the space outside the border of an element. It creates separation between the element and surrounding elements.

Example:

```
div {  
  margin: 15px;
```

```
}
```

This creates a 15-pixel margin on all four sides.

Individual margins can also be specified:

```
div {  
  margin-top: 10px;  
  margin-right: 20px;  
  margin-bottom: 15px;  
  margin-left: 20px;  
}
```

Margins are particularly important when arranging multiple elements on a webpage [7].

3.8 CSS Layout Techniques

CSS layout techniques determine how elements are positioned, aligned, sized, and arranged on a webpage. A well-designed layout helps organize content into clear sections and ensures that websites are easy to navigate and use.

CSS provides several layout mechanisms, including the display, position, and float properties. Modern CSS also provides powerful layout systems such as Flexbox and CSS Grid, which are widely used for responsive web design.

3.8.1 Display Property

The display property specifies how an HTML element participates in the layout of a webpage. It is one of the most commonly used CSS properties for controlling element arrangement.

Example:

```
Div  
{  
  display: block;  
}
```

A block-level element generally starts on a new line and occupies the available horizontal space within its containing block.

Common display values include:

- block
- inline
- inline-block
- none
- flex

- grid

3.8.2 Position Property

The position property determines how an element is positioned within a webpage.

Common values include:

- static
- relative
- absolute
- fixed
- sticky

The default position of most elements is static.

Example:

```
.box
{
  position: relative;
  left: 20px;
}
```

Here, the element is shifted 20 pixels to the right from its normal position.

3.8.3 Float Property

The float property was traditionally used to position elements to the left or right and allow surrounding content to flow around them.

Example:

```
img
{
  float: right;
}
```

This places the image toward the right side of its containing area, allowing text to flow around it [11].

3.9 Flexbox Layout

Flexbox (Flexible Box Layout) is a one-dimensional CSS layout model used to arrange elements efficiently in a row or column. It provides a simple and flexible way to control the alignment, spacing, and distribution of elements within a container.

Example:**HTML:**

```
<divclass="container">
<div>Item 1</div>
<div>Item 2</div>
<div>Item 3</div>
</div>
```

CSS:

```
.container {
  display: flex;
  justify-content: space-around;
}
```

In this example, `display: flex` converts the container into a flex container. The `justify-content: space-around` property distributes equal space around the flex items [12].

Important Flexbox Properties:

- **display: flex** – Defines an element as a flex container.
- **flex-direction** – Specifies whether items are arranged in a row or column.
- **justify-content** – Controls the alignment of items along the main axis.
- **align-items** – Controls the alignment of items along the cross axis.
- **flex-wrap** – Allows items to move to the next line when necessary.
- **gap** – Defines the space between flex items.

Advantages of Flexbox:

- Easy alignment of elements
- Supports responsive layouts
- Provides flexible spacing
- Simplifies positioning of components
- Reduces CSS code complexity

3.10 CSS Grid Layout

CSS Grid is a two-dimensional layout system used to arrange webpage elements in both rows and columns. It provides precise control over the placement, alignment, and spacing of elements, making it suitable for designing complex and responsive web pages.

Example:**CSS:**

```
.container {  
  display: grid;  
  grid-template-columns: 1fr1fr1fr;  
  gap: 10px;  
}
```

Explanation:

- **display: grid;** – Converts the container into a CSS Grid layout.
- **grid-template-columns: 1fr 1fr 1fr;** – Creates three equal-width columns.
- **gap: 10px;** – Adds a 10-pixel gap between rows and columns.
- **1fr** – Represents one fraction of the available space.

Applications of CSS Grid: CSS Grid is commonly used for:

- **Dashboards** – Organizing widgets, charts, and information panels.
- **Magazine layouts** – Arranging text, images, and articles in structured grids.
- **E-commerce websites** – Displaying products in rows and columns.
- **Admin panels** – Structuring navigation menus, cards, tables, and reports [13].

3.11 Responsive Web Design

Responsive Web Design (RWD) is a web development approach that enables a website to automatically adjust its layout, content, images, and other elements according to the size and orientation of the user's device. It provides a consistent and user-friendly experience across smartphones, tablets, laptops, and desktop computers.

Importance of Responsive Web Design

A responsive website:

- Adapts automatically to different screen sizes.
- Improves readability and navigation on smaller screens.
- Reduces the need for separate mobile and desktop versions.
- Provides a better user experience across different devices.
- Helps websites work effectively with different screen resolutions and orientations.

Media Queries

Media queries are an important CSS feature used to apply different styles based on the characteristics of a device or screen, such as its width, height, or orientation.

Example:

```
@media screen and (max-width:768px)
{
  body
  {
    background-color: lightgray;
  }
}
```

Responsive design improves usability on:

- Smartphones
- Tablets
- Laptops
- Desktop computers

Modern websites must support multiple devices and screen resolutions [14].

3.12 CSS3 Effects and Animations

CSS3 provides a wide range of visual effects and animation features that allow developers to create attractive, interactive, and modern web pages without relying heavily on external scripts or image-editing software. These features can be used to enhance buttons, containers, images, menus, and other webpage elements.

3.12.1 Border Radius

The border-radius property is used to create rounded corners on HTML elements. It is commonly used for buttons, cards, images, and input fields.

Example:

```
button {
  border-radius: 10px;
}
```

In this example, the corners of the button are rounded by 10 pixels.

3.12.2 Box Shadow

The box-shadow property adds a shadow around an element. It can create a sense of depth and make elements such as cards, boxes, and buttons visually distinct.

Example:

```
div {  
  box-shadow: 5px5px10pxgray;  
}
```

Here, the shadow is shifted 5 pixels horizontally and vertically, with a blur radius of 10 pixels.

3.12.3 Transition

The transition property creates a smooth change between different CSS property values. It is frequently used with the :hover pseudo-class to create interactive effects.

Example:

```
button {  
  transition: 0.5s;  
}  
  
button:hover {  
  transform: scale(1.1);  
}
```

When the user moves the mouse over the button, it smoothly increases in size.

3.12.4 Animation

CSS animations allow elements to change their appearance or position over a specified period. Animations are defined using the @keyframes rule.

Example:

```
.box {  
  position: relative;  
  animation: move2sinfinitealternate;  
}  
  
@keyframesmove {  
  from {  
    left: 0;  
  }  
  to {  
    left: 200px;  
  }  
}
```

In this example, the element moves from its initial position to 200 pixels to the right. The animation repeats continuously and alternates between the starting and ending positions.

Common CSS3 Effects:

Some commonly used CSS3 effects include:

- **Border Radius** – Creates rounded corners.
- **Box Shadow** – Adds depth and visual emphasis.
- **Transitions** – Produces smooth changes between styles.
- **Transforms** – Rotates, scales, moves, or skews elements.
- **Animations** – Creates dynamic movement and visual effects.
- **Opacity** – Controls the transparency of elements.

CSS3 effects and animations improve the visual appearance, usability, and interactivity of websites. When used appropriately, they can make webpages more engaging while maintaining a clean and professional design [15].

3.13 Best Practices for CSS Development

To create efficient style sheets, developers should:

- Use external CSS files
- Follow consistent naming conventions
- Minimize code duplication
- Organize styles logically
- Optimize CSS for performance
- Test designs across browsers

Following best practices results in maintainable and scalable web projects [1, 2].

Conclusion

CSS3 is an essential technology for modern web development that enhances the visual appearance, layout, and usability of HTML-based web pages. It provides developers with powerful features for styling text, colors, backgrounds, spacing, borders, and positioning, while advanced layout techniques such as Flexbox and CSS Grid enable the creation of flexible and well-structured interfaces. Responsive design and media queries allow websites to adapt effectively to different screen sizes and devices. CSS3 also supports transitions, shadows, rounded corners, and animations that improve visual appeal and user interaction. By following proper CSS development practices, developers can create websites that are attractive, responsive, maintainable, and efficient. Thus, a strong understanding of CSS3 is fundamental for developing modern, user-friendly, and professional web applications.

References

1. Meyer, E. A., & Weyl, E. (2023). *CSS: The Definitive Guide: Visual Presentation for the Web* (5th ed.). O'Reilly Media.
2. World Wide Web Consortium (W3C). (2023). *CSS Snapshot 2023*. W3C.

3. World Wide Web Consortium (W3C). (2011). *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*. W3C.
4. World Wide Web Consortium (W3C). (2023). *CSS Cascading and Inheritance Level 5*. W3C.
5. World Wide Web Consortium (W3C). (2018). *Selectors Level 3*. W3C.
6. World Wide Web Consortium (W3C). (2022). *Selectors Level 4*. W3C.
7. Bos, B., Etemad, E., & Kemper, B. (2023). *CSS Backgrounds and Borders Module Level 3*. W3C.
8. World Wide Web Consortium (W3C). (2022). *CSS Color Module Level 3*. W3C.
9. World Wide Web Consortium (W3C). (2023). *CSS Text Module Level 3*. W3C.
10. World Wide Web Consortium (W3C). (2022). *CSS Fonts Module Level 4*. W3C.
11. World Wide Web Consortium (W3C). (2022). *CSS Display Module Level 3*. W3C.
12. World Wide Web Consortium (W3C). (2017). *CSS Flexible Box Layout Module Level 1*. W3C.
13. World Wide Web Consortium (W3C). (2020). *CSS Grid Layout Module Level 1*. W3C.
14. Rivoal, F. (2022). *Media Queries Level 3*. World Wide Web Consortium (W3C).
15. Baron, D., Jackson, D., Birtles, B., & Hyatt, D. (2018). *CSS Transitions*. World Wide Web Consortium (W3C).

Chapter 4

Interactive Web Pages with JavaScript

Ms. Nisha*, Er. Ritika Mishra¹, Dr. Pertik Garg²

***Assistant Professor**, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>.

1. Assistant Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (ritikamishra494@gmail.com) <https://orcid.org/0009-0007-6552-279X>.

2. Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601. (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>

Abstract

JavaScript is one of the most widely used programming languages for developing interactive, dynamic, and responsive web applications. It plays a vital role in modern web development by adding functionality and behavior to webpages, complementing HTML, which provides structure, and CSS, which manages presentation and visual appearance. JavaScript enables web pages to respond to user interactions such as button clicks, mouse movements, keyboard input, form submissions, and other events. It can also dynamically modify webpage content, styles, and elements through the Document Object Model (DOM), thereby improving interactivity and user experience. This chapter provides a comprehensive introduction to the fundamental concepts of JavaScript, including variables, data types, operators, conditional statements, loops, functions, events, arrays, objects, DOM manipulation, and form validation. Practical examples are presented to demonstrate how JavaScript concepts can be implemented within HTML documents to develop functional web interfaces. The chapter also explains different approaches for incorporating JavaScript into webpages, including inline, internal, and external scripting, with emphasis on the advantages of organized and maintainable code. Form validation and dynamic style modification are discussed as important applications of JavaScript in improving data accuracy, usability, and responsiveness. Furthermore, the chapter highlights essential JavaScript development practices such as meaningful naming conventions, reusable functions, input validation, code documentation, and systematic testing. By understanding these fundamental concepts, learners can develop a strong foundation for creating interactive websites and can progress toward advanced topics such as asynchronous programming, API integration, single-page applications, and full-stack web development. Overall, JavaScript serves as a crucial technology for building modern, user-centered, and feature-rich web applications.

Keywords

JavaScript, DOM, Events, Functions, Variables, Form Validation.

Introduction

Modern websites are expected to be interactive and responsive. Users want immediate feedback when clicking buttons, submitting forms, navigating menus, or interacting with content. HTML and CSS

alone cannot provide such dynamic behavior. JavaScript fills this gap by enabling web pages to respond to user actions and update content without reloading the page.

JavaScript was developed by Brendan Eich in 1995 and has since become one of the core technologies of the World Wide Web. Today, JavaScript is supported by all major web browsers and is used extensively for frontend as well as backend development.

JavaScript allows developers to manipulate webpage content, perform calculations, validate user input, create animations, and communicate with web servers. As a result, it has become an essential technology for modern web development [1, 3, 15].

4.1 What is JavaScript?

JavaScript is a high-level, interpreted, lightweight, and object-oriented programming language that enables developers to create dynamic and interactive web pages. It is one of the three core technologies of web development, alongside HTML and CSS. While HTML defines the structure of a webpage and CSS controls its appearance, JavaScript adds functionality and interactivity by allowing web pages to respond to user actions and update content without reloading the page. JavaScript is primarily executed in the user's web browser (client-side), although it can also be used for server-side development through environments such as Node.js.

JavaScript allows developers to build modern web applications by manipulating HTML elements, validating user input, handling events, performing calculations, communicating with web servers, and creating animations. It supports features such as event-driven programming, asynchronous processing, and object-oriented programming, making it suitable for developing responsive and feature-rich websites. Today, JavaScript is widely used in web development frameworks and libraries such as React, Angular, and Vue.js to create interactive user interfaces.

JavaScript can:

- Respond to user actions
- Manipulate webpage content
- Validate form data
- Create animations
- Communicate with servers
- Develop web applications

JavaScript executes directly within a web browser, making it highly efficient for client-side programming [2, 3, 8].

Example

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("Welcome to JavaScript!");
```

```
</script>
```

```
</body>
```

```
</html>
```

Output:

Welcome to JavaScript!

4.2 Adding JavaScript to Web Pages

JavaScript code can be added to an HTML document in three different ways: Inline JavaScript, Internal JavaScript, and External JavaScript. Each method has its own advantages and is suitable for different situations depending on the size and complexity of the website. The `<script>` element is used to include JavaScript code in HTML pages [1, 3].

1. Inline JavaScript

Inline JavaScript places JavaScript code directly inside an HTML element using event attributes such as `onclick`, `onmouseover`, or `onchange`. It is mainly used for simple tasks involving a single element.

Example:

```
<buttononclick="alert('Hello User!')">
```

Click Me

```
</button>
```

Advantages:

- Easy to implement for small programs.
- Suitable for testing simple functionality.
- No separate JavaScript file is required.

Disadvantages:

- Mixes HTML and JavaScript code.
- Difficult to maintain for large websites.
- Code cannot be reused easily.
- Generally discouraged for professional web development because event listeners provide a cleaner approach [4, 5].

2. Internal JavaScript

Internal JavaScript is written inside the `<script>` element within the same HTML document. The `<script>` element is commonly placed before the closing `</body>` tag so that HTML elements are loaded before the script executes.

Example:

```
<script>  
alert("Welcome to JavaScript");  
</script>
```

Advantages:

- Suitable for single-page applications.
- Keeps all code within one HTML file.
- Easier to manage than inline JavaScript.

Disadvantages:

- Cannot be reused across multiple webpages.
- Makes large HTML files difficult to maintain.
- Repeated code may be required in multiple pages [1, 5].

3. External JavaScript

External JavaScript stores JavaScript code in a separate file with the .js extension. The HTML page links the file using the src attribute of the <script> element.

HTML File:

```
<scriptsrc="script.js"></script>
```

JavaScript File (script.js)

```
alert("External JavaScript File");
```

Advantages:

- Improves code organization and readability.
- The same JavaScript file can be reused by multiple webpages.
- Simplifies maintenance because changes are made in one file.
- Browsers can cache external JavaScript files, improving website performance.
- Supports collaborative development for large projects.

Disadvantages:

- Requires an additional JavaScript file.
- The webpage depends on the correct file path.
- If the file is missing or incorrectly linked, the script will not execute.
- Small webpages may not require a separate JavaScript file [1, 8].

4.3 Variables and Data Types

Variables are named memory locations used to store information (data) that can be accessed, modified, and reused during the execution of a JavaScript program. They make programs more flexible by allowing values to change as the program runs. Instead of writing the same value repeatedly, developers store it in a variable and use the variable name whenever needed. Modern JavaScript primarily uses the `let` keyword to declare variables whose values may change and `const` for values that should remain constant. The older `var` keyword is still supported but is generally discouraged in modern JavaScript because of its function-scoping behavior. JavaScript is a dynamically typed language, meaning a variable can hold different types of values during program execution without explicitly declaring its data type.

Variable Declaration

A variable is declared using the `let` keyword, followed by a variable name and an optional initial value.

```
let name = "Nisha";  
let age = 25;
```

In the above example:

- name stores the text "Nisha".
- age stores the numeric value 25.

Variables can be updated whenever required.

```
age = 26;
```

Here, the value of age changes from 25 to 26.

Common Data Types

JavaScript supports several built-in data types. The most commonly used are:

1. String

A String stores text enclosed within single (' ') or double (" ") quotation marks.

```
let student = "Ritika";
```

2. Number

A Number stores both integers and decimal values.

```
let marks = 85;
```

3. Boolean

A Boolean stores one of two values: true or false. It is mainly used in decision-making and conditional statements.

```
letpassed=true;
```

4. Null

The Null data type represents the intentional absence of a value.

```
letresult=null;
```

5. Undefined

A variable that has been declared but not assigned a value is undefined.

```
letaddress;
```

The value of address is undefined until a value is assigned.

Example:

```
letstudent="Ritika";  
letmarks=85;  
letpassed=true;
```

Output:

```
Student: Ritika  
Marks: 85  
Passed: true
```

Advantages of Variables:

- Store data for later use.
- Allow values to be modified during program execution.
- Reduce repetition and make code easier to read.
- Improve program flexibility and maintainability.
- Simplify calculations and data processing [1, 2, 8].

4.4 Operators in JavaScript

Operators are special symbols or keywords used to perform operations on values and variables. They are mainly used for calculations, comparisons, and decision-making in JavaScript programs.

1. Arithmetic Operators

Arithmetic operators are used to perform mathematical calculations such as addition, subtraction, multiplication, and division.

Example:

```
let a = 10;
let b = 5;
console.log (a + b);
console.log (a - b);
console.log (a * b);
console.log (a / b);
console.log(a/b);
```

Output:

```
15
5
50
2
```

2. Comparison Operators

Comparison operators are used to compare two values. They return a Boolean value (true or false).

Example:

```
let x = 20;
console.log(x > 10);
console.log(x == 20);
```

Output:

```
true
true
```

3. Logical Operators

Logical operators are used to combine multiple conditions and produce a Boolean result.

Example:

```
let age = 18;  
console.log (age >= 18 && age < 60);
```

Output:

true

Operators form the foundation of decision-making and computations [2, 8].

4.5 Conditional Statements

Conditional statements are used to make decisions in a program. They allow JavaScript to execute different blocks of code based on whether a given condition is true or false.

1. If Statement

The if statement executes a block of code only when the specified condition is true.

Syntax:

```
if (condition)  
{  
  // code to execute  
}
```

Example:

```
letmarks=80;  
  
if (marks>=50)  
{  
  document.write("Pass");  
}
```

Output:

Pass

2. If-Else Statement

The if-else statement is used when a program needs to perform one action if a condition is true and another action if the condition is false.

Syntax:

```
if (condition)  
{  
  // code when condition is true  
}  
else
```

```
{  
// code when condition is false  
}
```

Example:

```
letmarks=40;
```

```
if (marks>=50)  
{  
document.write("Pass");  
}  
else  
{  
document.write("Fail");  
}
```

Output:

Fail

3. Switch Statement

The switch statement is used to execute one block of code from multiple choices. It compares a value with different cases and executes the matching case [1, 2, 8].

Syntax:

```
switch(expression)  
{  
casevalue1:  
// code  
break;  
  
casevalue2:  
// code  
break;  
  
default:  
// code  
}
```

Example:

```
letday=3;
```

```
switch(day)  
{  
case1:
```

```
document.write("Monday");  
break;  
  
case2:  
document.write("Tuesday");  
break;  
  
default:  
document.write("Other Day");  
}
```

Output:

Other Day

4.6 Loops in JavaScript

A **loop** is a programming structure that allows a block of code to execute repeatedly until a specific condition becomes false. Loops are used to reduce repeated code, save time, and improve program efficiency. JavaScript provides different types of loops such as for loop, while loop, and do-while loop.

1. For Loop

A for loop is used when the number of repetitions is known in advance. It contains three parts:

1. **Initialization** – Sets the starting value of the counter.
2. **Condition** – Checks whether the loop should continue.
3. **Increment/Decrement** – Updates the counter value after each iteration.

Syntax:

```
for (initialization; condition; increment)  
{  
  // code to execute  
}
```

Example:

```
for (let i=1; i<=5; i++)  
{  
  document.write(i+"<br>");  
}
```

Output:

1
2
3
4

5

2. While Loop

A while loop executes a block of code as long as the given condition is true. The condition is checked before executing the loop body.

Syntax:

```
while (condition)
{
// code to execute
}
```

Example:

```
leti=1;
```

```
while (i<=5)
{
document.write(i+"<br>");
i++;
}
```

Output:

```
1
2
3
4
5
```

3. Do-While Loop

A do-while loop is similar to a while loop, but the condition is checked after executing the code. Therefore, the loop body executes at least once even if the condition is false initially.

Syntax:

```
do
{
// code to execute
}
while (condition);
```

Example:

```
leti=1;
```

```
do
{
document.write(i+"<br>");
```

```
i++;  
}  
while (i<=5);
```

Output:

```
1  
2  
3  
4  
5
```

Advantages of Loops:

- Reduce code repetition.
- Make programs shorter and easier to understand.
- Improve execution efficiency.
- Help process large amounts of data easily.
- Allow automatic repetition of tasks [2, 8].

4.7 Functions in JavaScript

A function is a reusable block of code designed to perform a specific task. Instead of writing the same code multiple times, we can create a function once and call it whenever required. Functions make programs more organized, easier to understand, and reduce code duplication.

Function Syntax:

```
function functionName()  
{  
  // code to be executed  
}
```

Components of a Function:

- **function** → Keyword used to create a function.
- **functionName** → Name given to the function.
- **()** → Parentheses used for parameters.
- **{ }** → Contains the statements that execute when the function is called.

1. Simple Function Example

```
function greet()  
{  
  alert("Welcome to JavaScript");  
}  
  
greet();
```

Explanation:

- greet() is a function name.
- The alert() statement displays a message.
- greet(); calls or invokes the function.
- When the function is called, the message "Welcome to JavaScript" appears.

A function does not execute when it is defined; it runs only when it is called.

2. Function with Parameters

Parameters allow a function to accept values from outside. These values are used inside the function to perform operations.

Example:

```
functionadd(a,b)
{
returna+b;
}
```

```
document.write(add(10,20));
```

Output:

30

Explanation:

- a and b are parameters of the function.
- add(10,20) passes two values (arguments) to the function.
- The return statement sends the result back.
- The result 30 is displayed using document.write().

Types of Functions in JavaScript:

1. Built-in Functions

These functions are already provided by JavaScript.

Examples:

```
alert();
Math.random();
parseInt();
```

2. User-defined Functions

Functions created by programmers according to their requirements.

Example:

```
functiondisplay()  
{  
document.write("Hello");  
}
```

Advantages of Functions:

- **Code Reusability:** Write code once and use it multiple times.
- **Modularity:** Divides a large program into smaller manageable parts.
- **Easy Debugging:** Errors can be found and fixed easily.
- **Improves Readability:** Makes programs simpler and organized.
- **Saves Time:** Reduces repeated coding [1, 2, 8].

4.8 Events in JavaScript

Events are actions or occurrences that happen in a web page and are detected by the browser. JavaScript uses events to respond to user interactions such as clicking a button, moving the mouse, pressing a key, submitting a form, or loading a webpage. An event runs a specific piece of code called an event handler when the event occurs.

Events make web pages interactive and dynamic by allowing programs to react to user actions [3, 4, 8].

Common events include:**1. Click Event**

The click event occurs when a user clicks on an HTML element such as a button or link.

Example:

```
<buttononclick="showMessage()">  
  Click Here  
</button>  
  
<script>  
functionshowMessage()  
{  
alert("Button Clicked!");  
}  
</script>
```

Explanation:

- The button contains an onclick event.
- When the user clicks the button, the showMessage() function is called.
- The alert () function displays the message "Button Clicked!".

2. Mouseover Event

The mouseover event occurs when the mouse pointer moves over an element.

Example:

```
<ponmouseover="changeText()">  
  Move mouse here  
</p>
```

It is commonly used for changing colors, displaying information, or creating animations.

3. Keypress Event

The keypress event occurs when a user presses a key on the keyboard.

Example:

```
<inputonkeypress="displayMessage()">
```

It is used in forms, search boxes, and keyboard-based controls.

4. Submit Event

The submit event occurs when a user submits an HTML form.

Example:

```
<formonsubmit="validateForm()">
```

It is used for checking user input before sending form data.

5. Load Event

The load event occurs when a webpage and its required resources have finished loading.

Example:

```
<bodyonload="welcome()">
```

```
<script>  
function welcome()
```

```
{  
alert("Page Loaded Successfully");  
}  
</script>
```

4.9 Document Object Model (DOM)

The Document Object Model (DOM) is a programming interface that represents an HTML document as a tree structure of objects (nodes). It allows JavaScript to access, modify, and control the content, structure, and style of a webpage dynamically.

When a browser loads an HTML page, it creates a DOM tree where each HTML element becomes an object that JavaScript can interact with.

JavaScript Uses of DOM:

Using the DOM, JavaScript can perform various operations:

1. Change Content

JavaScript can modify the text or HTML content of an element.

Example:

```
document.getElementById("demo").innerHTML = "Text Changed Successfully";
```

The `getElementById()` method finds an HTML element using its unique ID and allows JavaScript to manipulate it.

2. Modify Styles

JavaScript can change the appearance of webpage elements.

Example:

```
document.getElementById("demo").style.color = "blue";
```

This changes the text color of the selected element.

3. Add New Elements

New HTML elements can be created and added dynamically.

Example:

```
let p=document.createElement("p");  
p.innerHTML ="New Paragraph Added";  
document.body.appendChild(p);
```

4. Remove Elements

JavaScript can remove existing elements from a webpage. [3, 10, 11].

Example:

```
document.getElementById("demo").remove();
```

4.10 Changing CSS with JavaScript

JavaScript can be used to modify CSS styles dynamically after a webpage has loaded. By using the DOM, JavaScript can access HTML elements and change their appearance, such as color, size, background, font, and visibility. This helps create interactive and attractive web pages.

The CSS properties of an HTML element can be changed using the styleproperty with JavaScript

Example:

```
<p id="text">  
Hello World  
</p>
```

```
<button onclick="changeColor()">  
Change Color  
</button>
```

JavaScript:

```
function changeColor()  
{  
    document.getElementById("text").style.color = "red";  
}
```

This capability enhances user interaction and visual effects [3, 4, 10].

4.11 Form Validation

Form validation is the process of checking whether the information entered by a user in a web form is correct and complete before submitting it. It helps prevent incorrect data from being sent and improves the overall user experience. JavaScript can be used for client-side validation to check user input immediately in the browser.

Purpose of Form Validation:

- Ensures users enter correct information.
- Prevents empty or invalid form submissions.
- Reduces errors in collected data.
- Provides instant feedback to users.
- Improves usability of web applications.

Example:

HTML:

```
<form onsubmit="return validateForm()">
```

Name:

```
<input type="text" id="name">
```

```
<input type="submit">
```

```
</form>
```

JavaScript:

```
function validateForm()
{
    let name =
    document.getElementById("name").value;

    if(name=="")
    {
        alert("Name cannot be empty");
        return false;
    }

    return true;
}
```

Validation improves data accuracy and user experience [1, 3, 8].

4.12 JavaScript Arrays

A JavaScript Array is a special type of variable that is used to store multiple values in a single variable. Instead of creating separate variables for each value, an array allows us to store all related data together.

Example:

```
let subjects =  
["HTML","CSS","JavaScript"];
```

```
document.write(subjects[0]);
```

Explanation:

- let subjects creates an array named subjects.
- The values "HTML", "CSS", and "JavaScript" are stored inside the array.
- Array elements are stored with an index number.
- Indexing starts from 0 in JavaScript.

Advantages of Arrays:

- Store multiple values using a single variable.
- Make data organization easier.
- Allow quick access to data using index numbers.
- Simplify data processing using loops and built-in methods [1, 2, 8].

4.13 JavaScript Objects

JavaScript objects are used to store related information in a structured way. An object stores data in the form of key-value pairs, where:

- **Key** → Name or property of the data
- **Value** → Information stored for that property

JavaScript objects are collections of properties, and each property connects a name (key) with a value.

Example:

```
let student = {  
  name: "Nisha",  
  age:22,  
  course:"BCA"  
};
```

```
document.write(student.name);
```

Objects are widely used in web applications for managing structured data [1, 2, 8].

Advantages of JavaScript Objects:

1. Organized Data Storage
2. Easy Data Access
3. Supports Complex Data
4. Used in Web Applications

4.14 Advantages of JavaScript

JavaScript offers several benefits:

- Faster client-side execution
- Interactive user interfaces
- Cross-browser compatibility
- Easy integration with HTML and CSS
- Rich web application development
- Large developer community

These advantages have made JavaScript the most popular programming language for web development [1, 3, 8].

4.15 Best Practices for JavaScript Development

Developers should follow these practices:

- Use meaningful variable names
- Write reusable functions
- Avoid unnecessary global variables
- Validate user input
- Comment complex code
- Test applications thoroughly

Good coding practices improve software quality and maintainability [2, 8].

Conclusion

JavaScript plays a fundamental role in transforming static HTML and CSS-based webpages into dynamic, interactive, and responsive web applications. This chapter introduced the essential concepts of JavaScript, including variables, data types, operators, conditional statements, loops, functions, events, arrays, objects, and form validation. The use of the Document Object Model (DOM) enables developers to dynamically modify webpage content and styles in response to user actions. JavaScript also improves user experience by providing immediate feedback, validating input, and supporting interactive features without requiring complete page reloads. By following appropriate coding practices, developers can create efficient, maintainable, and scalable web applications. Overall, a

strong understanding of JavaScript fundamentals provides an essential foundation for advanced web development and modern client-side application development.

References

1. Flanagan, D., *JavaScript: The Definitive Guide*, 7th Edition, O'Reilly Media, 2020.
2. Haverbeke, M., *Eloquent JavaScript: A Modern Introduction to Programming*, 3rd Edition, No Starch Press, 2018.
3. Mozilla Developer Network (MDN), *JavaScript Guide*, Mozilla Foundation
4. Duckett, J., *JavaScript and JQuery: Interactive Front-End Web Development*, Wiley, 2014.
5. Robbins, J. N., *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics*, 5th Edition, O'Reilly Media, 2018.
6. Powell, T. A., *HTML & CSS: The Complete Reference*, 5th Edition, McGraw-Hill Education, 2010.
7. Souders, S., *High Performance Web Sites: Essential Knowledge for Front-End Engineers*, O'Reilly Media, 2007.
8. Zakas, N. C., *Professional JavaScript for Web Developers*, 4th Edition, Wiley, 2019.
9. Freeman, E., Robson, E., *Head First JavaScript Programming: A Brain-Friendly Guide*, O'Reilly Media, 2014.
10. Keith, J., *DOM Scripting: Web Design with JavaScript and the Document Object Model*, 2nd Edition, Friends of ED, 2007.
11. W3C, *Document Object Model (DOM) Standard*, World Wide Web Consortium.
12. Weyl, E., *HTML5: The Missing Manual*, 2nd Edition, O'Reilly Media, 2014.
13. Cederholm, D., *CSS3 for Web Designers*, A Book Apart, 2010.
14. McPeak, J., *JavaScript: The Complete Reference*, McGraw-Hill Education, 2015.
15. Eich, B., *JavaScript: The First 20 Years*, Association for Computing Machinery (ACM), 2020.

Chapter 5

Modern Frontend Development Practices

Ms. Nisha*, Er. Ritika Mishra¹, Dr. Pertik Garg²

***Assistant Professor**, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>.

1. Assistant Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (ritikamishra494@gmail.com) <https://orcid.org/0009-0007-6552-279X>.

2. Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601. (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>

Abstract

Frontend development has evolved from the creation of simple static web pages into the development of highly interactive, responsive, accessible, and performance-oriented web applications. The increasing use of smartphones, tablets, laptops, and different web browsers has created a strong need for websites that can adapt seamlessly to diverse devices, screen sizes, and user requirements. Modern frontend development practices integrate HTML, CSS, JavaScript, responsive design principles, UI/UX methodologies, accessibility standards, CSS frameworks, version control systems, and performance optimization techniques to deliver effective digital experiences.

This chapter presents the fundamental concepts and contemporary practices used in modern frontend development. It discusses the evolution of frontend technologies and explains responsive and mobile-first design approaches for creating flexible interfaces. The principles of user interface and user experience design are explored with emphasis on consistency, simplicity, visual hierarchy, navigation, feedback, and user-centered design. The chapter also highlights the importance of web accessibility, including the use of alternative text, meaningful headings, keyboard navigation, and appropriate color contrast.

The role of CSS frameworks such as Bootstrap and Tailwind CSS in accelerating development and maintaining design consistency is also introduced. Furthermore, version control using Git and collaborative development through GitHub are discussed as essential practices for managing source code and supporting teamwork. Website performance optimization techniques, including minification, image optimization, caching, and lazy loading, are examined to improve loading speed and overall user experience. Browser compatibility, code quality, maintainability, and frontend security considerations are also addressed. Finally, the chapter introduces emerging trends such as Progressive Web Apps (PWAs), artificial intelligence integration, serverless architectures, component-based frameworks, voice user interfaces, and WebAssembly. Overall, the chapter provides a practical foundation for understanding modern frontend development practices and emphasizes the importance of creating web applications that are responsive, accessible, secure, efficient, maintainable, and capable of meeting the changing expectations of users and the evolving requirements of the web industry.

Keywords

Frontend Development, Responsive Design, UI/UX, Accessibility, CSS Frameworks, Version Control.

Introduction

Frontend development refers to the process of designing and developing the visual and interactive aspects of a website or web application. It encompasses everything users see and interact with directly in their web browsers. Modern frontend development aims to create websites that are visually attractive, highly functional, responsive, and accessible.

With the rapid growth of mobile devices and diverse screen sizes, developers must ensure that websites provide a consistent experience across desktops, tablets, and smartphones. Additionally, users expect websites to load quickly, be easy to navigate, and comply with accessibility standards.

Modern frontend development combines HTML, CSS, JavaScript, frameworks, design principles, and optimization techniques to create high-quality digital experiences [1-3].

5.1 Evolution of Frontend Development

Frontend development has undergone significant transformation over the years.

Traditional Websites

Earlier websites consisted of static HTML pages with minimal styling and interactivity.

Dynamic Websites

The introduction of CSS and JavaScript enabled developers to create dynamic and visually appealing websites.

Modern Frontend Applications

Today's frontend applications utilize advanced frameworks, responsive design techniques, and performance optimization strategies to provide rich user experiences [1, 2].

Characteristics of modern frontend development include:

- Mobile responsiveness
- Interactive interfaces
- Accessibility compliance
- Component-based architecture
- Faster loading times
- Enhanced user engagement

5.2 Responsive Web Design

Responsive Web Design (RWD) ensures that a website adapts to different screen sizes and devices.

Importance of Responsive Design

- Improved user experience
- Better mobile accessibility
- Higher search engine rankings
- Reduced maintenance effort

Media Query Example

```
@media screen and (max-width:768px)
{
  body
  {
    background-color: lightgray;
  }
}
```

The above media query changes the background color when the screen width is less than or equal to 768 pixels.

Responsive Image Example

```
img
{
  max-width:100%;
  height:auto;
}
```

This ensures that images automatically resize according to the screen dimensions.

Responsive design has become essential because a large percentage of web traffic originates from mobile devices [4, 5].

5.3 Mobile-First Design Approach

The mobile-first approach involves designing websites for smaller screens before adapting them for larger devices.

Benefits

- Improved performance
- Better user experience
- Cleaner code structure

- Enhanced accessibility

Example

```
.container
{
  width:100%;
}

@media(min-width:768px)
{
  .container
  {
    width:80%;
  }
}
```

This approach ensures that websites perform efficiently on mobile devices before additional features are added for larger screens [5, 6].

5.4 User Interface (UI) Design Principles

User Interface (UI) design focuses on the appearance and presentation of digital products.

Key UI Principles***Consistency***

Elements should behave similarly throughout the website.

Simplicity

Interfaces should be easy to understand and navigate.

Visual Hierarchy

Important information should stand out clearly.

Feedback

Users should receive immediate responses to their actions.

Example

A submit button changing color when hovered over provides visual feedback.

```
button:hover
{
```

```
background-color:green;
}
```

Good UI design improves user satisfaction and usability [7].

5.5 User Experience (UX) Design Principles

User Experience (UX) focuses on how users interact with a website.

Characteristics of Good UX

- Easy navigation
- Fast loading speed
- Clear content organization
- Accessibility
- User-centered design

Example

A website with a simple navigation menu allows users to find information quickly.

```
<nav>
<ul>
<li><a href="#">Home</a></li>
<li><a href="#">About</a></li>
<li><a href="#">Contact</a></li>
</ul>
</nav>
```

UX design aims to reduce user frustration and improve engagement [7, 8].

5.6 Web Accessibility

Accessibility ensures that websites can be used by people with disabilities.

Importance of Accessibility

- Inclusive user experience
- Legal compliance
- Improved usability
- Better SEO performance

Accessibility Practices

- Use descriptive headings
- Provide alternative text for images

- Ensure keyboard navigation support
- Use sufficient color contrast

Example

```

```

The alt attribute helps screen readers describe images to visually impaired users [9, 10].

5.7 CSS Frameworks

CSS frameworks provide pre-designed styles and components that accelerate development.

Popular CSS frameworks include:

- Bootstrap
- Tailwind CSS
- Foundation

Bootstrap Example

```
<button class="btn btn-primary">  
Submit  
</button>
```

Advantages of CSS Frameworks:

- Faster development
- Responsive components
- Consistent design
- Reduced coding effort

Frameworks are widely used in modern web development projects [11, 12].

5.8 Introduction to Version Control

Version control systems help developers track changes in code.

Benefits

- Collaboration among team members
- Backup of project history
- Easy rollback of changes
- Efficient project management

Git Commands

Initialize Repository:

```
git init
```

Check Status:

```
git status
```

Add Files:

```
git add .
```

Commit Changes:

```
git commit -m "Initial Commit"
```

Git has become the industry standard for source code management [13].

5.9 Introduction to GitHub

GitHub is a cloud-based platform that hosts Git repositories.

Features

- Code hosting
- Team collaboration
- Issue tracking
- Version management

Developers use GitHub to share projects, collaborate with teams, and maintain software repositories [13, 14].

5.10 Website Performance Optimization

Website performance directly affects user experience and search engine rankings.

Common Optimization Techniques

Minification

Removing unnecessary characters from code.

Image Optimization

Compressing images to reduce file size.

Browser Caching

Storing frequently accessed resources locally.

Lazy Loading

Loading images only when needed.

Example

```

```

This delays image loading until it becomes visible on the screen.

Performance optimization leads to faster websites and better user engagement [2, 15].

5.11 Browser Compatibility

Different browsers may interpret code differently.

Popular browsers include:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Safari

Developers should test websites across multiple browsers to ensure consistent functionality.

Example

CSS vendor prefixes may be used for compatibility.

```
.box
{
  -webkit-border-radius:10px;
  border-radius:10px;
}
```

Cross-browser testing helps identify and resolve compatibility issues [1, 10].

5.12 Code Quality and Best Practices

Professional frontend developers follow coding standards to improve maintainability.

Best Practices

- Use meaningful class names
- Organize files logically
- Comment important code sections
- Avoid code duplication
- Validate HTML and CSS
- Follow accessibility standards

Example:

```
.main-navigation
{
  background-color:#333;
}
```

Meaningful naming conventions improve code readability [1, 2].

5.13 Security Considerations in Frontend Development

Frontend applications should protect users from common security risks.

Common Security Practices

- Validate user input
- Sanitize form data
- Use HTTPS connections
- Avoid exposing sensitive information

Example:

```
let username =
document.getElementById("username").value;
```

User input should always be validated before processing.

Security should be considered throughout the development lifecycle [15].

5.14 Emerging Trends in Frontend Development

Frontend development continues to evolve rapidly.

Current trends include:

- Progressive Web Apps (PWAs)
- Artificial Intelligence Integration

- Serverless Architectures
- Component-Based Frameworks
- Voice User Interfaces
- WebAssembly

Developers must stay updated with emerging technologies to remain competitive in the industry [1, 10].

Conclusion

Modern frontend development has transformed web development by combining responsive design, interactive interfaces, accessibility, performance optimization, version control, and security practices. This chapter highlighted the importance of designing websites that provide consistent and user-friendly experiences across different devices and browsers. Responsive and mobile-first approaches help developers build flexible applications, while UI and UX principles improve usability, navigation, and user engagement. Accessibility practices ensure that web applications remain inclusive and usable by people with diverse abilities. CSS frameworks and version control systems such as Git and GitHub further support efficient development, collaboration, and maintainability. Performance optimization techniques, browser compatibility testing, code quality standards, and frontend security practices contribute to reliable and professional web applications. Emerging technologies such as Progressive Web Apps, artificial intelligence, WebAssembly, voice interfaces, and component-based frameworks are further expanding the capabilities of frontend development. Therefore, a strong understanding of modern frontend practices enables developers to create efficient, accessible, secure, responsive, and high-quality web applications that meet the evolving expectations of users and the web industry.

References

1. World Wide Web Consortium (W3C). *HTML and CSS Standards*. W3C, World Wide Web Consortium.
2. Flanagan, D. *JavaScript: The Definitive Guide*. 7th ed., O'Reilly Media, 2020.
3. Keith, J., and Andrew, R. *HTML5 for Web Designers*. A Book Apart, 2010.
4. Marcotte, E. *Responsive Web Design*. A Book Apart, 2011.
5. Frain, B. *Responsive Web Design with HTML5 and CSS*. 4th ed., Packt Publishing, 2022.
6. Wroblewski, L. *Mobile First*. A Book Apart, 2011.
7. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., and Diakopoulos, N. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 6th ed., Pearson, 2016.
8. Krug, S. *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. 3rd ed., New Riders, 2014.
9. World Wide Web Consortium (W3C). *Web Content Accessibility Guidelines (WCAG) 2.2*. W3C Recommendation, 2023.
10. Mozilla Developer Network (MDN). *MDN Web Docs: Web Development Documentation*. Mozilla.
11. Otto, M., and Thornton, J. *Bootstrap Documentation*. Bootstrap.
12. Tailwind Labs. *Tailwind CSS Documentation*. Tailwind CSS.
13. Chacon, S., and Straub, B. *Pro Git*. 2nd ed., Apress, 2014.
14. GitHub. *GitHub Documentation: Collaboration and Development*. GitHub.

15. OWASP Foundation. *OWASP Top 10: The Ten Most Critical Web Application Security Risks*. OWASP Foundation.

Chapter 6

Building and Deploying a Complete Website

Ms. Nisha*, Er. Ritika Mishra¹, Dr. Pertik Garg²

***Assistant Professor**, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (nishasviet@gmail.com)
<https://orcid.org/0009-0005-0323-6443>.

1. Assistant Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601 (ritikamishra494@gmail.com) <https://orcid.org/0009-0007-6552-279X>.

2. Professor, Swami Vivekanand Institute of Engineering & Technology, Banur (Patiala) Punjab, 140601. (waytogarag@gmail.com)
<https://orcid.org/0000-0003-3757-7623>

Abstract

Building and deploying a complete website is a comprehensive process that involves a series of interconnected stages, beginning with requirement analysis and planning and extending to design, development, testing, deployment, and continuous maintenance. A successful website requires more than simply writing HTML, CSS, and JavaScript code; it demands careful consideration of usability, accessibility, responsiveness, performance, security, content organization, and user experience.

This chapter presents a systematic approach to the complete website development life cycle and explains the role of wireframing, website structure, HTML5, CSS3, JavaScript, responsive design, debugging, and testing in developing modern web applications. It further discusses web hosting, domain names, Domain Name System (DNS), deployment methods, and commonly used deployment platforms such as GitHub Pages, Netlify, and Vercel. The chapter also highlights the importance of Search Engine Optimization (SEO), regular maintenance, security updates, performance monitoring, and data backup for ensuring the long-term reliability and accessibility of websites. A case study of developing a college website is included to demonstrate the practical integration of planning, design, development, testing, and deployment concepts. Finally, emerging trends such as Progressive Web Applications, WebAssembly, Artificial Intelligence integration, cloud-native applications, and serverless computing are introduced to provide an understanding of the future direction of web development.

The chapter provides learners and aspiring developers with a practical foundation for creating responsive, accessible, secure, efficient, and user-centered websites and deploying them successfully for real-world use.

Keywords

Website Development, Deployment, Hosting, Wireframe, Testing, Web Server

Introduction

The process of building a website extends beyond writing HTML, CSS, and JavaScript code. A successful website requires careful planning, structured development, comprehensive testing, and proper deployment. Modern websites serve various purposes, including education, business, entertainment, e-commerce, and social networking.

Developers must ensure that websites are visually appealing, functional, responsive, secure, and accessible across different devices and browsers. Once developed, websites must be deployed on web servers so that users worldwide can access them through the Internet.

This chapter explains the complete lifecycle of website development and deployment, providing practical guidance for beginners [1, 2].

6.1 Website Development Life Cycle

The Website Development Life Cycle (WDLC) is a structured process used to design, develop, test, deploy, and maintain a website. Following a systematic approach helps ensure that the website meets user requirements, performs efficiently, and remains secure and up to date.

The major phases of the Website Development Life Cycle are explained below:

1. Requirement Analysis

Requirement analysis is the first phase of website development. In this stage, the developer gathers information about the client's objectives, target audience, business requirements, desired features, and technical specifications. The purpose is to clearly understand what the website should achieve and prepare a detailed project requirement document.

2. Website Planning

After gathering the requirements, a detailed plan is prepared. This includes defining the website structure, creating a sitemap, selecting technologies, estimating the project timeline and budget, and allocating resources. Proper planning ensures smooth execution of the project.

3. Wireframing

Wireframing is the process of creating a basic blueprint or layout of the website. It outlines the placement of elements such as headers, menus, images, content sections, buttons, and footers without focusing on visual design. Wireframes help developers and clients visualize the website structure before development begins.

4. Design

During the design phase, the visual appearance of the website is created. Designers select colors, fonts, images, icons, and layouts that match the organization's branding and provide a user-friendly experience. The objective is to create an attractive, responsive, and easy-to-navigate interface.

5. Development

In this phase, the actual coding of the website takes place. Front-end developers build the user interface using technologies such as HTML, CSS, and JavaScript, while back-end developers implement server-side functionality, databases, and application logic. All planned features are integrated to create a fully functional website.

6. Testing

Testing ensures that the website functions correctly before it is launched. Different types of testing are performed, including functionality testing, compatibility testing, performance testing, security testing, and usability testing. Errors and bugs identified during testing are corrected to ensure reliable performance.

7. Deployment

Deployment is the process of making the website available on the internet. The website files are uploaded to a web hosting server, the domain name is configured, and DNS settings are updated. After deployment, the live website is tested to verify that all pages and features work correctly.

8. Maintenance

Maintenance is the continuous process of updating and improving the website after deployment. It includes fixing bugs, updating content, improving security, performing software updates, monitoring performance, creating backups, and adding new features based on user feedback. Regular maintenance keeps the website secure, efficient, and relevant [1, 3].

Requirements → Planning → Design → Development → Testing → Deployment → Maintenance

Following a structured lifecycle improves project quality and reduces development errors.



Figure 6.1 Website Development Life Cycle

6.2 Website Planning

Website planning is the first and one of the most important stages of website development. It involves defining the purpose, scope, and strategy of the website before the actual design and development begin. Proper planning helps developers understand project requirements, avoid unnecessary changes during development, and ensure that the final website meets user expectations and business objectives.

Objectives of Planning:

1. Define Website Goals

The primary objective of planning is to identify the purpose of the website. The goals may include providing information, promoting products or services, selling products online, offering educational content, or improving customer engagement. Clearly defined goals guide the overall development process.

2. Determine Website Features

Planning involves deciding the features and functionalities that the website should provide. These may include user registration and login, contact forms, search functionality, image galleries, blogs, online payment systems, live chat, social media integration, and responsive design. Identifying these features in advance helps in selecting appropriate technologies and estimating development effort.

3. Create Content Structure

A well-organized content structure improves navigation and usability. During this stage, developers prepare a sitemap that outlines the hierarchy of web pages such as Home, About Us, Services, Products, Blog, Contact, and other sections. Proper content organization enables users to find information quickly and easily.

4. Estimate Development Resources

Website planning also includes estimating the resources required for successful project completion. These resources include the development team, software tools, hardware, hosting services, domain registration, budget, and project timeline. Accurate resource estimation ensures efficient project management and timely delivery.

Example

For a College Website:

Objectives:

- Provide admission information
- Display academic programs
- Publish notices and events
- Offer online inquiry forms

Proper planning minimizes future modifications and improves project success [1, 4].

6.3 Wireframing and Website Layout Design

Wireframing is an essential step in the website design process. A wireframe is a simple visual blueprint or skeletal representation of a webpage that illustrates the placement of various elements without focusing on colors, images, fonts, or other graphical details. It serves as a planning tool that helps designers, developers, and clients understand the structure and functionality of a website before the actual design and coding begin.

Wireframes provide a clear overview of the webpage layout and help ensure that the website is user-friendly, well-organized, and easy to navigate. Since they focus on structure rather than appearance, wireframes allow changes to be made quickly during the early stages of development, saving both time and cost.

It helps developers and designers:

- Visualize page structure
- Organize content
- Improve user experience
- Plan navigation systems

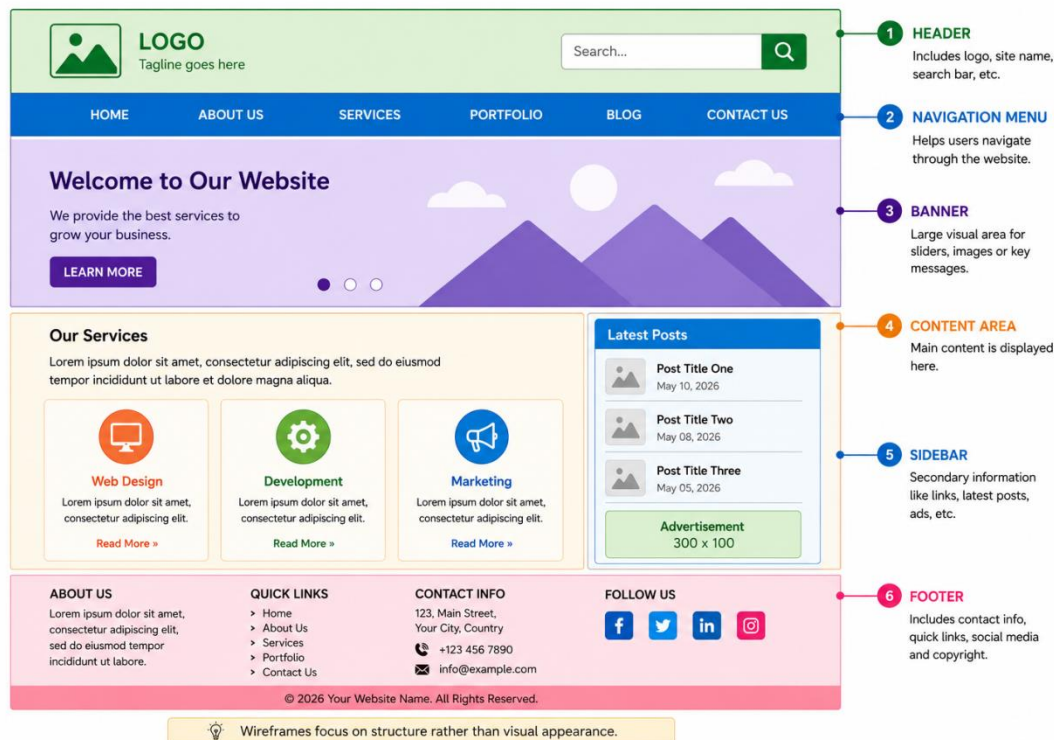


Figure 6.2 Sample Website Wireframe

Typical Homepage Sections:

A standard website homepage generally consists of the following sections:

1. Header

The header appears at the top of the webpage and usually contains the website logo, title, search bar, and important navigation links. It provides users with immediate access to key information.

2. Navigation Menu

The navigation menu contains links to major pages such as Home, About Us, Services, Products, Blog, and Contact. A well-designed navigation menu helps users browse the website efficiently.

3. Banner (Hero Section)

The banner is a large section displayed below the header. It often includes attractive images, promotional messages, announcements, or call-to-action buttons to capture visitors' attention.

4. Content Area

The content area is the main section of the webpage where the primary information, articles, products, services, or multimedia content is displayed. It is the most important part of the website.

5. Sidebar

The sidebar is an optional section placed on the left or right side of the content area. It may contain quick links, recent posts, advertisements, categories, social media links, or additional navigation options.

6. Footer

The footer is located at the bottom of the webpage. It typically contains copyright information, contact details, privacy policy, terms and conditions, social media links, and other useful navigation links.

Wireframes focus on structure rather than visual appearance [4, 5].

6.4 Designing a Website Structure

Designing a website structure is an important stage in website development. It involves organizing the web pages in a logical and hierarchical manner before coding begins. A well-planned website structure helps users navigate the website easily, improves content organization, and enhances the overall user experience. It also assists developers in building a website that is easy to maintain and expand in the future.

A clear website structure defines the relationship between different pages and ensures that visitors can quickly find the information they need. It is commonly represented using a site map or a hierarchical tree diagram.

Example Website Structure:

```
Home
|
├── About Us
├── Services
├── Gallery
├── Contact Us
└── Blog
```

1. Home

The Home page is the main entry point of the website. It provides an overview of the organization, highlights important information, and contains links to all major sections of the website. It is designed to attract visitors and guide them to relevant pages.

2. About Us

The About Us page provides information about the organization, company, institution, or individual. It typically includes the organization's history, mission, vision, objectives, achievements, and team members. This page helps build trust and credibility among visitors.

3. Services

The Services page describes the products or services offered by the organization. It explains each service in detail, including its features, benefits, pricing (if applicable), and other relevant information to help users understand what is available.

4. Gallery

The Gallery page contains images, videos, or multimedia content related to the organization, products, events, or activities. It provides visitors with a visual representation of the organization's work and achievements.

5. Contact Us

The Contact Us page enables visitors to communicate with the organization. It generally includes contact details such as address, phone number, email address, location map, contact form, and links to social media platforms.

6. Blog

The Blog page contains articles, news updates, announcements, tutorials, or informative content published regularly. Blogs help keep the website updated, improve user engagement, and enhance search engine optimization (SEO).

A well-structured website improves navigation and usability [1, 5].

6.5 Creating the Website Using HTML

HTML provides the structure of the website.

Example Homepage

```
<!DOCTYPE html>  
<html>  
<head>
```

```
<title>My College Website</title>
</head>

<body>

<header>
<h1>SVIET College</h1>
</header>

<nav>
<a href="#">Home</a>
<a href="#">About</a>
<a href="#">Courses</a>
<a href="#">Contact</a>
</nav>

<section>
<h2>Welcome to SVIET</h2>
<p>Excellence in Technical Education.</p>
</section>

<footer>
<p>Copyright 2026</p>
</footer>

</body>
</html>
```

This structure forms the foundation of a webpage [1, 6].

6.6 Styling the Website Using CSS

CSS enhances appearance and visual appeal.

Example

```
body
{
font-family: Arial;
margin: 0;
}
header
{
background-color: navy;
color: white;
padding: 20px;
```

```
text-align: center;  
}
```

```
nav  
{  
background-color: #333;  
padding: 10px;  
}
```

```
nav a  
{  
color: white;  
margin: 10px;  
text-decoration: none;  
}
```

CSS helps create attractive and professional interfaces [1, 7].

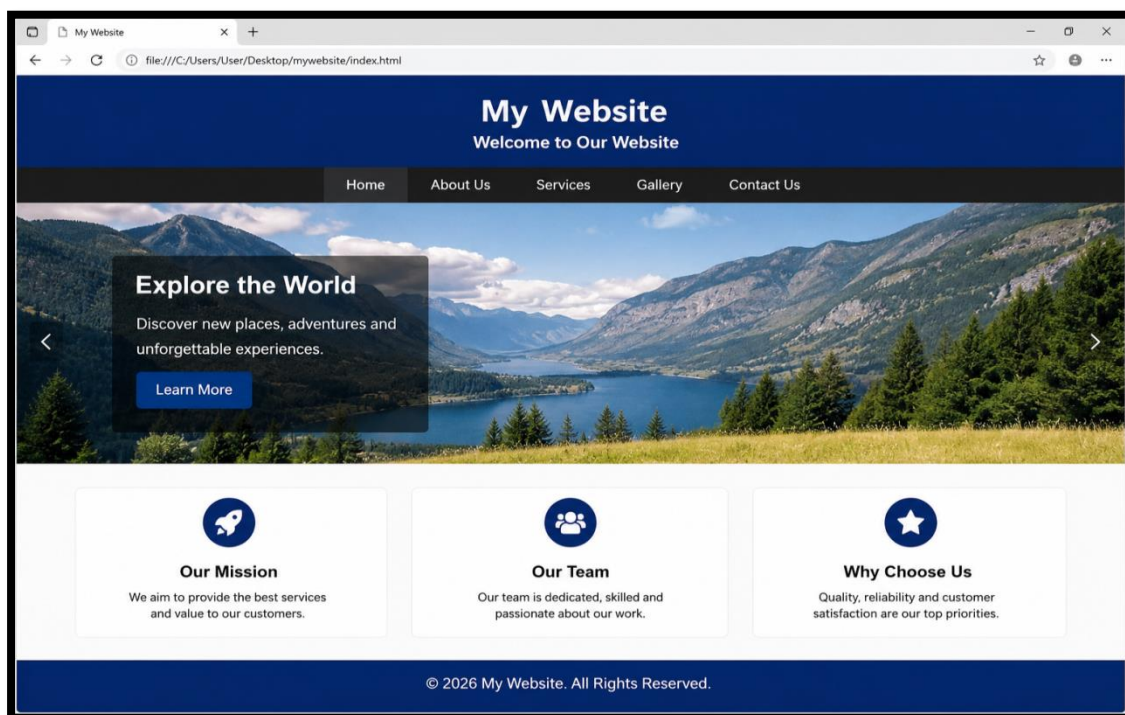


Figure 6.3 Styled Homepage Output

6.7 Adding Interactivity Using JavaScript

JavaScript makes websites interactive.

Example

```
<button onclick="showMessage()">
```

Click Here

```
</button>
```

```
<script>
```

```
function showMessage()
```

```
{
```

```
  alert("Welcome to SVIET");
```

```
}
```

```
</script>
```

Interactive elements improve user engagement and website usability [1, 8].

6.8 Creating Responsive Web Pages

Responsive websites automatically adapt to different screen sizes [2, 7].

Example

```
@media(max-width:768px)
```

```
{
```

```
  nav
```

```
{
```

```
  display:block;
```

```
}}
```

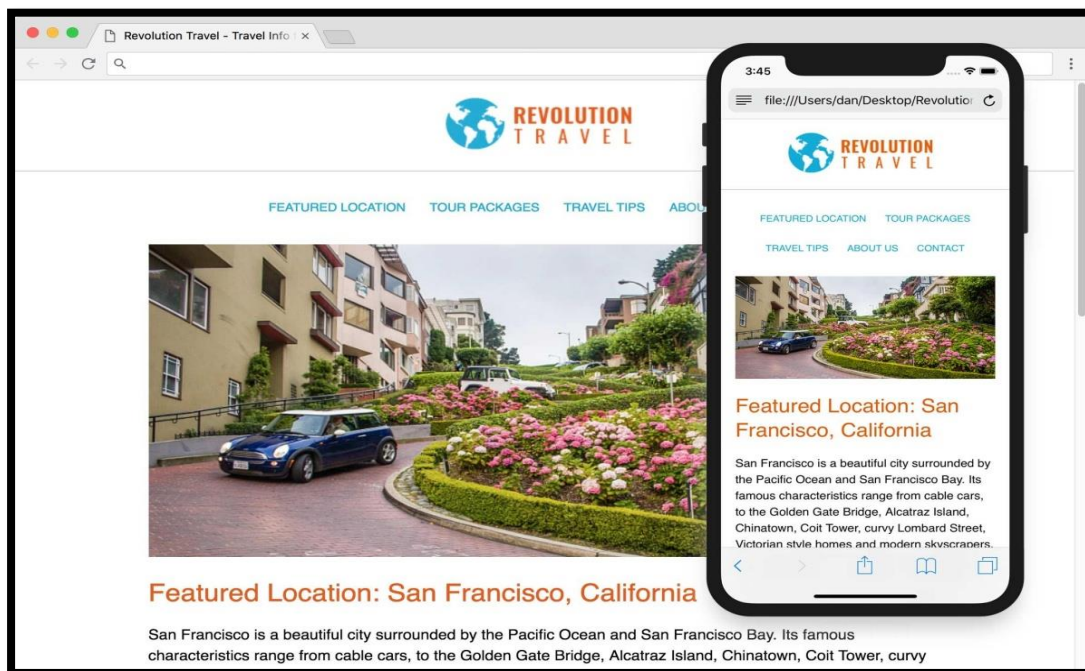


Figure 6.4 Responsive Website View

6.9 Website Testing

Website testing is the process of evaluating a website to ensure that it functions correctly, performs efficiently, and provides a smooth user experience before it is deployed. It helps identify and fix errors, broken links, security issues, and performance problems. Thorough testing ensures that the website is reliable, secure, and compatible with different devices and web browsers.

Testing should be carried out throughout the development process and before the website goes live.

Types of Testing:

1. Functional Testing

Functional testing verifies that all website features and functionalities work as intended according to the specified requirements. It checks whether buttons, links, forms, menus, search functions, login systems, and other interactive elements perform correctly.

Examples:

- Verifying navigation links.
- Testing user registration and login.
- Checking contact forms and search functionality.
- Ensuring buttons perform the correct actions.

2. Usability Testing

Usability testing evaluates how easy and convenient the website is for users. It focuses on user experience (UX), navigation, readability, accessibility, and overall satisfaction. The objective is to ensure that visitors can easily find information and complete their tasks without confusion.

Examples:

- Checking whether navigation is simple and intuitive.
- Ensuring content is easy to read.
- Evaluating the overall layout and user interface.
- Testing accessibility for different users.

3. Performance Testing

Performance testing measures how efficiently the website performs under different conditions. It evaluates page loading speed, response time, server performance, and the website's ability to handle multiple users simultaneously.

Examples:

- Measuring webpage loading time.
- Testing server response under heavy traffic.
- Evaluating website performance on slow internet connections.
- Monitoring resource usage.

4. Compatibility Testing

Compatibility testing ensures that the website functions properly across different web browsers, operating systems, devices, and screen sizes. A responsive website should provide a consistent experience regardless of the platform being used.

Examples:

- Testing on Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
- Checking functionality on Windows, macOS, Android, and iOS.
- Verifying responsiveness on desktops, tablets, and smartphones.

5. Security Testing

Security testing identifies vulnerabilities that could expose the website to cyber threats. It helps protect user data, prevent unauthorized access, and ensure secure communication between users and the server.

Examples:

- Checking for SQL injection vulnerabilities.
- Testing authentication and authorization mechanisms.
- Verifying HTTPS implementation.
- Identifying cross-site scripting (XSS) vulnerabilities.
- Ensuring sensitive data is securely stored and transmitted [3, 9].

6.10 Debugging Web Applications

Debugging is the process of identifying and fixing errors.

Common Tools:

- Browser Developer Tools
- Console Logs
- Error Inspectors

Example

```
console.log("Testing Website");
```

Developers tools help diagnose issues quickly [1, 8].

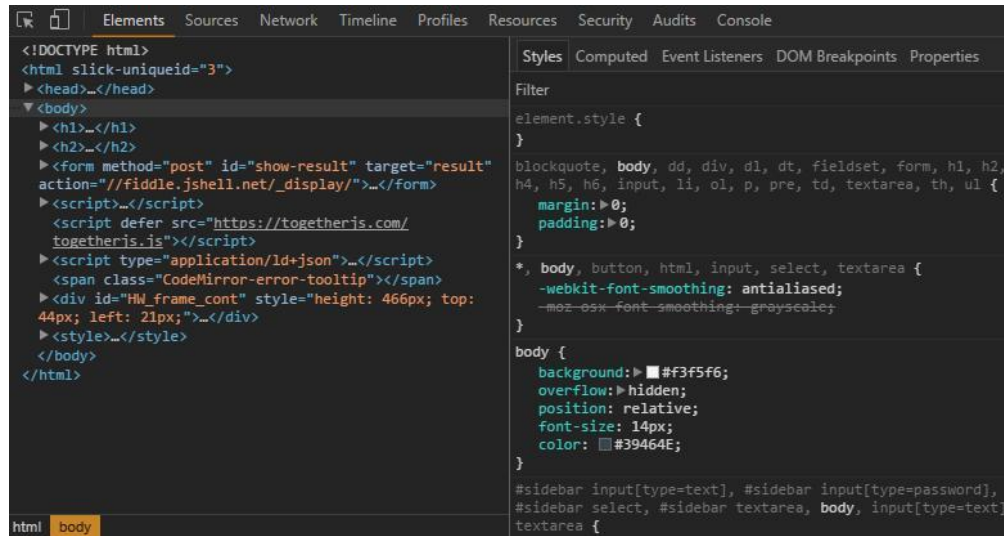


Figure 6.5 Browser Developer Tools

6.11 Web Hosting

Web hosting is a service that provides storage space and computing resources on a web server to make a website accessible over the Internet. All website files, including HTML pages, CSS stylesheets, JavaScript files, images, videos, databases, and other resources, are stored on a hosting server. When users enter a website's domain name in a web browser, the hosting server delivers the requested web pages to their devices.

Choosing the appropriate web hosting service is essential because it affects the website's performance, security, reliability, scalability, and overall user experience.

Types of Hosting:

1. Shared Hosting

Shared hosting is the most common and cost-effective type of web hosting. In this hosting model, multiple websites share the same server resources, such as CPU, memory, storage, and bandwidth. It is suitable for small websites, blogs, personal websites, and startups with low to moderate traffic.

2. VPS (Virtual Private Server) Hosting

VPS hosting divides a physical server into several virtual servers using virtualization technology. Each virtual server has dedicated resources, providing better performance, security, and control than shared hosting.

3. Dedicated Hosting

Dedicated hosting provides an entire physical server exclusively for a single website or organization. Since all server resources are allocated to one user, it offers maximum performance, security, and customization.

4. Cloud Hosting

Cloud hosting uses a network of interconnected servers to host websites instead of relying on a single physical server. Resources can be scaled up or down based on website traffic, making cloud hosting highly flexible and reliable.

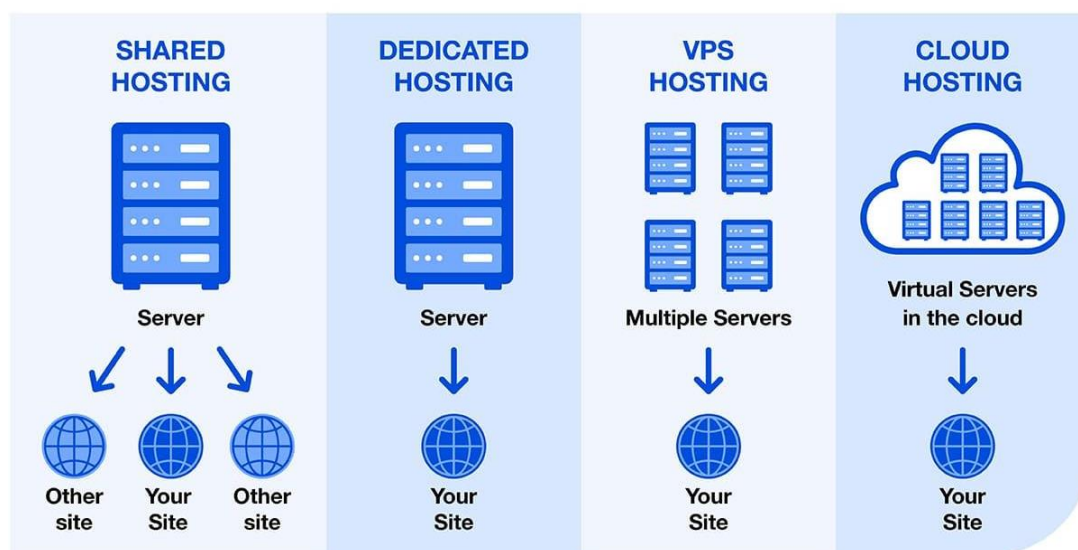


Figure 6.6 Types of Web Hosting

Selecting appropriate hosting improves website reliability and performance [10, 11].

6.12 Domain Names and DNS

A domain name is a unique, human-readable address used to identify a website on the Internet. Instead of remembering a numerical IP (Internet Protocol) address, users can simply enter a domain name in a web browser to access a website. Domain names make websites easy to locate, remember, and share.

Every website on the Internet has an IP address, which is a series of numbers used by computers to communicate. Since numerical IP addresses are difficult to remember, the Domain Name System (DNS) translates domain names into their corresponding IP addresses, allowing users to access websites using simple names.

Examples:

- www.sviet.ac.in
- www.google.com
- www.wikipedia.org

DNS (Domain Name System)

The Domain Name System (DNS) is often called the "phonebook of the Internet." It is a distributed system that translates domain names into IP addresses, allowing web browsers to locate the correct web server hosting a website [12].

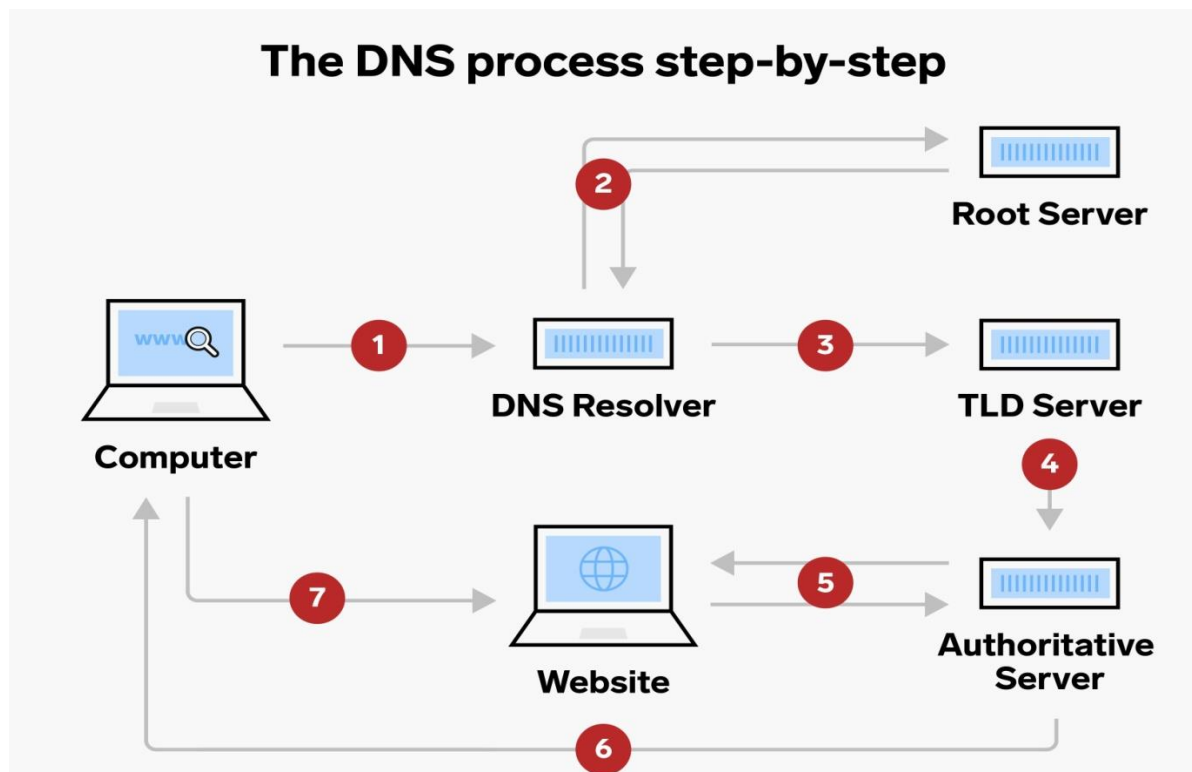


Figure 6.7 Domain Name Resolution Process

6.13 Deploying a Website

Website deployment is the process of making a completed website available on the Internet so that users can access it through a domain name or public URL. After development and testing, website files are transferred to a web server, DNS settings are configured, and the website becomes live.

Deployment Steps:

1. Purchase Domain Name

A domain name is the unique address of a website, such as www.example.com. Users enter this address in a browser to access the website.

Example:

- www.google.com
- www.wikipedia.org

The domain name is connected to the hosting server through DNS configuration.

2. Purchase Hosting Plan

Web hosting provides server space where website files, databases, images, and other resources are stored.

Common hosting types include:

- Shared Hosting
- Cloud Hosting
- VPS Hosting
- Dedicated Hosting

After purchasing hosting, the provider gives access details such as control panel login, FTP credentials, and server information.

3. Upload Website Files

The developed website files are uploaded to the hosting server.

Website files may include:

- HTML files
- CSS files
- JavaScript files
- Images
- Database files

Files can be uploaded using:

- File Manager provided by hosting companies
- FTP (File Transfer Protocol) clients such as FileZilla
- Version control platforms like GitHub

4. Configure DNS

DNS (Domain Name System) connects the domain name with the hosting server's IP address.

The DNS configuration process involves:

- Updating Name Servers (NS Records)
- Adding A Records to connect the domain with an IP address
- Adding CNAME Records for subdomains

After DNS propagation, users can access the website using the domain name.

5. Test Live Website

After deployment, the website should be tested to ensure proper functioning.

Testing includes:

- Checking all pages and links
- Testing website speed
- Verifying mobile responsiveness
- Checking forms and database connections
- Ensuring security settings are working properly

Deployment Tools:

1. File Manager

A web-based tool provided by hosting companies to upload, edit, and manage website files directly from a browser.

2. FTP Client

FTP clients allow users to transfer website files between a local computer and a web server.

Examples:

- FileZilla
- WinSCP

3. GitHub Pages

GitHub Pages is a free hosting service that publishes websites directly from a GitHub repository. It is commonly used for static websites such as portfolios, documentation sites, and project pages.

4. Netlify

Netlify is a cloud-based deployment platform that allows developers to deploy websites automatically from Git repositories.

5. Vercel

Vercel provides fast deployment and hosting solutions, especially for modern web applications built using frameworks like Next.js and React [10,11, 13].

Example Using GitHub Pages

1. Upload Project to GitHub
2. Open Repository Settings
3. Select Pages
4. Publish Website
5. Access Generated URL

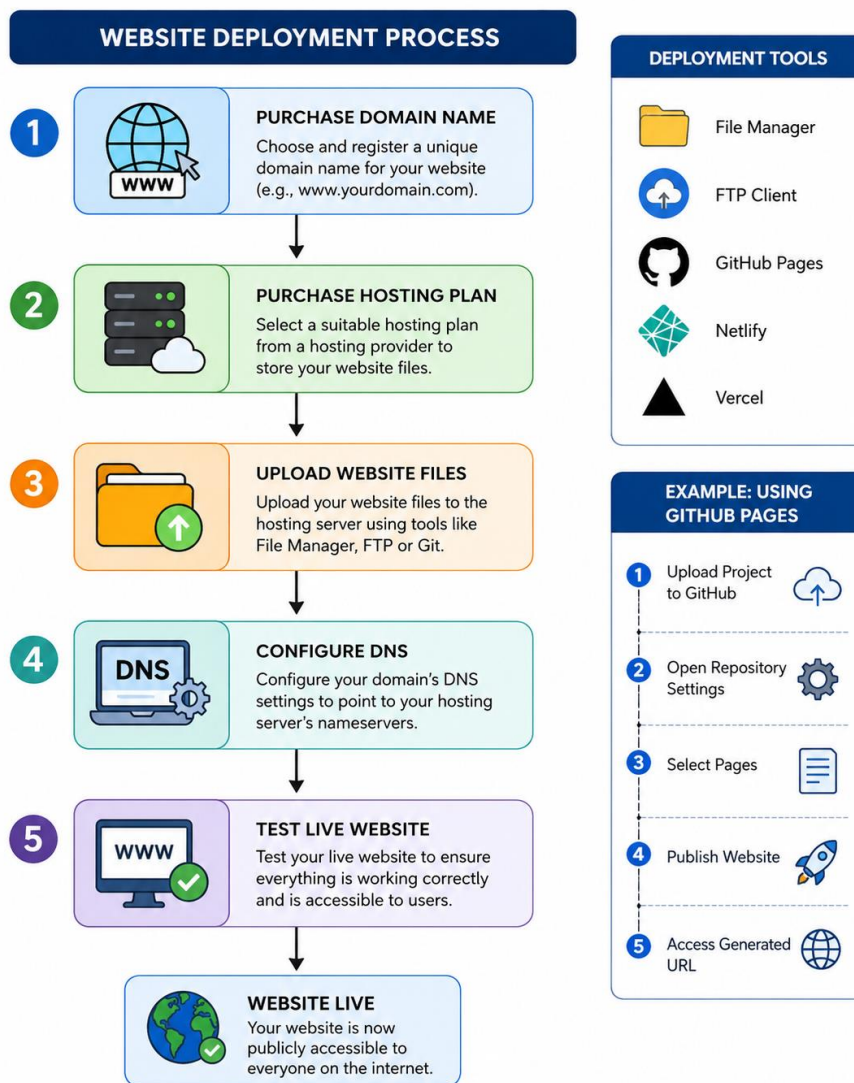


Figure 6.8 Website Deployment Process

6.14 Website Maintenance

Website maintenance is the continuous process of updating, monitoring, improving, and protecting a website after it has been launched. Unlike website development, which happens during the creation stage, maintenance is an ongoing activity required to keep the website secure, functional, fast, and user-friendly. Regular maintenance helps prevent security issues, data loss, outdated information, and performance problems.

Maintenance Tasks:

1. Update Content

Website content should be updated regularly to provide accurate and fresh information to users.

Content updates include:

- Adding new articles, announcements, and news
- Updating contact details and services
- Removing outdated information
- Updating images and multimedia content

Fresh content improves user experience and helps maintain website visibility.

2. Fix Broken Links

Broken links are hyperlinks that no longer work or lead to unavailable pages. They create a poor user experience and may affect website credibility.

Maintenance activities include:

- Checking internal and external links
- Removing outdated links
- Redirecting deleted pages to new locations
- Fixing 404 (Page Not Found) errors

Regular link checking ensures smooth navigation for visitors.

3. Improve Security

Website security maintenance protects websites from cyber threats, unauthorized access, and malware attacks.

Security tasks include:

- Installing software and plugin updates
- Updating security certificates (SSL)
- Monitoring suspicious activities
- Using strong passwords and access controls
- Scanning for malware and vulnerabilities

Regular security updates reduce the risk of attacks and keep user data protected.

4. Monitor Performance

Website performance monitoring ensures that the website loads quickly and works properly on different devices.

Performance monitoring includes:

- Checking page loading speed
- Monitoring website uptime
- Optimizing images and files
- Reducing unnecessary scripts
- Testing mobile responsiveness

A fast and reliable website improves user satisfaction and overall experience.

5. Backup Data Regularly

Backups create copies of website files, databases, and important information. They help restore the website if data is lost due to server failure, hacking, or accidental deletion.

Backup activities include:

- Creating automatic backups
- Storing backups in secure locations
- Testing backup restoration
- Maintaining multiple backup versions

Regular backups ensure business continuity and prevent permanent data loss [3, 9].

6.15 Search Engine Optimization (SEO)

Search Engine Optimization (SEO) is the process of improving a website so that it appears higher in search engine results pages (SERPs) such as Google Search. SEO helps search engines understand website content and makes it easier for users to find relevant information. A well-optimized website can attract more visitors, improve online visibility, and increase user engagement.

SEO Techniques:**1. Use Semantic HTML**

Semantic HTML means using meaningful HTML tags that clearly describe the structure and purpose of website content.

Examples:

- `<header>` – Defines the top section of a webpage
- `<nav>` – Defines navigation links
- `<article>` – Represents independent content
- `<footer>` – Defines the bottom section of a webpage

Semantic HTML helps search engines understand page structure and improves accessibility.

2. Optimize Page Titles

The page title (<title> tag) is an important element that describes the topic of a webpage. Search engines use titles to understand page content and display them in search results.

Good page title example:

```
<title>Best Engineering College in Punjab | SV Institute</title>
```

A good title should be:

- Unique for each page
- Short and descriptive
- Related to page content
- Include important keywords naturally

3. Add Meta Descriptions

A meta description provides a short summary of a webpage. Search engines may use this description in search results to help users understand the page before clicking.

Example:

```
<metaname="description"  
content="Best Engineering College in Punjab">
```

A good meta description should:

- Describe page content clearly
- Encourage users to visit the website
- Include relevant keywords
- Be unique for each webpage

4. Use Descriptive URLs

A descriptive URL helps both users and search engines understand the purpose of a webpage.

Example:

Poor URL:

www.example.com/page123

Better URL:

www.example.com/engineering-courses

Benefits of descriptive URLs:

- Easier for users to understand
- Improves website organization
- Helps search engines identify page topics

5. Improve Page Speed

Website loading speed is an important factor for user experience. Faster websites provide better usability and reduce visitor drop-off.

Methods to improve page speed:

- Compress images
- Reduce unnecessary JavaScript and CSS files
- Use browser caching
- Optimize website code
- Use reliable hosting services

A fast-loading website provides a better experience for users and supports effective SEO practices [14].

6.16 Future Trends in Web Development

Future trends in web development focus on creating websites and web applications that are faster, smarter, more secure, and highly interactive. Emerging technologies such as Artificial Intelligence, cloud computing, WebAssembly, and Progressive Web Applications are changing the way developers design and deliver modern web solutions.

Trends:**1. Progressive Web Applications (PWAs)**

Progressive Web Applications (PWAs) are modern web applications that combine the features of websites and mobile applications. They provide an app-like experience directly through a web browser without requiring users to install applications from an app store.

Features of PWAs:

- Work offline using service workers
- Load quickly even on slow networks
- Support push notifications
- Can be installed on mobile devices
- Provide responsive user experiences

2. Artificial Intelligence Integration

Artificial Intelligence (AI) is transforming web development by making websites more intelligent and personalized. AI helps websites understand user behavior and provide automated services.

Applications of AI in Web Development:

- AI-powered chatbots for customer support
- Personalized content recommendations
- Automated website design assistance
- Predictive analysis of user behavior
- Intelligent search systems

3. Voice Search Optimization

Voice search optimization focuses on designing websites that can respond effectively to voice-based queries through digital assistants such as Google Assistant, Siri, and Alexa.

Voice Search Techniques:

- Use natural language keywords
- Optimize content for question-based searches
- Provide clear and structured information
- Improve website loading speed

4. WebAssembly (Wasm)

WebAssembly is a technology that allows programming languages such as C, C++, and Rust to run inside web browsers at high speed. It improves the performance of web applications by enabling near-native execution speed.

Uses of WebAssembly:

- Online gaming
- Video editing applications
- 3D graphics
- Scientific simulations
- High-performance computing

5. Cloud-Native Applications

Cloud-native applications are web applications designed specifically to run on cloud platforms. They use technologies such as microservices, containers, and automated deployment systems.

Features:

- High scalability
- Flexible resource management
- Faster updates
- Improved reliability

6. Serverless Computing

Serverless computing is a cloud-based approach where developers build and run applications without managing physical servers. Cloud providers automatically handle server maintenance, scaling, and infrastructure management [2, 6, 15].

Advantages of Serverless Computing:

- Reduced server management
- Automatic scaling according to demand
- Lower operational costs
- Faster application deployment

6.17 Case Study: Building a College Website

A college website is an online platform that provides information about an educational institution to students, parents, faculty members, and visitors. It acts as a digital identity of the college by presenting details about courses, faculty, campus facilities, admissions, notices, and contact information. Modern college websites are developed using front-end technologies such as HTML5, CSS3, and JavaScript to create responsive and interactive web pages

Requirements:

The main requirements for developing a college website include the following pages:

1. Homepage

The homepage is the main entry page of the website. It provides a brief introduction about the college and highlights important information.

Features include:

- College name and logo
- Navigation menu
- Latest announcements
- Campus images
- Important links
- Admission information

2. About Us Page

The About Us page provides detailed information about the institution.

Content includes:

- College history
- Vision and mission
- Achievements
- Infrastructure details
- Accreditation and approvals

This page helps visitors understand the background and reputation of the college.

3. Courses Page

The Courses page displays academic programs offered by the institution.

Information includes:

- Course names
- Duration
- Eligibility criteria
- Department details
- Curriculum information

Example:

- B.Tech Programs
- MBA Programs
- Pharmacy Programs

4. Faculty Information

The faculty section provides details about teaching staff.

Details include:

- Faculty names
- Designations
- Qualifications
- Department information
- Research interests

This section helps students know about academic experts available in the institution.

5. Contact Form

A contact page allows users to communicate with the college.

Features include:

- Name field
- Email address
- Phone number
- Message box
- Submit button
- College address and location details

The form helps students and visitors send queries easily.

Technologies Used:

1. HTML5

HTML5 is used to create the basic structure of the website.

Uses:

- Creating webpage layouts
- Adding headings, paragraphs, images, and links
- Creating forms and navigation menus
- Using semantic elements such as header, section, and footer

2. CSS3

CSS3 is used for designing and improving the visual appearance of the website.

Uses:

- Applying colors and fonts
- Creating layouts
- Designing buttons and menus
- Making the website responsive for different screen sizes

CSS media queries and flexible layouts help websites work properly on desktops, tablets, and mobile devices.

3. JavaScript

JavaScript adds interaction and dynamic functionality to the website.

Uses:

- Form validation
- Image sliders
- Interactive menus
- Dynamic content updates
- User interface enhancements

JavaScript helps create a more engaging experience for website visitors.

Development Process:

The college website development process follows several stages:

1. Planning

In this stage, requirements are collected and the website structure is decided.

Activities include:

- Identifying target users
- Selecting required pages
- Preparing website layout
- Collecting content and images

2. Design

The design stage focuses on creating the visual appearance of the website.

Tasks include:

- Designing page layouts
- Selecting colors and fonts
- Creating navigation structure
- Preparing responsive designs

3. Development

During development, the actual coding of the website is performed.

Activities include:

- Creating HTML pages
- Applying CSS styles
- Adding JavaScript features
- Integrating forms and interactive elements

4. Testing

Testing ensures that the website works correctly before deployment.

Testing includes:

- Checking links
- Testing forms
- Checking browser compatibility
- Testing mobile responsiveness
- Identifying and fixing errors

5. Deployment

After successful testing, the website is published on a web server.

Deployment activities include:

- Purchasing domain and hosting
- Uploading website files
- Configuring DNS
- Making the website available online

Outcome:

The final outcome is a responsive and user-friendly college website that can be accessed from different devices such as desktops, tablets, and smartphones.

Benefits of the developed website:

- Provides 24×7 access to college information
- Improves communication between college and users
- Helps students explore courses and facilities
- Provides online enquiry facilities
- Enhances the digital presence of the institution

A well-designed college website serves as an effective communication platform and improves the overall online experience for students, faculty, and visitors [1, 2, 3].

Conclusion

Building and deploying a complete website is a systematic process that involves planning, designing, development, testing, deployment, and continuous maintenance. This chapter explained the major stages of the website development life cycle, including requirement analysis, wireframing, HTML structure, CSS styling, JavaScript interactivity, responsive design, debugging, testing, hosting, domain configuration, DNS, deployment, SEO, and maintenance. It emphasized that a successful website

should be functional, responsive, accessible, secure, user-friendly, and optimized for performance across different devices and browsers. The case study of a college website demonstrated how these concepts can be integrated to create a practical web solution. With emerging technologies such as Progressive Web Applications, WebAssembly, Artificial Intelligence, cloud-native applications, and serverless computing, developers must continuously enhance their skills and adopt modern practices to build efficient, reliable, and future-ready websites.

References

1. J. Duckett, *HTML and CSS: Design and Build Websites*. Indianapolis, IN, USA: Wiley, 2014.
2. J. Frain, *Responsive Web Design with HTML5 and CSS*, 4th ed. Birmingham, U.K.: Packt Publishing, 2022.
3. M. A. W. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2019.
4. J. Garrett, *The Elements of User Experience: User-Centered Design for the Web and Beyond*, 2nd ed. Berkeley, CA, USA: New Riders, 2011.
5. S. Krug, *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*, 3rd ed. Berkeley, CA, USA: New Riders, 2014.
6. Mozilla Developer Network (MDN), "HTML: HyperText Markup Language," *MDN Web Docs*.
7. Mozilla Developer Network (MDN), "CSS: Cascading Style Sheets," *MDN Web Docs*.
8. Mozilla Developer Network (MDN), "JavaScript," *MDN Web Docs*. Mozilla Developer Network (MDN), "Web Developer Tools and Web Development," *MDN Web Docs*.
9. GitHub, "GitHub Pages Documentation," *GitHub Docs*.
10. Netlify, "Netlify Documentation: Deploying Websites," *Netlify Docs*.
11. P. Mockapetris, "Domain Names—Concepts and Facilities," RFC 1034, Internet Engineering Task Force (IETF), 1987.
12. Vercel, "Vercel Documentation," *Vercel Docs*.
13. Google Search Central, "SEO Starter Guide," *Google for Developers*.
14. World Wide Web Consortium (W3C), *Web Content Accessibility Guidelines (WCAG) 2.2*, W3C Recommendation, Dec. 2024.

Editor's Biography



Ms. Nisha is an experienced academician and education professional with 9 years of teaching experience. She is currently serving as an Assistant Professor at Swami Vivekanand Institute of Engineering & Technology (SVIET), Banur, Punjab. She holds an MCA and B.Sc. in Computer Applications and is a Life Member of the Indian Society for Technical Education (ISTE). Alongside her teaching responsibilities, Ms. Nisha is actively involved in academic and administrative operations, contributing to institutional planning, coordination, and quality enhancement initiatives. She serves as an Accreditation & Ranking Coordinator and contributes to institutional documentation, academic coordination, compliance, and quality-related activities. With a strong commitment to academic excellence and professional growth, Ms. Nisha actively contributes to institutional development through her teaching, coordination, and administrative responsibilities. Her expertise in academic and administrative processes enables her to effectively support institutional initiatives, quality enhancement, and student development. She continues to contribute to creating a positive academic environment and promoting excellence in teaching, learning, and institutional practices.



Er. Ritika Mishra is an accomplished academician and researcher in the field of Computer Science and Engineering, with a strong commitment to teaching, research, innovation, and academic development. She currently serves as an Assistant Professor at Swami Vivekanand Institute of Engineering & Technology (SVIET), Banur, Punjab, and brings around 10 years of experience in academics. Her professional journey encompasses teaching, student mentoring, academic coordination, research, and institutional academic initiatives. She holds a Master's degree in Computer Science from Sri Guru Granth Sahib World University and a Bachelor's degree in Information Technology from SVIET. She is also actively involved in academic planning, institutional quality initiatives, accreditation-related documentation, research activities, faculty coordination, and professional development, reflecting her broader engagement with academic and institutional advancement. Her professional contributions have also received recognition through Best Faculty and Appreciation Awards, reflecting her commitment to teaching, institutional responsibilities, and academic excellence.



Prof. (Dr.) Pertik Garg is a distinguished academician, researcher, and educational administrator with over 20 years of experience in teaching, research, and academic leadership. He currently serves as Professor and Principal at Swami Vivekanand Institute of Engineering & Technology, Banur, Punjab. He holds a Ph.D. in Computer Science and Engineering from Punjab Technical University, Jalandhar, along with M.Tech and B.Tech degrees in the same discipline. Throughout his academic career, he has undertaken diverse teaching, research, and administrative responsibilities, including project supervision, institutional coordination, training and placement, academic planning, and student development. His research contributions encompass various areas of Computer Science and Engineering. He has published 23 research papers in national and international journals and has also contributed to academic books. He is actively involved in professional and academic initiatives and is a Life Member of the Indian Society for Technical Education (ISTE). His commitment to research, innovation, and quality education has contributed significantly to the advancement of technical and higher education. Through his academic leadership and research-oriented approach, he continues to foster excellence, innovation, and the holistic development of students and faculty.

e-ISBN

ISBN: 978-81-68473-23-2



9

788168

473232



p-ISBN

ISBN: 978-81-68473-22-5



9

788168

473225