Module 1: Introduction to Web Development

1. What is Web Development?

Web development refers to the process of building, creating, and maintaining websites or web applications. It encompasses a wide range of activities, from designing the website's layout and interface to writing code, managing server infrastructure, and optimizing the user experience.

---

2. What is a website? And explain the differences between static and dynamic Websites.

A website is a collection of interconnected web pages that are hosted on a server and accessible through the internet. Websites are designed to provide information, services, or functionality to users via a web browser, such as Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.

Static Websites:
● Content is fixed and does not change dynamicall
● Examples: Portfolio sites, simple informational websites(Wikipedia).

Dynamic Websites:
● Content changes based on user interactions or data from the server.
● Examples: Social media platforms, e-commerce sites.

---

3. Explain the differences of web design, front-end and back-end development.

1. **Web Design**: Focuses on the aesthetics and usability of a website.

2. **Frontend Development (Client-side)**:
● Focuses on the user interface (UI) and user experience (UX).
● Involves coding the visual elements of a website that users interact with directly.
● Common technologies:
○ HTML (HyperText Markup Language): Structures the content of a webpage.
○ CSS (Cascading Style Sheets): Styles the layout, colors, and fonts.
○ JavaScript: Adds interactivity and dynamic behavior to web pages.
● Example tools: React, Angular, Vue.js.

3. **Backend Development (Server-side**):
● Handles the behind-the-scenes functionality, such as data processing, business logic, and database management.
● Involves building and managing the server, application, and database.
● Common technologies:
○ Server-side languages: Python, Ruby, PHP, Java, Node.js.
○ Databases: MySQL, PostgreSQL, MongoDB.
● Example tools: Django, Flask, Express.js, Laravel.

Module 2: HTML

1. What does HTML stand for? And Explain what HTML is and why we use it?

HTML stands for HyperText Markup Language.
HTML is the standard markup language used to create and structure content on the web. It is the foundation of every webpage and serves as the skeleton for displaying text, images, links, videos, and other multimedia on the internet.

● HyperText: Refers to text that contains links (hyperlinks) to other documents or resources.
● Markup Language: A system of tags that define the structure and layout web content.

HTML works by using a system of elements (tags) to organize and format information.
We use HTML forStructuring Web Pages: HTML organizes content into headings, paragraphs, lists,
tables, etc., ensuring a logical layout and readability.
Embedding Multimedia: HTML allows the inclusion of images, videos, audio, and other media into web pages.
Hyperlinking: Hyperlinks enable navigation between different web pages or resources.
Browser Interpretation: Web browsers (like Chrome, Firefox, and Safari) interpret HTML to render the content of a website visually for users.

Foundation for Web Development: HTML serves as the base layer for websites, often combined with:
● CSS: To style the appearance of elements.
● JavaScript: To add interactivity and dynamic features.
Accessibility: HTML, when used correctly, ensures content is accessible to all users, including those using assistive technologies like screen readers.

---

2. Explain the HTML structure. E.g. what <!DOCTYPE html> is and how many parts in HTML structure and so on.

HTML documents follow a standardized structure that organizes content and metadata. This structure ensures that browsers can correctly interpret and render the webpage.

Breakdown:
- <!DOCTYPE html>: Declares the document type and version of HTML being used. It tells the web browser that the document adheres to the HTML5 standard. Always at the very beginning of the HTML document.
- <html>: The root element of the page and
- attribute lang: Specifies the language of the document (e.g., lang="en" f English).
- <head>: Contains metadata like the page title.
- <body>: Contains the content visible to users.
- <h1>: A large heading.
- <p>: A paragraph.
- <a>: A hyperlink.

---

3. What are the differences between HTML tags and HTML elements?

A **tag** is the syntax used in HTML to mark the beginning and/or end of an HTML element.
Tags are enclosed in angle brackets (< >).

Types of Tags:
- **Opening Tag**: Marks the start of an element. Example: <p>
- **Closing Tag**: Marks the end of an element, preceded by a slash (/). Example: </p>

An element consists of an opening tag, content, and a closing tag (if required).

An element is the complete structure that defines the functionality or content on t webpage.
Example: <h1>Welcome to My Website</h1>
Though the terms HTML tags and HTML elements are often used interchangeably, they have distinct meanings in the context of HTML.

4. Describe at least 10 HTML tags and explain their definition.

1. <html>
● Definition: The root element of an HTML document. It contains all t
other elements and represents the entire webpage.
2. <head>
● Definition: Contains metadata and links to resources, such as C
stylesheets and JavaScript files. This section is not displayed direct
on the webpage.
3. <body>
● Definition: Contains the content of the webpage that is visible to user
such as text, images, and interactive elements.
4. <h1> to <h6>
● Definition: Represent headings, with <h1> being the largest and mo
important, and <h6> being the smallest.
5. <p>
● Definition: Defines a paragraph of t
6. <a>
● Definition: Defines a hyperlink, used to navigate to another pag
resource.
● Attributes:
○ href: Specifies the URL of the lin
7. <img>
● Definition: Embeds an image in the webpag
● Attributes:
○ src: Specifies the image source (URL or file pa
○ alt: Provides alternative text for accessibility.
8. <ul> and <li>
● Definition: <ul> creates an unordered (bulleted) list, and <l
represents an individual list item.
9. <table>, <tr>, <td>
● Definition: Used to create table
○ <table>: Defines the tabl
○ <tr>: Defines a row in the tabl
○ <td>: Defines a cell in a ro
10. <form>
● Definition: Defines a form for user in
● Attributes:
○ action: Specifies the URL to send form dat
○ method: Specifies the HTTP method (GET or POS)

10 types of HTML tags

1. Structural Tags:
● Define the basic structure of an HTML documen
● Eg. <html> , <head>, <body>
2. Headings and Text Formatting Tags:
● Used to structure and format text.
● Eg. <h1> to <h6>, <p>, <b>, <i>
3. Linking and Navigation Tags:
● Used to create links and navigation menus.
● Eg. <nav>, <a>
4. Media Tags:
● Used to embed images, audio, and video.
● Eg. <img>, <audio>, <video>
5. List Tags:
● Used to create ordered, unordered, or description lists.
● Eg. <ul>, <ol>, <li>, <dl>
6. Table Tags:
● Used to create tables.
● Eg. <table>, <tr>, <td>, <th>
7. Form Tags:
● Used to collect input from users.
● Eg. <form>, <input>, <textarea>, <option>, <button>, <label>, <select>
8. Semantic Tags (HTML5):
● Introduce meaning to the structure of a webpage.
● Eg. <header>, <footer>, <article>, <section>, <aside>, <main>
9. Scripting Tags:
● Enable dynamic behavior on a webpage.
● Eg. <script>, <noscript>
10. Self-Closing Tags:
● Do not require a closing tag.
● Eg. <img>, <br>, <hr>, <meta>, <input>

---

5. What is HTML Attribute?

An HTML attribute provides additional information about an HTML element, enhancing its functionality and controlling its behavior or appearance. Attributes are
always included within the opening tag of an element.

Common HTML attributes are:
1. **Id** : Uniquely identifies an element on the pag
2. **class** : Groups elements for styling or scripting.
3. **style** : Adds inline CSS to an element.

4. **href** : Specifies the URL for a hyperlink
5. **src** : Specifies the source file for an embedded resource, like an imag video.
6. **alt** : Provides alternative text for an image, improving accessibility.
7. **title** : Adds a tooltip that appears when the user hovers over the element.
8. **disabled** : Disables an input element or button.
9. **name** : Identifies form elements for backend processing
10. **value** : Specifies the initial value of form elements like input

---

6. What is Comment and why do we use it?

Comments are notes added to the code for documentation purposes. They are ignored by the compiler or interpreter, which means they do not affect the execution of the code. Comments are a crucial part of programming and serve several important functions.
Eg. <!-- --> in html, /* */ in css, # in python

We use Comment for -
Documentation: Comments explain what the code does, making it easier for others (or yourself at a later time) to understand the logic and purpose behind it.
Clarity: Complex sections of code can be clarified with comments, breaking down complicated logic into understandable parts.
Debugging: Comments can be used to temporarily disable parts of code during debugging without deleting it.
Maintenance: Comments help in maintaining and updating code by providing insights into why certain decisions were made.
Collaboration: When working in teams, comments are essential for effective communication and collaboration.

---

7. Explain the usage of _self and _blank target attributes.

The `target` attribute in HTML is used to specify where to open the linked document when a hyperlink (`<a>`) or form submission is activated.

Usage: `target="_self", default value of target`
Behavior: Opens the linked document in the same frame or tab as it was clicked.

Usage: `target="_blank"`
Behavior: Opens the linked document in the new frame or tab as it was clicked.

---

8. Give a brief explanation of the HTML table.

An HTML table is a structured way to organize and display data in rows and columns, much like a spreadsheet. Tables are commonly used for displaying tabular data such as charts, schedules, and comparison lists. Tables can also be used to get neat form structure along with <form> tag.

`<table>`: Defines the table.
`<caption>`: Provides a title or caption for the table.
`<th>`: Defines a header cell within a row.
`<tr>`: Defines a row in the table.
`<td>`: Defines a data cell within a row.

`<thead>`: Groups the header content.
`<tbody>`: Groups the body content.
`<tfoot>`: Groups the footer content.

---

9. How many list elements in HTML and what are they?

HTML provides three types of list elements that yo can use to display lists of information.

1. **Ordered List (`<ol>`)**:
- An ordered list creates a numbered list.
- Each item in the list is wrapped in a `<li>` (list item) tag. (eg. <ol><li>..</li></ol>)
2. **Unordered List (`<ul>`)**:
- An unordered list creates a bulleted list.
- Display style can be changed using attribute style(eg. style="list-style-type : square;")
- Each item in the list is wrapped in a `<li>` (list item) tag. (eg. <ul><li>..</li></ul>)
3. **Description List (`<dl>`)**:
- A description list is used for definitions or descriptions.
- It includes `<dt>` (description term) for the term being described and `<dd>` (description detail) for the description. (eg. <dl><dt></dt><dd>..</dd></dl>)

---

10. Explain the differences between inline and block elements?

Block Elements

- **Definition**: Block elements take up the full width available (by default), starting on a new line.
- **Characteristics**:

    - They always start on a new line.
    - They occupy the full width of their parent container (unless a specific width is set).
    - They can contain other block elements and inline elements.
- **Common Block Elements**: `<div>`, `<p>`, `<h1>`, `<ul>`, `<li>`, `<section>`, `<article>`, `<footer>`, `<header>`

Inline Elements

- **Definition**: Inline elements take up only as much width as necessary and do not start on a new line.
- **Characteristics**:
    - They do not start on a new line; instead, they flow inline with the text and other inline elements.
    - They occupy only the space bounded by the tags defining the element.
    - They cannot contain block elements, only other inline elements.
- **Common Inline Elements**: <span>, <a>, <strong>, <em>, <img>, <code>

Block elements are great for creating the basic structure of a webpage, while inline elements are useful for styling individual bits of content within those structures.

---

11. What are HTML form elements? Provide the HTML form elements with particular definition?

HTML form elements are used to collect user input. These elements can capture data entered by the user and send it to a server for processing.

1. `<form>`:
- **Definition**: Defines a form that collects user input.
- **Usage**: Acts as a container for various input elements.
2. `<input>`:
- **Definition**: Defines an input field.
- **Types**: Text, password, email, number, radio, checkbox, submit, etc.
3. `<textarea>`:
- **Definition**: Defines a multi-line text input control.
- **Usage**: Allows users to enter longer pieces of text.
4. `<label>`:
- **Definition**: Defines a label for an input element.
- **Usage**: Improves accessibility by associating text labels with form elements.
5. `<button>`:

- **Definition**: Defines a clickable button.
- **Types**: Submit, reset, button.

6. `<select>`:

- **Definition**: Defines a drop-down list.
- **Usage**: Allows users to choose from a list of options.

7. `<option>`:

- **Definition**: Defines an option in a drop-down list.
- **Usage**: Used within a `<select>` element

8. `<fieldset>`:

- **Definition**: Groups related elements within a form.
- **Usage**: Provides a visual grouping with a border

9. `<legend>`:

- **Definition**: Defines a caption for a `<fieldset>`.
- **Usage**: Provides a title for the group of elements.

10. `<datalist>`:

- **Definition**: Provides an autocomplete feature for input fields.
- **Usage**: Contains a set of `<option>` elements that define the possible values for an `<input>`.

11. `<output>`:

- **Definition**: Represents the result of a calculation or user action.
- **Usage**: Used to display the output of scripts or calculations.

---

12. Please create a simple HTML project on your own that contains most of the elements you have learned in this HTML module.

→ file ←

CSS

1. What does CSS stand for? And Explain what CSS is and why we use it?

CSS stands for **Cascading Style Sheets**. It's a language used to describe the presentation of a document written in HTML or XML. CSS controls how the content of web pages is displayed, including layout, colors, fonts, and overall visual aesthetics. CSS is a cornerstone technology of the web, alongside HTML and JavaScript.

We use CSS for -

1. **Separation of Concerns**: CSS allows developers to separate content (HTML) from design (CSS), making it easier to maintain and update websites.
2. **Consistency**: With CSS, you can apply consistent styling across multiple pages of a website.
3. **Flexibility**: CSS provides a wide range of styling options, from fonts and colors to complex layouts and animations.
4. **Efficiency**: By using CSS, you can reduce the amount of repetitive code in your HTML, leading to cleaner and more efficient code.
5. **Responsive Design**: CSS enables responsive web design, ensuring websites look good on all devices, from desktops to mobile phones.

---

2. How many ways can we use CSS in HTML documents and which one is the best?

There are three primary ways to use CSS in HTML documents

inline CSS : use within HTML element using *style* attribute, best for quick styling changes

internal(embedded CSS) : within a `<style>` tag inside the `<head>` section of the HTML document, suitable for styling single-page documents

external CSS : in a separate `.css` file and linked to the HTML document using the `<link>` tag, ideal for larger websites with multiple pages

Basically, the choice depends on the specific needs of our project.

External CSS is generally considered the best practice for most projects. It promotes a clean separation of content (HTML) and presentation (CSS), making it easier to maintain and update your website. It also allows for reusable styles across multiple pages, leading to more efficient and consistent styling.

3. Give a brief explanation of CSS syntax.

The syntax of Cascading Style Sheets (CSS) is relatively straightforward and consists of selectors, properties, and values.

Selectors

Selectors are used to target the HTML elements you want to style. Examples include *element selectors*, *class selectors*, and *ID selectors*.

Eg. p { properties : value;}, .class { properties : value;}, #id { properties : value;}

Properties

Properties define the aspects of the element that you want to style, such as color, font size, margin, padding, etc.

Values

Values specify the settings for the properties. Each property can have one or more values.

CSS Comments

CSS comments are used to annotate your code and are ignored by the browser. They are enclosed within `/* */`.

---

4. What are the selectors?

Selectors are a fundamental part of CSS that allow you to target and style specific HTML elements on a webpage. They determine which elements the CSS rules apply to.

1. Universal Selector

- **Syntax**: `*`

- **Description**: Selects all elements on the page.
- Eg. * {..}

2. Type Selector (Element Selector)

- **Syntax**: `element`
- **Description**: Selects all elements of a given type.
- Eg. p {..}

3. Class Selector

- **Syntax**: `.class`
- **Description**: Selects all elements with a specific class attribute.
- Eg. .myClass {..}

4. ID Selector

- **Syntax**: `#id`
- **Description**: Selects an element with a specific ID attribute (IDs should be unique within a page).
- Eg. #myId {..}

5. Attribute Selector

- **Syntax**: `[attribute]`
- **Description**: Selects elements with a specific attribute
- Eg. input[type="text"] {..}

6. Descendant Selector

- **Syntax**: `ancestor descendant`
- **Description**: Selects elements that are descendants of a specified ancestor element.
- Eg. div p {..}

7. Child Selector

- **Syntax**: `parent > child`
- **Description**: Selects elements that are direct children of a specified parent element.
- Eg. ul > li {..}

8. Sibling Selectors

- **Adjacent Sibling Selector**:
    - **Syntax**: `element + element`
    - **Description**: Selects an element that is the next sibling of a specified element.
    - Eg. h1 + p {..}

General Sibling Selector:

- **Syntax**: `element ~ element`
- **Description**: Selects all elements that are siblings of a specified element.
- Eg. h1 ~ p {..}

9. Pseudo-Class Selector

- **Syntax**: `:pseudo-class`
- **Description**: Selects elements based on their state
- Eg. a:hover {..}

10. Pseudo-Element Selector

- **Syntax**: `::pseudo-element`
- **Description**: Selects and styles a part of an element.
- Eg. p::first-line {..}

Combining Selectors

Selectors can be combined to create more specific rules.

Eg. div#content .main > p:first-child {..}

---

5. To make an element bordered, what properties are needed?

We need to use a combination of CSS properties that define the style, width, and color of the border.

`border`: A shorthand property to set the border's width, style, and color.

Eg. element { border : 2px solid red;}

`border-width`: Specifies the width of the border.

Eg. element { border-width: 2px; }

`border-style`: Specifies the style of the border (e.g., `solid`, `dashed`, `dotted`, `double`, `groove`, `ridge`, `inset`, `outset`, `none`, `hidden`).

Eg. element { border-style: solid; }

`border-color`: Specifies the color of the border

Eg. element { border-color: black; }

`border-radius`: Rounds the corners of the border.

Eg. element { border-radius: 5px; }

`border-top`, `border-right`, `border-bottom`, `border-left`: Apply borders to specific sides of the element.

```
Eg. element { border-top: 2px solid black;
              border-right: 2px dashed red;
              border-bottom: 2px dotted blue
             ; border-left: 2px solid green; }
```