Assignment-HTML

Module 1 – Foundation

THEORY EXERCISE:

1. **What is a HTTP?**
    - HTTP (HyperText Transfer Protocol) is a communication protocol used to transfer data between a web browser (client) and a web server over the internet.
    - HTTP is a protocol (set of rules) used to transfer data like web pages, images, videos, etc., between a web browser (client) and a web server.

How it Works:

1. **Client (Browser)** sends a request for a web page (e.g., when you type a URL and press enter).
2. **Server** processes the request and sends back the response (like HTML, CSS, images, etc.).
3. The **browser displays** the received data.

Example:

- When you open https://www.example.com, your browser sends an **HTTP request** to the server.
- The server responds with an **HTTP response** containing the web page data.

HTTP Methods:

- GET: Request data (e.g., view a web page)
- POST: Submit data (e.g., fill out a form)
- PUT: Update data
- DELETE: Remove data

2. **What is a Browsers? How they works?**
    - A **browser** (or **web browser**) is a **software application** used to access, retrieve, and display content from the **World Wide Web** (WWW).
    - Common browsers include **Google Chrome, Mozilla Firefox, Microsoft Edge, Safari**, and **Opera**.

How Does a Browser Work?

Here's a simple step-by-step explanation of how browsers work:

1. User Enters a URL

- You type something like https://www.google.com into the browser's address bar.

2. DNS Lookup

- The browser sends the domain name (google.com) to a **DNS (Domain Name System)** server to get the **IP address** of the server.

3. HTTP/HTTPS Request

- The browser sends an **HTTP or HTTPS request** to the server at that IP address asking for the web page.

4. Server Response

- The web server responds with the necessary files (HTML, CSS, JavaScript, images, etc.).

5. Rendering the Page

- The browser receives the files and:
    - Parses the HTML
    - Applies the CSS for styling
    - Executes JavaScript for interactivity
    - Displays the final page on your screen

6. Displaying Content

- The browser renders (displays) the complete webpage as you see it.

3. What is Domain Name?

- A **domain name** is the **human-readable address** of a website on the internet.
- It is used to identify and access websites easily without needing to remember complex IP addresses.
- Using different Domain name for different countries is beneficial for several reasons.
- It helps with geographical targeting in search results, builds local trust and credibility and can improve search engine optimization(SEO) whiting specific countries.

Why Are Domain Names Important?

Computers use **IP addresses** like 142.250.195.78 to communicate.
But IPs are hard to remember, so we use **domain names** like google.com, which are easier for humans.

Structure of a Domain Name:

pgsql
CopyEdit

```
www.example.com
|   |       |
|   |       └── Second-level domain (example)
|   └────────────── Subdomain (www)
└───────────────── Top-level domain (TLD - .com)
```

Common Top-Level Domains (TLDs):

- .com – Commercial
- .org – Organization
- .net – Network
- .edu – Educational
- .in, .uk, .us – Country-specific

4. What is hosting?

- **Hosting** (or **Web Hosting**) is a **service** that allows individuals or organizations to **store their website files** (HTML, images, videos, etc.) on a **server** that is connected to the **internet**, so people can access the website anytime, anywhere.
- **Hosting is like renting space on the internet** to keep your website online

Module 2 – Fundamentals of World Wide Web

THEORY EXERCISE:

5. **Difference between Web Designer and Web Developer**

Difference Between Web Designer and Web Developer

ASPECT	WEB DESIGNER	WEB DEVELOPER
MAIN ROLE	Focuses on the **look and feel** of a website	Focuses on the **functionality** and structure
KEY SKILLS	UI/UX design, Color theory, Typography, Adobe XD, Figma, Photoshop, HTML, CSS	HTML, CSS, JavaScript, PHP, Python, Databases, APIs
TOOLS USED	Figma, Adobe XD, Sketch, Canva	VS Code, Git, Terminal, Databases
LANGUAGES	HTML, CSS, some JavaScript	HTML, CSS, JavaScript, plus backend languages
FOCUS	User interface (UI), User experience (UX)	Site logic, server, database, interactivity
OUTPUT	Creates wireframes, mockups, page layouts	Builds the website based on design

Simple Example:

- A **Web Designer** designs a beautiful online store layout.
- A **Web Developer** writes code to make the store actually work (shopping cart, login, payment, etc.)

6. **What is a W3C?**

   - **W3C** stands for the **World Wide Web Consortium**.
   - It is the **main international organization** that develops **web standards** to ensure the **long-term growth and compatibility** of the web.

7. **What is Domain?**
   A **domain** is the **unique name** used to identify a website on the internet.

8. **What SEO?**

- **SEO** stands for **Search Engine Optimization**.
- SEO is the process of **improving a website's visibility** on **search engines** like Google, Bing, or Yahoo — to get **more free (organic) traffic**.

Simple Definition:

- SEO helps your website **rank higher** in search results when people search for something related to your content.
- For example:
  If you have a blog about healthy food, SEO helps your site appear on **Google's first page** when someone searches for "healthy recipes".

Key Elements of SEO:

1. **Keywords** – Search terms people use (e.g., “best laptop 2025”)
2. **Title Tags & Meta Descriptions** – Shown in search results
3. **Backlinks** – Links from other sites (builds trust)
4. **Content Quality** – Useful, relevant, and original content
5. **Mobile Optimization** – Works well on phones and tablets
6. **Site Speed** – Fast loading = better user experience
7. **Alt Text** – Descriptions for images (important for SEO & accessibility)

9. What is SDLC life cycle?

- **SDLC** stands for **Software Development Life Cycle**.
- It is a **step-by-step process** used by software developers and project teams to **design, develop, test, and maintain** software applications.

Purpose of SDLC:

- To ensure **high-quality software**
- Delivered **on time**
- Within **budget**
- With **minimum risk**

Phases of SDLC:

[Planning] → [Requirement Analysis] → [Design] →

[Development] → [Testing] → [Deployment] → [Maintenance]

↑--↑

Why SDLC is Important:

- Helps teams **stay organized**
- Reduces **errors and risks**
- Ensures **customer satisfaction**
- Allows for **proper documentation**

Module 2 – Frontend- HTML

HTML Basics

Theory Assignment

Question 1: Define HTML. What is the purpose of HTML in web development?

Definition of HTML:

HTML (HyperText Markup Language) is the standard markup language used to create and structure content on the web. It tells the web browser how to display text, images, links, and other media on a webpage.

Purpose of HTML in Web Development:

1. **Structure the Content:**
    - HTML provides a structural foundation for web pages using elements like <header>, <footer>, <section>, and <article>.
2. **Display Text and Media:**
    - It enables the display of text, images, videos, audio, and other media using tags like <p>, <img>, <video>, etc.
3. **Link Webpages (Hypertext):**
    - HTML allows pages to be interconnected using hyperlinks (<a> tags), forming the basis of website navigation.
4. **Form User Input:**
    - HTML includes form elements like <form>, <input>, <textarea>, and <button> to collect data from users.
5. **Embed Scripts and Styles:**
    - HTML can integrate CSS for styling and JavaScript for behavior, helping build interactive and visually appealing websites.

Question 2: Explain the basic structure of an HTML document. Identify the mandatory tagsand their purposes.

Basic Structure of an HTML Document

An HTML document has a well-defined structure made up of essential tags that organize and define the content of a web page.

Basic HTML Structure:

```
<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>
```

<body>

<h1>Main Heading</h1>

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

</body>

</html>

Mandatory HTML Tags and Their Purposes:

Tag	*Purpose*
<!DOCTYPE html>	Declares the document type and version of HTML (HTML5). Helps browsers render the page correctly.
<html>	Root element of the HTML document. Wraps all the content on the page.
<head>	Contains meta-information about the document (not visible to users), such as title, CSS, charset, etc.
<title>	Sets the title of the webpage (shown in the browser tab).
<body>	Contains all the visible content of the page like text, images, links, etc.

Question 3: What is the difference between block-level elements and inline elements in HTML? Provide examples of each.

1. Block-Level Elements

- **Definition:** Block-level elements take up the full width of the container and start on a **new line**.
- **Purpose:** Used to create larger structures (like paragraphs, sections, dividers).
- **Examples:**
  - <div>
  - <p>
  - <h1> to <h6>
  - <ul>, <ol>, <li>
  - <table>, <section>, <article>

📌 Example:

```
html
<p>This is a paragraph.</p>
<h1>This is a heading</h1>
<div>This is a division block</div>
```

2. Inline Elements

- **Definition:** Inline elements do **not start on a new line**. They only take up as much width as necessary.
- **Purpose:** Used for formatting small pieces of content within block elements.
- **Examples:**
  - <span>
  - <a>
  - <strong>, <em>
  - <img>
  - <label>, <input>

📌 Example:

```
<p>This is a <span style="color: red;">red word</span> inside a paragraph.</p>
<a href="#">Click here</a>
```

Question 4: Discuss the role of semantic HTML. Why is it important for accessibility and SEO? Provide examples of semantic elements.

Semantic HTML uses HTML5 elements that clearly describe their meaning and purpose in the document structure — making the content **more understandable for browsers, developers, assistive technologies, and search engines**.

Why Semantic HTML is Important

1. Accessibility (A11Y):

- **Screen readers** and assistive tools rely on semantic tags to navigate and interpret content correctly.
- Example: <nav> helps a screen reader know it's a navigation menu.

2. SEO (Search Engine Optimization):

- Search engines like Google understand the **context** and **importance** of different sections using semantic tags.
- Helps improve **ranking** and **visibility** in search results.

3. Code Clarity & Maintainability:

- Makes HTML **easier to read** and **maintain** for developers.
- Reduces confusion caused by using generic tags like <div> for everything.

Examples of Semantic HTML Elements:

Tag	Purpose
`<header>`	Represents the top section of a page or section
`<nav>`	Defines navigation links
`<main>`	Specifies the main content of the document
`<section>`	Groups related content into a logical section
`<article>`	Represents a self-contained piece of content
`<aside>`	Holds side content like ads or related links
`<footer>`	Represents the bottom area of a page or section
`<figure>` / `<figcaption>`	Used for images with captions
`<time>`	Represents a specific time or date

Semantic HTML Example:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Semantic HTML Example</title>
 </head>
 <body>
  <header>
   <h1>My Blog</h1>
  </header>

  <nav>
```

```
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Articles</a></li>
      </ul>
    </nav>
    <main>
      <article>
        <h2>HTML for Beginners</h2>
        <p>Learn the basics of HTML in this article.</p>
      </article>
    </main>

    <footer>
      <p>© 2025 My Blog</p>
    </footer>
  </body>
</html>
```

HTML Forms

Theory Assignment

Question 1: What are HTML forms used for? Describe the purpose of the input, textarea, select, and button elements.

<u>What Are HTML Forms Used For?</u>

HTML forms are used to **collect user input** on a webpage and send it to a server for processing.

Forms are essential for tasks like:

- Logging in
- Registering users
- Searching
- Submitting feedback or surveys
- Uploading files

<u>Key Form Elements and Their Purposes:</u>

1. 🔘 <input>

- Used to collect **single-line** data (text, email, password, checkbox, radio, file, etc.).
- Type is defined using the type attribute.

Example:

```
<input type="text" name="username" placeholder="Enter your name">

<input type="email" name="email" placeholder="Enter your email">

<input type="checkbox" name="subscribe"> Subscribe
```

---

2. 📝 <textarea>

- Used for **multi-line** text input such as comments, messages, or feedback.

Example:

```
<textarea name="message" rows="4" cols="30" placeholder="Write your message..."></textarea>
```

---

3. ▼ <select> (with <option>)

- Creates a **dropdown list** for selecting one or more options.

Example:

```
<select name="country">
  <option value="india">India</option>
  <option value="usa">USA</option>
  <option value="uk">UK</option>
</select>
```

4. 🔘 <button>

- Creates a clickable **button**, often used to submit the form or trigger JavaScript actions.

Example:

```
<button type="submit">Submit</button>
<button type="reset">Reset</button>
```

Complete Form Example:

```
<form action="/submit" method="post">
  <label>Name:</label>
  <input type="text" name="name"><br>

  <label>Message:</label>
  <textarea name="message"></textarea><br>

  <label>Country:</label>
  <select name="country">
    <option value="india">India</option>
    <option value="canada">Canada</option>
  </select><br>

  <button type="submit">Send</button>
</form>
```

Question 2: Explain the difference between the GET and POST methods in form submission. When should each be used?

1. GET Method

- **Sends data as URL parameters** (visible in the address bar).
- Data is **appended** to the URL after a ?.
- **Example URL:**
  example.com/search?query=html&sort=recent

✅ Use GET when:

- You are **retrieving data** (searches, filters).
- The form data is **not sensitive**.
- You want to **bookmark or share the URL** with data.

⚠ Avoid GET when:

- Sending **passwords, personal, or sensitive information**.
- Submitting **large amounts of data** (limited by URL length).

2. POST Method

- **Sends data in the body** of the HTTP request (not visible in URL).
- **More secure** than GET for sending sensitive data.
- No length limitations.

✅ Use POST when:

- You are **modifying data** on the server (registration, login, form submission).
- The data is **confidential or large** in size.
- You don't want data to appear in the URL.

Question 3: What is the purpose of the label element in a form, and how does it improve accessibility?

The <label> element is used to **define a text description for a form input**, such as a textbox, checkbox, or radio button.

Main Purposes of <label>:

1. **Describes the input field:**
    - Helps users understand what to enter in the field.
2. **Connects text to input fields:**
    - Clicking on the label sets focus to the input automatically.
3. **Improves user experience and usability.**

HTML Tables

Theory Assignment

Question 1: Explain the structure of an HTML table and the purpose of each of the following elements: <table>, <tr>, <th>, <td>, and <thead>.

An **HTML table** is used to display data in a structured format using rows and columns. Here's a breakdown of its structure and the purpose of each key element:

Basic Structure of an HTML Table:

```
<table>
  <thead>
    <tr>
      <th>Heading 1</th>
      <th>Heading 2</th>
    </tr>
  </thead>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
    </table>
```

Explanation of Each Element:

ELEMENT	PURPOSE
<TABLE>	**Container** element that defines the table and holds all table content.
<TR>	**Table Row** – groups a row of cells (can include <th> or <td>).
<TH>	**Table Header Cell** – used inside a <tr> to define a column header. Displayed in **bold** and **centered** by default.
<TD>	**Table Data Cell** – contains regular data in a row. Appears under the corresponding header.
<THEAD>	Groups the header content (typically the first row). Helps with **semantic structure** and styling, and is useful when printing or scrolling tables.

Question 2: What is the difference between `colspan` and `rowspan` in tables? Provide examples.

The `colspan` and `rowspan` attributes in HTML tables are used to **merge cells** across **columns** and **rows**, respectively.

Difference Between colspan and rowspan:

ATTRIBUTE	PURPOSE	EFFECT
COLSPAN	Merges cells **across columns**	A cell spans multiple columns in a row
ROWSPAN	Merges cells **across rows**	A cell spans multiple rows in a column

1.**colspan** Example**:**

Merges **2 columns** in a row (horizontal merge):

```
<table border="1">
  <tr>
    <th colspan="2">Name & Score</th>
  </tr>
  <tr>
    <td>Alice</td>
    <td>95</td>
  </tr>
</table>
```

Output: The first row has a single cell spanning two columns with the heading "Name & Score".

2. **rowspan** Example:

Merges **2 rows** in a column (vertical merge):

```
<table border="1">
  <tr>
    <th rowspan="2">Name</th>
    <td>Math</td>
  </tr>
  <tr>
    <td>Science</td>
  </tr>
</table>
```

Output: The "Name" cell spans two rows vertically, next to "Math" and "Science".

Question 3: Why should tables be used sparingly for layout purposes? What is a better alternative?

Using **tables for layout purposes** (like positioning elements on a webpage) is **discouraged** in modern web development. Here's why—and what you should use instead:

Why Tables Should Be Used Sparingly for Layout:

REASON	EXPLANATION
NOT SEMANTIC	Tables are meant for displaying **tabular data**, not for layout. Misusing them confuses screen readers and assistive technologies.
HARD TO MAINTAIN	Table-based layouts are **complex, bulky, and harder to edit or style** than modern layout techniques.
POOR ACCESSIBILITY	Screen readers and other accessibility tools rely on **semantic HTML**. Layout tables break this structure.

LESS RESPONSIVE	Tables are **rigid** and don’t adapt well to different screen sizes (e.g. phones, tablets).
SLOWER RENDERING	Browsers may take longer to render large or nested tables than simpler layouts using CSS.

Better Alternative: CSS with **<div>**, Flexbox, and Grid

1.Flexbox – Good for 1-dimensional layouts (row or column):

```
<div style="display: flex;">

  <div style="flex: 1;">Left</div>

  <div style="flex: 2;">Right</div>

</div>
```

2. CSS Grid **– Best for 2-dimensional layouts (rows *and* columns):**

```
<div style="display: grid; grid-template-columns: 1fr 2fr;">

  <div>Left</div>

  <div>Right</div>

</div>
```