Chapter 1: Internet and World Wide Web

1.1 Working of Internet

The Internet is a worldwide network of millions of computers connected together. These computers share information using a set of rules called **protocols** (the most common is **TCP/IP – Transmission Control Protocol / Internet Protocol**).

When you open a website or send an email, your computer sends a request through your Internet Service Provider (ISP). The ISP passes this request to the destination computer (such as a web server). The web server then sends back the information to your computer. This process happens in seconds, which makes communication fast and easy.

In simple words, the Internet works like a **post office**, but instead of letters, it delivers **data packets** between computers all over the world.

1.1.1 Benefits of Internet

The Internet provides many benefits in our daily life. Some of the major benefits are:

1. **Communication** – We can send emails, use chat apps, and make video calls with people worldwide.

2. **Information Access** – The Internet is the biggest source of information. Students, teachers, and professionals can easily search anything they want.

3. **Education** – Online classes, e-books, tutorials, and educational videos help students learn easily.

4. **Entertainment** – We can watch movies, listen to music, play games, and use social media for fun.

5. **E-Commerce** – People can buy and sell products online from anywhere.

6. **Banking** – Online banking allows people to transfer money, pay bills, and shop safely.

7. **Job Opportunities** – Many people work online as freelancers or remote workers.

1.1.2 Ethics of Internet

Although the Internet is very useful, it must be used responsibly. The rules and guidelines for good behavior on the Internet are called **Internet Ethics**.

Some important Internet ethics are:

1. **Do not spread false information** – Always check the truth before sharing.
2. **Respect others' privacy** – Do not hack or share personal data of others.
3. **Avoid cyberbullying** – Never use bad language or harass people online.
4. **Follow copyright laws** – Do not copy or steal others' work without permission.
5. **Use the Internet for positive purposes** – Focus on learning, work, and communication instead of wasting time.
6. **Be careful with online transactions** – Always use trusted websites to stay safe from fraud.

1.2 Services of Internet

The Internet is not just about connecting computers; it provides many services that people use in daily life. These services make communication, learning, shopping, and entertainment easier. Below are some important services of the Internet.

1.2.1 World Wide Web (WWW)

- The **World Wide Web (WWW)** is the most popular service of the Internet.
- It is a collection of millions of web pages that contain text, pictures, videos, and links.
- People can access these web pages using a **web browser** (like Chrome, Firefox, or Edge).

- Websites are connected through **hyperlinks**, which allow users to move from one page to another easily.
- Example: Using Google to search information, watching YouTube videos, or reading online news.

👉 **In simple words:** WWW is like a huge digital library where you can find almost everything.

1.2.2 Email (Electronic Mail)

- Email is one of the oldest and most widely used Internet services.
- It allows people to send and receive messages electronically within seconds.
- Emails can contain text, documents, pictures, and videos as **attachments**.
- Every user has a unique **email address** (example: ali@gmail.com).
- Free email services: Gmail, Yahoo Mail, Outlook.

👉 **Example:** A teacher can send assignments to students via email.

1.2.3 Social Networking

- Social networking websites connect people and allow them to share thoughts, pictures, and videos.
- They help people communicate with family, friends, and even strangers worldwide.
- Social networking is also used for **business promotion, education, and entertainment**.
- Popular platforms: Facebook, Twitter (X), Instagram, LinkedIn, TikTok.

👉 **Example:** A student can join a study group on Facebook to learn with classmates.

1.2.4 Mailing List

- A mailing list is a group of email addresses used to send the same information to many people at once.
- It is useful for **organizations, colleges, or companies** that want to share announcements or newsletters.
- Members of the mailing list automatically receive all emails sent to the group.

👉 **Example:** A college can create a mailing list to send exam schedules to all students.

1.2.5 News Group

- A newsgroup is an online discussion forum where people can post messages about a specific topic.
- It works like a **bulletin board** where users can share ideas, ask questions, and give answers.
- Newsgroups are divided into categories such as technology, health, sports, education, etc.
- Although they are less popular today (replaced by social media), they are still used in some areas.

👉 **Example:** A newsgroup on programming where students discuss coding problems.

1.3 Web Browser

A **Web Browser** is a software application that allows users to access and interact with information available on the **World Wide Web (WWW)**. It acts as a bridge between the user and the internet, enabling them to view web pages, download content, watch videos, and use online applications. Common examples include **Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, and Opera**.

A web browser works by sending a request to a **web server** for a webpage and then displaying it on the user's device. It interprets the code written in HTML, CSS, and JavaScript to show content in a readable and interactive way.

1.3.1 Functions of Web Browser

Web browsers provide several important functions that make browsing possible and convenient:

1. **Accessing Websites** – Browsers allow users to enter a website address (URL) and open webpages stored on different servers.

2. **Rendering Web Pages** – Browsers convert the HTML, CSS, and scripts into a visual display so that users can read and interact with websites easily.

3. **Navigation Tools** – They provide buttons like **Back, Forward, Refresh, and Home** to move between web pages.

4. **Bookmarking** – Users can save their favorite websites for quick access later.

5. **Downloading & Uploading** – Browsers support downloading files, images, and videos as well as uploading documents or pictures to websites.

6. **Security & Privacy** – Modern browsers include security features such as **private browsing (incognito), blocking pop-ups, detecting phishing sites, and managing cookies**.

7. **Extensions & Add-ons** – Browsers allow additional tools (like ad blockers, translators, or password managers) to improve functionality.

8. **Search Integration** – Most browsers include a search bar to quickly find information using search engines like Google or Bing.

1.3.2 Types of Browsers

There are several types of web browsers based on their design, purpose, and usage:

1. **Text-Based Browsers** –

- These only display text, without images or multimedia.
- Example: **Lynx**.
- Used in simple or low-resource environments.

2. **Graphical Browsers** –
   - Display both text and multimedia (images, videos, animations).
   - Most popular browsers today (Chrome, Firefox, Edge, Safari, Opera).
3. **Mobile Browsers** –
   - Designed for smartphones and tablets with smaller screens.
   - Examples: **Google Chrome Mobile, Safari (iOS), Samsung Internet**.
4. **Specialized Browsers** –
   - Built for specific purposes, such as accessing only one platform or offering higher security.
   - Examples: **Tor Browser** (for anonymous browsing), **Brave** (focus on privacy).
5. **Voice-Based Browsers** –
   - Allow users to interact with the internet through voice commands instead of typing.
   - Example: **Browsers in smart assistants (Google Assistant, Siri)**.

1.4 Web Server

A **web server** is a computer that stores websites and delivers them to users when requested through a web browser. When you type a website address in your browser, the request goes to the web server. The server processes the request and sends back the required web pages.

- Example: If you type *www.google.com*, your browser sends a request to Google's web server, and the server responds by sending the homepage of Google.
- A web server works with two main parts:

1. **Hardware** – the physical computer that stores website files.
2. **Software** – the program that delivers the website (like Apache, Nginx, Microsoft IIS).

In short: Web server = storage + delivery of websites to users.

1.5 Web Directories

A **web directory** is a collection of websites arranged in categories and subcategories. It helps users find websites based on topics. Unlike search engines, directories are organized manually by humans.

- Example: If you want to find websites about *education*, you can go to a web directory and choose the *Education* category to see related sites.
- Famous example: Yahoo Directory (very popular in early days of the web).

Difference from Search Engine:

- **Web Directory**: Organized manually, sites are grouped by category.
- **Search Engine**: Uses software (robots/crawlers) to find sites automatically.

1.6 Websites

A **website** is a collection of web pages stored on a web server. A website has a unique address called a **URL (Uniform Resource Locator)**, such as *www.facebook.com*.

Websites are mainly divided into two types:

1.6.1 Static Websites

- A **static website** shows fixed content.
- Every visitor sees the same page.

- These pages are created using HTML and CSS.
- They do not change unless the developer updates the content manually.
- Example: A company's profile website that shows its history, services, and contact details.

Advantages:

- Simple and easy to create.
- Loads faster.

Disadvantages:

- Cannot show updated or personalized content automatically.

1.6.2 Dynamic Websites

- A **dynamic website** shows content that changes according to user interaction or database information.
- These websites use programming languages like PHP, JavaScript, or ASP.NET.
- Example: Facebook, YouTube, or Online Shopping websites (content changes for every user).

Advantages:

- Provides fresh and updated information.
- Allows users to interact (comments, likes, accounts).

Disadvantages:

- More complex to create.
- Needs more server resources.

1.7 Search Engine

A **search engine** is a software system that helps users find information on the Internet quickly and efficiently. When a user types a keyword or phrase, the search engine searches its database and provides a list of relevant websites.

Popular search engines: Google, Bing, Yahoo, DuckDuckGo.

Functions of a Search Engine:

1. **Crawling:** Search engines use bots (called spiders) to browse the web and find new pages.
2. **Indexing:** All collected pages are stored in a database for quick access.
3. **Searching:** When a user enters a keyword, the engine looks through its index for relevant results.
4. **Ranking:** The most relevant results are displayed at the top based on algorithms.

Example: If you search *Python programming tutorial*, the search engine will show websites, videos, and articles related to Python tutorials.

1.8 Web Page Program Development

Creating a website is called **web development**. It involves designing, coding, testing, and maintaining web pages. Web development can be **simple static pages** or **complex dynamic applications**.

Web development is usually done by a team of people with different roles.

1.8.1 Roles in Website Development Team

1. **Web Designer:**
    - Creates the layout, colors, graphics, and overall appearance of the website.
2. **Web Developer:**

- Writes the code using HTML, CSS, JavaScript, PHP, or other languages.

3. **Content Writer:**
    - Prepares text, articles, blogs, and other written content for the website.
4. **SEO Expert:**
    - Optimizes the website so that it ranks higher in search engines.
5. **Project Manager:**
    - Manages the development team, assigns tasks, and ensures the project is completed on time.
6. **Database Administrator (if needed):**
    - Manages databases for dynamic websites, like user accounts or online stores.

1.8.2 Web Development Scope

Web development has a **wide scope** in today's digital world.

- **Businesses**: Online stores, company websites, customer portals.
- **Education**: Online learning platforms, schools, and colleges.
- **Entertainment**: Video streaming websites, gaming platforms, music websites.
- **Government**: Public service portals, online forms, e-governance systems.

Career opportunities in web development:

- Frontend Developer (focuses on design and interface)
- Backend Developer (focuses on server, database, and logic)
- Full Stack Developer (handles both frontend and backend)
- Web Designer, Content Writer, SEO Specialist

Summary:
Web development is not just coding; it requires teamwork, creativity, and understanding the purpose of the website. With the Internet growing every day, web developers are always in high demand.

1.9 Scripting Languages

A **scripting language** is a type of programming language used to create dynamic content on websites. Unlike static pages, which only show fixed content, scripting languages allow websites to **interact with users, process data, and update information automatically**.

Scripting languages are often used to:

- Validate forms on websites (e.g., check if an email is correct)
- Create animations or interactive buttons
- Communicate with databases to store or retrieve data
- Customize the user experience on websites

The two most popular scripting languages used in web development are **JavaScript** and **PHP**.

1.9.1 JavaScript

- **JavaScript** is a **client-side scripting language**, which means it runs on the user's web browser.
- It is mainly used to **make websites interactive** and responsive.
- Example uses of JavaScript:
    1. Pop-up messages
    2. Image sliders on a website
    3. Changing website content without reloading the page (AJAX)
    4. Form validation before submission

- JavaScript is easy to learn and works in almost every web browser.

Example:
When you click a button that changes the text on a webpage without refreshing the page, that is JavaScript at work.

1.9.2 PHP

- **PHP** stands for **Hypertext Preprocessor**. It is a **server-side scripting language**, meaning it runs on the web server, not on the user's computer.
- PHP is mostly used for **dynamic websites and web applications**.
- It can communicate with **databases** to retrieve, store, and update information.
- Example uses of PHP:
    1. Login and registration forms
    2. Online shopping carts
    3. Content management systems (like WordPress)
    4. Displaying personalized content for users

In short:

- JavaScript → runs in the **browser**, makes web pages interactive
- PHP → runs on the **server**, makes web pages dynamic and connected to databases

1.10 Web Hosting

A **web hosting service** is a service that allows individuals, organizations, and businesses to make their websites accessible on the Internet. Simply put, web hosting provides **space on a web server** where your website files, images, and content are stored. Without web hosting, a website cannot be accessed by users online.

When a user types your website address in a browser, the request is sent to the **web server** where your website is hosted, and the server sends the requested web pages to the user.

1.10.1 Web Hosting Services

Web hosting services provide storage space and technologies to make websites available online. Some common web hosting services include:

1. **Shared Hosting** – Multiple websites are hosted on a single server. It is **cheap** and easy for beginners, but performance may slow down if other websites use too many resources.

2. **Dedicated Hosting** – A website gets a **whole server dedicated** to it. It is **faster and more secure**, suitable for large businesses.

3. **Cloud Hosting** – Websites are hosted on **multiple connected servers**. It is highly **scalable and reliable**, as downtime is minimal.

4. **VPS Hosting (Virtual Private Server)** – A physical server is divided into multiple virtual servers. Each website has its **own dedicated resources**, making it more stable than shared hosting.

5. **Managed Hosting** – The hosting provider manages the server, security, and updates, allowing the website owner to focus on content.

6. **Colocation Hosting** – The website owner places their own server in a hosting provider's data center. It provides maximum control but is **expensive**.

1.10.2 Types of Web Hosting

Type	Description	Best for
Shared Hosting	Multiple websites share the same server	Small websites, personal blogs
Dedicated Hosting	Entire server dedicated to one website	Large businesses, high-traffic sites

Cloud Hosting	Website hosted on multiple servers	Growing businesses, scalable projects
VPS Hosting	Virtual private server with dedicated resources	Medium-sized businesses, online stores
Managed Hosting	Hosting provider handles server management	Beginners, people without technical knowledge
Colocation Hosting	Owner provides server, data center provides space	Large enterprises with IT teams

In short: Web hosting is like **renting a space for your website** on the Internet. The type of hosting you choose depends on your website size, traffic, and technical requirements.

1.11 Cookies

A **cookie** is a small piece of data sent from a website and stored on a user's computer by their web browser. Cookies help websites **remember information about the user**, such as login details, preferences, or shopping cart items.

Cookies are important for improving the **user experience** and personalizing website content.

1.11.1 Types of Cookies

1. **Session Cookies** –
    - Temporary cookies that are deleted when the browser is closed.
    - Example: Keeping a user logged in while browsing a website.

2. **Persistent Cookies** –
    - Stored on the computer for a fixed period even after the browser is closed.
    - Example: Remembering login details or site preferences for the next visit.
3. **First-Party Cookies** –
    - Created and stored by the website you are visiting.
    - Example: A website saving your language preference.
4. **Third-Party Cookies** –
    - Created by other websites, often for tracking and advertising purposes.
    - Example: Ads that follow you on different websites.

1.11.2 Uses of Cookies

- Remembering login details and preferences.
- Saving items in shopping carts on e-commerce websites.
- Tracking user behavior for analytics.
- Personalizing advertisements or content.

1.11.3 Browser Settings for Cookies

- Users can **enable or disable cookies** in their web browser settings.
- Browsers allow:
    - Deleting cookies after each session
    - Blocking third-party cookies

- Managing exceptions for specific websites

1.11.4 Privacy Concerns about Cookies

- Cookies can track user activity across websites, raising **privacy issues**.
- Third-party cookies may collect personal information for **advertising purposes**.
- Users should be cautious while enabling cookies and regularly **clear cookies** to protect privacy.

1.12 Web 2.0

- **Web 2.0** refers to the **second generation of the web**, which focuses on **user interaction, collaboration, and sharing**.
- Unlike the static Web 1.0, Web 2.0 allows users to **create content** and communicate easily.
- Examples: Social media platforms (Facebook, Twitter), blogs, wikis, and video sharing sites (YouTube).
- Key features:
  - Social networking
  - User-generated content
  - Collaborative tools
  - Rich media

1.13 Web 3.0

- **Web 3.0** is the **next generation of the web**, often called the **Semantic Web**.
- It focuses on **intelligent, personalized, and decentralized** web experiences.
- Uses **AI (Artificial Intelligence), blockchain, and smart contracts** to provide smarter services.
- Key features:
    - Personalized content based on user preferences
    - Decentralized applications (dApps)
    - Enhanced data privacy and security
    - Integration with IoT (Internet of Things) devices

Example: Web 3.0 applications could suggest a product you need before you search for it, or allow secure financial transactions without intermediaries using blockchain.

✅ **Summary:**

- **Cookies** help websites remember user information but have privacy concerns.
- **Web 2.0** focuses on user interaction and collaboration.
- **Web 3.0** is intelligent, decentralized, and personalized for a better web experience.

Chapter 2: HTML5 Introduction

HTML (Hypertext Markup Language) is the standard language used to create and design **web pages**. It provides the structure of a webpage, which browsers display as content including text, images, links, and multimedia. **HTML5** is the latest version of HTML and includes advanced features for multimedia, graphics, and interactive web applications.

2.1 HTML Editors

An **HTML editor** is a software tool used to write HTML code. There are two types of editors:

1. **Text-Based Editors** – Simple editors where you write code manually.
    - Examples: **Notepad, Notepad++, Sublime Text**
2. **WYSIWYG Editors** – (What You See Is What You Get) allow you to design web pages visually without coding much.
    - Examples: **Adobe Dreamweaver, Microsoft Expression Web**

Tip: Beginners can start with text-based editors to learn the basics of coding.

2.2 HTML Basic

HTML documents have a standard structure:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <!-- Content goes here -->
  </body>
</html>
```

- `<!DOCTYPE html>` → Declares the document type as HTML5.
- `<html>` → Root element of the page.
- `<head>` → Contains metadata like page title and links to stylesheets.
- `<body>` → Contains visible content like text, images, and links.

2.3 HTML Elements

- HTML **elements** are building blocks of a webpage.
- Example: `<p>This is a paragraph.</p>`
    - `<p>` → Opening tag
    - `</p>` → Closing tag
    - `This is a paragraph.` → Content

2.4 HTML Attributes

- **Attributes** provide additional information about an element.
- Example: `<a href="https://www.google.com">Google</a>`
    - `href` is an attribute specifying the link URL.

2.5 HTML Headings

- Headings are used to define titles and subtitles on a webpage.

- HTML has six levels: `<h1>` to `<h6>`
    - `<h1>` → Largest heading
    - `<h6>` → Smallest heading

Example:

```
<h1>Main Title</h1>
<h2>Sub Title</h2>
```

2.6 HTML Paragraphs

- Paragraphs are defined using the `<p>` tag.
- Browsers add space before and after paragraphs automatically.

Example:

```
<p>This is a paragraph of text.</p>
```

2.7 HTML Styles

- HTML uses the **style attribute** or CSS to change the appearance of text and elements.
- Example:

```
<p style="color:blue; font-size:16px;">Styled paragraph</p>
```

2.8 HTML Formatting

- HTML provides tags to format text:

- `<b>` → Bold
- `<i>` → Italic
- `<u>` → Underline
- `<mark>` → Highlight text

Example:

```
<p>This is <b>bold</b> and <i>italic</i> text.</p>
```

2.9 HTML Quotations

- To display quotes, use:
  - `<q>` → Short inline quotation
  - `<blockquote>` → Long block quotation

Example:

```
<q>Education is the key to success.</q>
<blockquote>This is a longer quote displayed as a block.</blockquote>
```

2.10 HTML Comments

- Comments are notes in HTML code **not visible in the browser**.
- Syntax: `<!-- This is a comment -->`

Use: Explaining code or leaving reminders for developers.

2.11 HTML Colors

- HTML allows you to set colors using **color names**, **HEX codes**, or **RGB values**.
- Example:

```
<p style="color:red;">Red text</p>
<p style="color:#00FF00;">Green text using HEX</p>
<p style="color:rgb(0,0,255);">Blue text using RGB</p>
```

2.12 HTML CSS

- **CSS (Cascading Style Sheets)** is used to **style HTML elements**.
- CSS allows you to change colors, fonts, spacing, borders, and layout.
- Three ways to use CSS in HTML:

1. **Inline CSS** – Applied directly in the element using the `style` attribute:

```
<p style="color:blue; font-size:18px;">Styled paragraph</p>
```

2. **Internal CSS** – Written inside `<style>` tags in the `<head>` section:

```
<head>
  <style>
    p { color:red; font-size:16px; }
  </style>
</head>
```

3. **External CSS** – Stored in a separate `.css` file and linked using `<link>`:

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

Benefits of CSS:

- Consistent styling across multiple pages
- Easy to update design without changing HTML
- Makes websites visually attractive

2.13 HTML Links

- Links allow users to **navigate from one webpage to another**.
- The `<a>` tag is used to create links.

Example:

```
<a href="https://www.google.com">Visit Google</a>
```

Attributes of Links:

- `href` → URL of the page
- `target` → Specifies where to open the link (`_blank` for new tab)
- `title` → Tooltip when hovering over the link

2.14 HTML Images

- Images make websites more engaging and informative.
- The `<img>` tag is used to display images.

Example:

```
<img src="image.jpg" alt="My Image" width="200" height="150">
```

Attributes:

- `src` → Image file path
- `alt` → Alternative text if image does not load
- `width` and `height` → Size of the image

2.15 HTML Favicon

- A **favicon** is a small icon displayed in the browser tab of a website.
- Helps users identify your website quickly.

Example:

```
<link rel="icon" href="favicon.ico" type="image/x-icon">
```

Tips:

- Usually 16x16 or 32x32 pixels
- Can be `.ico`, `.png`, or `.gif` format

2.16 HTML Tables

- Tables are used to **organize data in rows and columns**.
- The `<table>` tag creates a table, with `<tr>` for rows, `<th>` for headers, and `<td>` for data cells.

Example:

```
<table border="1">
  <tr>
    <th>Name</th>
    <th>Age</th>
    <th>City</th>
  </tr>
  <tr>
    <td>Alice</td>
    <td>20</td>
    <td>London</td>
  </tr>
  <tr>
    <td>Bob</td>
    <td>25</td>
    <td>Paris</td>
  </tr>
</table>
```

Attributes of Tables:

- `border` → Adds border around table
- `cellpadding` → Space inside cells
- `cellspacing` → Space between cells
- `width` and `height` → Size of table

Tips:

- Use tables for **data presentation**, not layout design
- Combine with CSS for better styling

2.17 HTML Lists

HTML provides ways to organize information in **lists**. There are three main types:

1. **Ordered List (`<ol>`)** – A numbered list:

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

2. **Unordered List (`<ul>`)** – A bulleted list:

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Orange</li>
</ul>
```

3. **Definition List (`<dl>`)** – A list of terms and definitions:

```
<dl>
  <dt>HTML</dt>
  <dd>Hypertext Markup Language</dd>
  <dt>CSS</dt>
  <dd>Cascading Style Sheets</dd>
</dl>
```

Tip: Lists make content easy to read and organize.

2.18 HTML Block & Inline Elements

- **Block Elements** – Occupy the **full width** of the page and start on a new line.
  Examples: `<div>`, `<p>`, `<h1>`, `<table>`

- **Inline Elements** – Occupy **only the space needed**, staying on the same line.
  Examples: `<span>`, `<a>`, `<b>`, `<i>`

Tip: Use block elements for large sections and inline elements for small text formatting.

2.19 HTML Classes

- A **class** is used to **group HTML elements** for styling or scripting.
- Multiple elements can share the same class.

Example:

```
<p class="highlight">This is a highlighted paragraph.</p>
<p class="highlight">Another paragraph with same style.</p>
```

CSS Example:

```
.highlight {
  color: red;
  font-weight: bold;
}
```

2.20 HTML Id

- An **id** uniquely identifies an HTML element.
- Each id must be **unique** on a page.

Example:

```
<p id="intro">Welcome to my website!</p>
```

CSS Example:

```
#intro {
  color: blue;
  font-size: 18px;
}
```

Tip: Ids are commonly used for **navigation links** and JavaScript manipulation.

2.21 HTML Iframes

- **Iframe** allows embedding another webpage inside your webpage.
- Tag: `<iframe>`

Example:

```
<iframe src="https://www.example.com" width="600"
height="400"></iframe>
```

Attributes:

- `src` → URL of the page to embed
- `width` and `height` → Size of iframe
- `frameborder` → Border of iframe (0 = no border)

2.22 HTML Head

- The `<head>` section contains **meta-information** about the webpage.
- Common elements inside `<head>`:
    - `<title>` → Page title

- `<meta>` → Metadata like keywords, description, charset
- `<link>` → Link to CSS file
- `<script>` → JavaScript code
- `<style>` → Internal CSS

Tip: Content in `<head>` is **not visible on the webpage**, but important for SEO and functionality.

2.23 HTML Layout

- HTML layout organizes content on a webpage. Common layout elements in HTML5:
  - `<header>` → Top section, usually contains logo and navigation
  - `<nav>` → Navigation menu
  - `<main>` → Main content area
  - `<section>` → A section of content
  - `<article>` → Independent content like blog post
  - `<aside>` → Sidebar content
  - `<footer>` → Bottom section with copyright or contact info

Tip: Use **semantic HTML5 tags** for better readability and SEO.

2.24 HTML Forms

- Forms allow users to **submit data** to a server.
- The `<form>` element contains input fields, buttons, and other controls.

Example:

```
<form action="submit.php" method="post">
  Name: <input type="text" name="username"><br>
  Email: <input type="email" name="email"><br>
  Password: <input type="password" name="password"><br>
  <input type="submit" value="Submit">
</form>
```

Common Input Types:

- `text` → Single line text
- `email` → Email input
- `password` → Password input
- `radio` → Radio buttons
- `checkbox` → Multiple choices
- `submit` → Submit button

Tip: Forms are the backbone of interactive websites like login pages, surveys, and online shopping.

Chapter 3: CSS Introduction

CSS (Cascading Style Sheets) is used to **style HTML elements**. It controls the layout, colors, fonts, spacing, and appearance of web pages. CSS helps separate content (HTML) from design (CSS), making websites easier to manage.

3.1 CSS Syntax

CSS consists of **selectors** and **declarations**.

Example:

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

- `p` → Selector (targets the `<p>` element)
- `{}` → Declaration block
- `color: blue;` → Property and value
- `font-size: 16px;` → Property and value

3.2 CSS Selectors

Selectors specify **which HTML elements** to style:

Element Selector: Targets all elements of a type

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

1.

Class Selector: Targets elements with a specific class

```
.highlight { background-color: yellow; }
```

2.

ID Selector: Targets a unique element by id

```
#intro { font-size: 18px; }
```

3.

Universal Selector: Targets all elements

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

4.

3.3 CSS How To Use

Three ways to apply CSS:

1. **Inline CSS** – Inside the element using `style` attribute
2. **Internal CSS** – Inside `<style>` tags in `<head>`
3. **External CSS** – Separate `.css` file linked using `<link>`

3.4 CSS Comments

- Comments are used to explain code.
- Syntax:

```
/* This is a CSS comment */
```

3.5 CSS Colors

Colors can be set using:

- **Color names:** `red`, `blue`
- **HEX values:** `#FF0000`
- **RGB values:** `rgb(255,0,0)`
- **RGBA values:** `rgba(255,0,0,0.5)` → With transparency

3.6 CSS Backgrounds

- Background can be **color or image**.

Example:

```
body {
  background-color: lightblue;
  background-image: url('image.jpg');
  background-repeat: no-repeat;
  background-size: cover;
}
```

3.7 CSS Borders

- Borders define the edges of an element.

Example:

```
p {
  border: 2px solid black;
  border-radius: 5px; /* Rounded corners */
```

```
}
```

Properties: `border-width`, `border-style`, `border-color`, `border-radius`

3.8 CSS Margins

- Margins create **space outside an element**.

Example:

```
p {
  margin: 20px; /* Space outside */
}
```

- `margin-top`, `margin-bottom`, `margin-left`, `margin-right`

3.9 CSS Padding

- Padding creates **space inside an element**, between content and border.

Example:

```
p {
  padding: 10px;
}
```

3.10 CSS Height/Width

- Set **size of elements** using:

```
div {
  width: 200px;
  height: 100px;
}
```

- Can also use **percentage values** for responsive design

3.11 CSS Box Model

- Every element is a **rectangular box**. Box model includes:

1. **Content** – Text, image, or data inside the element
2. **Padding** – Space between content and border
3. **Border** – Surrounds padding and content
4. **Margin** – Space outside the border

Example diagram:

```
Margin
  Border
    Padding
      Content
```

3.12 CSS Outline

- Outline is a **line drawn outside the element border**.
- Does not affect element size.

Example:

```
p {
```

```
  outline: 2px dashed red;
}
```

3.13 CSS Text

- Style text using properties:
    - `text-align` → left, right, center, justify
    - `text-decoration` → underline, overline, line-through, none
    - `text-transform` → uppercase, lowercase, capitalize
    - `letter-spacing` → space between letters

3.14 CSS Fonts

- Control font appearance with:

```
p {
  font-family: Arial, sans-serif;
  font-size: 16px;
  font-style: italic;
  font-weight: bold;
}
```

- `font-family` → Sets typeface
- `font-size` → Size of text
- `font-style` → Normal, italic, oblique
- `font-weight` → Normal, bold, lighter, bolder

3.15 CSS Icons

- Icons are **small images or symbols** used for buttons, menus, and navigation.
- Can be added using **Font Awesome**, **Material Icons**, or images.

Example with Font Awesome:

```
<i class="fa fa-home"></i> Home
```

Tip: Use icons to make websites visually appealing and easier to navigate.

3.16 CSS Links

- Links can be styled with CSS using pseudo-classes:
    - `a:link` → Normal unvisited link
    - `a:visited` → Visited link
    - `a:hover` → When mouse hovers
    - `a:active` → When clicked

Example:

```
a:link { color: blue; }

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

3.17 CSS Lists

- Style HTML lists with CSS:

Example:

```
ul {
  list-style-type: square; /* circle, disc, none */
  margin-left: 20px;
}
```

Tip: Use CSS to remove bullets, change colors, or create horizontal menus.

3.18 CSS Tables

- Style tables using CSS:

Example:

```
table {
  border-collapse: collapse;
  width: 80%;
}
th, td {
  border: 1px solid black;
  padding: 8px;
  text-align: center;
}
th {
```

```
  background-color: #f2f2f2;
}
```

3.19 CSS Display

- Controls how elements are **rendered**.
- Common values:
    - `block` → Starts on a new line
    - `inline` → Does not start on a new line
    - `inline-block` → Behaves like inline but allows height/width
    - `none` → Hides the element

3.20 CSS Max-width

- Sets the **maximum width** an element can occupy:

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

- Important for **responsive design**

3.21 CSS Position

- Controls element **placement** on a page:
    - `static` → Default, normal flow
    - `relative` → Relative to its normal position
    - `absolute` → Relative to nearest positioned parent
    - `fixed` → Stays fixed in the viewport
    - `sticky` → Stays within parent until scrolled

3.22 CSS Z-index

- Controls **stacking order** of overlapping elements.
- Higher `z-index` appears on top.

Example:

```
div {
  position: absolute;
  z-index: 10;
}
```

3.23 CSS Overflow

- Controls what happens when content **exceeds the container**:
    - `visible` → Shows overflow (default)
    - `hidden` → Hides overflow
    - `scroll` → Adds scrollbar
    - `auto` → Adds scrollbar only when needed

3.24 CSS Float

- Float an element to the **left** or **right** of its container.

Example:

```
img {
  float: left;
  margin-right: 10px;
}
```

- Commonly used for **image alignment** or **text wrapping**

3.25 CSS Inline-block

- Behaves like **inline element** but can have **height and width**.

Example:

```
div {
```

```
  display: inline-block;
  width: 200px;
  height: 100px;
}
```

3.26 CSS Align

- Align elements using:
    - `text-align` → Align text (left, right, center, justify)
    - `vertical-align` → Align inline elements vertically

3.27 CSS Opacity

- Controls **transparency** of an element:

```
div {
  opacity: 0.5; /* 0 = fully transparent, 1 = fully visible */
}
```

3.28 CSS Navigation Bar

- A navigation bar is a menu to **navigate between pages**.

Example:

```
nav {
  background-color: #333;
  overflow: hidden;
}
nav a {
  float: left;
  display: block;
  color: white;
  padding: 14px 20px;
  text-decoration: none;
}
nav a:hover {
  background-color: #ddd;
  color: black;
}
```

3.29 CSS Dropdowns

- Dropdown menus show options when hovered.

Example:

```
.dropdown:hover .dropdown-content {
```

```
  display: block;
}
```

Tip: Use CSS to style dropdown links, background, and hover effects.

3.30 CSS Attribute Selectors

- Style elements based on their **attributes**:

Examples:

```
a[target="_blank"] { color: red; } /* Links opening in new tab */
input[type="text"] { border: 1px solid black; } /* Text fields */
```

Common Uses:

- Highlight specific inputs
- Style external links differently
- Select elements dynamically based on attributes

Chpater 4: JavaScript

4.1 JavaScript (JS) Introduction

- **JavaScript** is a **high-level, interpreted programming language** used to make web pages **interactive**.
- It runs on the **client-side** (in the browser) but can also run on the **server-side** using Node.js.
- JS allows developers to:
    - Create dynamic content
    - Handle user interactions
    - Validate forms
    - Control multimedia
    - Communicate with web servers

Example:

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

Tip: JS is essential for modern web development alongside HTML and CSS.

4.2 JavaScript Output

- JavaScript can **display output** in different ways:

1. **Alert Box:**

```
alert("Hello World!");
```

2. **Console Output:**

```
console.log("This is a message in console");
```

3. **Writing to HTML Page:**

```
document.write("Hello on Web Page!");
```

Tip: Use `console.log()` for debugging and alerts for user messages.

4.3 JavaScript Statements

- Statements are **instructions** executed by the browser.
- Each statement ends with a **semicolon** `;`.
- Multiple statements can be combined to perform tasks.

Example:

```
var x = 10;
var y = 20;
var sum = x + y;
console.log(sum);
```

4.4 JavaScript Structure

- A basic JS program has these components:

Variables – Store data

```
var name = "Tahir";
```

1.

Operators – Perform arithmetic or logical operations

```
var total = 5 + 10;
```

2.

Expressions – Combination of values and operators

```
var result = total * 2;
```

3.

Functions – Group of statements to perform tasks

```
function greet() {
    alert("Hello Students!");
}
greet();
```

4.

4.5 JavaScript Comments

- Comments are **ignored by the browser** and help explain code.
- Types of comments:

1. **Single-line comment:**

```
// This is a single-line comment
```

2. **Multi-line comment:**

```
/*
   This is a multi-line comment
   You can write multiple lines here
*/
```

Tip: Always comment your code to make it readable and easier to debug.

4.6 JavaScript Variables

- **Variables** are used to **store data values** that can be used later in a program.
- In JavaScript, variables are **containers for data**.

Declaration using `var`:

```
var name = "Tahir";
var age = 20;
console.log(name, age);
```

Rules for variables:

1. Names must start with a **letter, `$`, or `_`**.
2. Cannot start with a number.
3. Cannot use **reserved keywords** like `var`, `if`, `for`.
4. Case-sensitive: `Name` and `name` are different.

4.7 JavaScript Let

- `let` is used to declare **block-scoped variables** (limited to the block `{}` where they are defined).
- Recommended over `var` for modern JavaScript.

Example:

```
let city = "Karachi";
if (true) {
    let city = "Lahore"; // This city is only inside this block
    console.log(city); // Lahore
}
console.log(city); // Karachi
```

Tip: `let` prevents accidental changes to variables outside their block.

4.8 JavaScript Const

- `const` is used to declare **constants**, i.e., variables whose values **cannot change**.
- Must be **initialized** when declared.

Example:

```
const pi = 3.1416;
// pi = 3.14; // ❌ This will cause an error
console.log(pi);
```

Tip: Use `const` for values that should **never change**, like mathematical constants or configuration values.

4.9 JavaScript Operators

Operators are symbols that **perform operations** on values and variables.

Types of Operators:

1. **Arithmetic Operators:**
    - `+` addition, `-` subtraction, `*` multiplication, `/` division, `%` remainder

```
let x = 10 + 5; // 15
```

2.
3. **Assignment Operators:**
    - `=` assign, `+=`, `-=`, `*=`, `/=`

```
let y = 10;
y += 5; // y = 15
```

4.
5. **Comparison Operators:**
    - `==` equal, `===` equal value and type, `!=` not equal, `<`, `>`, `<=`, `>=`

```
console.log(5 === "5"); // false
```

6.
7. **Logical Operators:**
    - `&&` AND, `||` OR, `!` NOT

```
console.log(true && false); // false
console.log(!true); // false
```

8.
9. **Increment/Decrement Operators:**
    - `++` increment, `--` decrement

```
let z = 5;
z++; // 6
z--; // 5
```

10.

✅ **Summary:**

- `var`, `let`, `const` are used to **store data** with different scopes and rules.
- Operators perform **calculations, comparisons, and logical decisions** in JS programs.

4.10 JavaScript Functions

- A **function** is a block of code designed to **perform a particular task**.
- Functions help **reuse code**, organize programs, and make debugging easier.
- Functions are executed **when they are called** in the program.

Syntax:

```
function functionName(parameters) {
    // Code to execute
}
```

Example:

```
function greet() {
    console.log("Hello Students!");
}
greet(); // Call the function
```

4.10.1 User-Defined Functions

- Created by the programmer to perform **specific tasks**.
- Can accept **parameters** and return **values**.

Example with parameters:

```
function addNumbers(a, b) {
    return a + b;
}
let sum = addNumbers(5, 10);
console.log(sum); // 15
```

Key points:

1. Function name should be **meaningful**.
2. Parameters are optional.
3. Functions can **return values** using `return`.

4.10.2 Built-in Functions

JavaScript provides many **predefined functions** for common tasks:

1. **`Math.abs(x)`** → Returns the absolute value of `x`.

```
console.log(Math.abs(-10)); // 10
```

2. **`Math.random()`** → Returns a random number between 0 (inclusive) and 1 (exclusive).

```
console.log(Math.random());
```

3. **`Math.max(a, b, ...)`** → Returns the **largest** value.

```
console.log(Math.max(5, 10, 3)); // 10
```

4. **`Math.min(a, b, ...)`** → Returns the **smallest** value.

```
console.log(Math.min(5, 10, 3)); // 3
```

5. **`eval(expression)`** → Evaluates a string as JavaScript code.

```
console.log(eval("5 + 10")); // 15
```

6. **`parseInt(string)`** → Converts a string to an **integer**.

```
console.log(parseInt("123")); // 123
```

7. **`parseFloat(string)`** → Converts a string to a **floating-point number**.

```
console.log(parseFloat("3.14")); // 3.14
```

Tip: Use built-in functions to **simplify coding tasks** instead of writing your own code for common operations.

✅ **Summary:**

- Functions organize and **reuse code**.
- **User-defined functions** are created by programmers.
- **Built-in functions** save time for **common calculations and conversions**.

4.11 JavaScript Arrays

- **Arrays** are used to store **multiple values in a single variable**.
- Each value in an array is called an **element**, and it has an **index** starting from 0.

Syntax:

```
let fruits = ["Apple", "Banana", "Mango"];
console.log(fruits[0]); // Apple
```

Common Array Methods:

1. `push()` → Add element at the end
2. `pop()` → Remove last element
3. `shift()` → Remove first element
4. `unshift()` → Add element at the beginning
5. `length` → Number of elements in array

Example:

```
fruits.push("Orange");
console.log(fruits.length); // 4
```

4.12 JavaScript If Else

- Used for **decision-making** based on a condition.
- **Syntax:**

```
if (condition) {
    // code if true
} else {
    // code if false
}
```

Example:

```
let age = 18;
if (age >= 18) {
    console.log("You can vote.");
} else {
    console.log("You cannot vote.");
}
```

4.13 JavaScript Switch

- `switch` is used when you have **multiple conditions**.
- Each case is checked, and the matching block is executed.

Syntax:

```
switch(expression) {
    case value1:
        // code
        break;
    case value2:
        // code
```

```
        break;
    default:
        // code if no case matches
}
```

Example:

```
let day = 3;
switch(day) {
    case 1: console.log("Monday"); break;
    case 2: console.log("Tuesday"); break;
    case 3: console.log("Wednesday"); break;
    default: console.log("Invalid Day");
}
```

4.14 JavaScript For Loop

- Executes a block of code **a specific number of times**.

Syntax:

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

Example:

```
for(let i = 1; i <= 5; i++) {
    console.log(i);
}
// Output: 1 2 3 4 5
```

4.15 JavaScript While Loop

- Repeats a block of code **while a condition is true**.

Syntax:

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

Example:

```
let i = 1;
while(i <= 5) {
    console.log(i);
    i++;
}
// Output: 1 2 3 4 5
```

Tip: Use loops to **avoid writing repetitive code**.

✅ **Summary:**

- Arrays store multiple values in **one variable**.
- `if-else` and `switch` are used for **decision-making**.
- `for` and `while` loops help **repeat tasks efficiently**.