WEB DEVELOPMENT

[SUMMMARY OF TRAINING]

Tajinder Kaur | Web Development | June 4, 2024- July 17, 2024

Web development:

Web development involves creating and maintaining websites and web applications. It encompasses various disciplines such as front-end development (client-side), involving HTML, CSS, and JavaScript for user interface and interaction; back-end development (server-side), managing databases, server logic, and APIs; and often includes aspects of design, usability, and security.

AREAS TO LEARN

HTML

CSS

PHP

BOOTSTRAP

JAVASCRIPT

HTML

HTML (HyperText Markup Language) is the standard markup language for creating web pages and web applications.

HTML is a markup language used to structure content on the web, defining elements and their attributes to organize text, images, and multimedia, enabling browsers to render and display information consistently across different platforms.

The basic structure of a html file is :

In this code:

- <html> encloses the entire HTML document.
- <head> contains meta-information about the document, such as its title (<title>).
- <body> contains the content of the document that is displayed in the browser.
- <h1> is a heading element, which in this case contains the text "hy".

This HTML structure will display "hy" as a heading (h1) when rendered in a web browser.

KEY POINTS TO REMEMBER:

1.TAGS: In HTML, tags are the fundamental building blocks used to define elements within a web page. Tags are enclosed in angle brackets < > and typically come in pairs: an opening tag and a closing tag. Here's a breakdown of how tags work in HTML:

1. **Opening Tag**: This is the first part of a tag and indicates the start of an element. It consists of the element's name enclosed in angle brackets. For example, <h1> is the opening tag for a level 1 heading.
2. **Closing Tag**: This is the second part of a tag and signifies the end of the element. It has the same element name as the opening tag, but with a forward slash / before the element name. For example, </h1> is the closing tag for a level 1 heading.

3.**Element Content**: This is the content between the opening and closing tags. It could be text, images, other elements, or nothing at all, depending on the element.

Example:

<h1>This is a Heading</h1>

- <h1> is the opening tag.
- This is a Heading is the content.
- </h1> is the closing tag.

Common HTML Tags:

- **<p>**: Defines a paragraph.
- **<a>**: Defines a hyperlink.
- **<img>**: Defines an image.
- ** **: Defines a division or section.
- ****: Defines a section of text within another element.
- **<ul>, <ol>, **: Defines unordered and ordered lists, and list items respectively.
- **<table>, <tr>, <td>**: Defines tables, table rows, and table cells respectively.

2. ATTRIBUTES:

Tags can also have attributes, which provide additional information about the element or modify its behavior. Attributes are specified within the opening tag and are written as name-value pairs. For example, in <a href="https://example.com">, href is an attribute that specifies the hyperlink reference.

3. SELF-CLOSING TAGS:

Some tags in HTML are self-closing, meaning they don't have separate opening and closing tags. Instead, they end with a forward slash / before the closing angle bracket. For example, <img src="image.jpg" alt="Image"> is a self-closing tag that inserts an image.

4. TABLE: In HTML, a <table> is a tag used to create structured grids for displaying tabular data. It allows you to organize information into rows and columns, making it easier for users to read and understand data presented on a web page. Here's an overview of how tables work in HTML:

Basic Structure of a Table

A simple table in HTML typically consists of the following elements:

1. **<table>**: This is the main container for the entire table.
2. **<tr> (table row)**: This tag defines a row within the table. Each row contains one or more <td> or <th> elements.
3. **<td> (table data)**: This tag defines a standard cell within a table row. It contains data cells, such as text or other HTML elements.
4. **<th> (table header)**: This tag defines a header cell within a table row. It is typically used to label the columns or provide other header information. Headers are visually bold and centered by default.

EXAMPLE:

```
<table border="1">
    <tr>
        <th>Heading 1</th>
        <th>Heading 2</th>
        <th>Heading 3</th>
    </tr>
    <tr>
        <td>Data 1A</td>
        <td>Data 1B</td>
        <td>Data 1C</td>
    </tr>
    <tr>
        <td>Data 2A</td>
        <td>Data 2B</td>
        <td>Data 2C</td>
    </tr>
</table>
```

Attributes of <table> Tag

The <table> tag can have various attributes to control the appearance and behavior of the table, such as:

- border: Specifies the width of the border around the table (e.g., border="1").
- width, height: Sets the width and height of the table.
- cellpadding, cellspacing: Specifies padding and spacing within and between cells, respectively.
- align: Aligns the table (left, center, right

5. LISTS:

In HTML, there are two main types of lists: ordered lists (<ol>) and unordered lists (<ul>). Lists are used to group related items together and are particularly useful for navigation menus, itemized content, or any sequence of items where the order or structure matters.

1. Unordered Lists (<ul>)

An unordered list is a list where the order of items does not explicitly matter, and items are typically displayed with bullet points by default.

Example:

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

- : Defines the unordered list container.
- : Defines each list item within the list. Each <li> tag represents one item in the list.

Output:

- Item 1
- Item 2
- Item 3

2. Ordered Lists (<ol>)

An ordered list is a list where the items are numbered or ordered sequentially.

Example:

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

- : Defines the ordered list container.
- : Defines each list item within the list, similarly to <ul>.

Output:

1. First item
2. Second item
3. Third item

6. DIV: The element is a fundamental part of HTML, not CSS. It stands for "division" or "division container" and is used to group together and logically organize content within a web page. Here's a breakdown of its purpose and usage:

Purpose of in HTML:

1. **Container**: acts as a container or a block-level element that can contain other HTML elements (such as text, images, forms, tables, etc.).
2. **Grouping**: It helps in grouping together related elements for styling purposes or to apply common behaviors using CSS or JavaScript.
3. **Semantic Structure**: While itself doesn't provide semantic meaning to the content (it's a generic container), it helps to structure the layout and organization of a web page.

Example :

```
<div id="header">
    <h1>Website Header</h1>
    <p>Welcome to our website!</p>
</div>

<div id="content">
    <h2>Content Section</h2>
    <p>This is the main content area.</p>
</div>

<div id="footer">
    <p>&copy; 2024 Example Company. All rights reserved.</p>
</div>
```

- elements (, ,) are used to logically group different sections of the webpage (header, content, and footer).

7. FORM: The <form> element in HTML serves as the foundation for creating interactive forms on web pages. It supports various attributes that control the behavior, appearance, and functionality of the form. These attributes help customize how form data is handled, validated, and processed both on the client-side and server-side. Understanding and using these attributes effectively can enhance the usability and functionality of forms in web development.

Various attributes of form are:

□ **autocomplete**: Specifies whether the browser should automatically complete input values based on the user's previous inputs in similar fields on the same website.

- **novalidate**: Prevents the browser from performing its default validation of form fields before submission. Useful for custom client-side validation scripts.
- **target**: Specifies where to display the response received after submitting the form. Values can include _self (default, same frame), _blank (new tab or window), _parent (parent frame), _top (top-level frame), or a custom frame name.
- **action**: Defines the URL where the form data should be sent when the form is submitted. Typically used with method.
- **method**: Specifies the HTTP method used to send the form data to the server. Common values are GET (default) and POST. GET appends form data to the URL, while POST sends it in the request body.
- **enctype**: Specifies how form data should be encoded before sending it to the server. Common values include application/x-www-form-urlencoded (default, key-value pairs), multipart/form-data (used when uploading files), and text/plain (plain text).
- **name**: Assigns a name to the form element. Useful for referencing the form in JavaScript, submitting multiple forms on a page, or handling form data server-side.
- **onsubmit**: Specifies a JavaScript function to execute when the form is submitted. Allows for custom validation, confirmation dialogs, or other actions before the form submission.
- **id**: Provides a unique identifier for the form element. Useful for targeting the form with CSS styles or JavaScript functions.
- **class**: Assigns one or more class names to the form element, allowing for styling with CSS. Multiple classes can be separated by spaces

CSS

CSS (Cascading Style Sheets) is a language used for describing the presentation of web pages, including their layout, colors, fonts, and more. It separates content from design, enhancing the visual appeal and usability of websites. CSS enables developers to style HTML elements selectively, ensuring consistent and responsive design across different devices. Its cascading nature allows styles to inherit and override based on specificity, facilitating efficient and modular web development.

three common ways to apply CSS styling, each with its syntax:

1. Inline CSS

Inline CSS applies styles directly to individual HTML elements using the style attribute.

Syntax:

```
<P STYLE="COLOR: RED; FONT-SIZE: 16PX;">HELLO, WORLD!</P>
```

2. Internal CSS

Internal CSS applies styles within the <style> element in the <head> section of an HTML document.

Syntax:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        p {
            color: blue;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>Hello, world!</p>
</body>
</html>
```

3. External CSS

External CSS links an external CSS file to an HTML document using the <link> element.

Syntax:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <p>Hello, world!</p>
</body>
</html>
```

```
/* styles.css */
p {
   color: green;
   font-size: 20px;
}
```

- **Inline CSS**: Applied directly to HTML elements using the style attribute.
- **Internal CSS**: Defined within the <style> element in the <head> section of the HTML document.
- **External CSS**: Linked externally to the HTML document using the <link> element to reference a separate CSS file.

KEYS TO REMEMBER

1.CLASS: Classes in CSS provide a powerful way to apply styles to multiple HTML elements efficiently. They promote reusability and maintainability in web design by allowing consistent styling across different elements without duplicating code. By defining and applying classes appropriately, you can achieve a cohesive and visually appealing design for your web pages.

Defining a Class: Classes are defined in CSS using the following syntax:

```
.classname {
   property: value;
```

/* Additional styles */

}

.classname: This is the selector for the class. Replace classname with a name of your choice (e.g., .button, .header, .card).

2. SELECTORS: Selectors in CSS are patterns used to select and style elements in HTML documents. They define which elements the styles should be applied to. CSS selectors can target elements based on various criteria, allowing for precise and flexible styling. TYPES OF SELECTORS:

Element selector :

- Targets elements based on their element type.

Class Selector:

- Targets elements based on their class attribute.

ID Selector:

- Targets a single element based on its id attribute.

Attribute Selector:

- Targets elements based on the presence or value of an attribute.

3.IDS: n CSS, IDs (Identifiers) are used to uniquely identify a single HTML element on a web page. They provide a specific and direct way to target and style that particular element.

Description:

- **Unique Identifier**: Each ID in an HTML document must be unique; that is, no two elements should have the same id attribute value.
- **Higher Specificity**: IDs have a higher specificity compared to classes and element selectors, meaning styles applied via IDs will override styles applied via classes or element selectors.
- **Limited Reusability**: Due to their uniqueness requirement, IDs are less reusable compared to classes, which can be applied to multiple elements.

Usage:

To apply styles to an element with a specific ID, use the following syntax:

4. FLEXBOX AND GRID:

Flexbox and CSS Grid are layout models in CSS that allow developers to create complex and responsive layouts with ease. Here's a brief description of each and how they are used:

Description: Flexbox (Flexible Box Layout) is a one-dimensional layout model. It helps in laying out items in a single row or column, making it easier to design flexible and dynamic layouts.

- **Key Features**:

  - Align items horizontally or vertically.

  - Distribute space among items.

  - Control the order of items.

  - Easily handle alignment, spacing, and wrapping of items.

Usage:

```
.container {
    display: flex; /* Establishes a flex container */
    flex-direction: row; /* or column, controls main axis */
    justify-content: center; /* Align items along the main axis */
    align-items: center; /* Align items along the cross axis */
}
```

CSS Grid:

- **Description**: CSS Grid Layout is a two-dimensional layout system that allows you to design complex layouts with rows and columns. It provides precise control over the placement and sizing of grid items.

- **Key Features**:

  - Define rows and columns.

  - Place items anywhere in the grid.

  - Control spacing between grid items.

  - Handle responsiveness with media queries.

Usage:

```css
.container {
    display: grid; /* Establishes a grid container */
    grid-template-columns: 1fr 1fr 1fr; /* Defines columns */
    grid-gap: 10px; /* Creates space between grid items */
}
```

5. BOX MODEL:

The CSS Box Model is fundamental to understanding how elements are structured and sized in web pages. It consists of several components that define an element's dimensions, spacing, padding, borders, and margins.

Components of the Box Model:

1. **Content**: This is the actual content of the HTML element, such as text, images, or other media.
2. **Padding**: The padding is the space between the content and the element's border. It helps create space within the element.
3. **Border**: The border surrounds the padding and content. It can have a width, style (e.g., solid, dashed), and color.
4. **Margin**: The margin is the space outside the border. It creates space between adjacent elements.

Box Model Structure:

- **Content Box**: Represents the actual content area where text and images are displayed.
- **Padding Box**: Area between the content and the border. Padding adds space inside the element.
- **Border Box**: Surrounds the padding and content. Defines the boundary of the element.
- **Margin Box**: Space outside the border. Determines the gap between adjacent elements.

Example:

```css
.box {
    width: 200px; /* Sets the width of the element */
    height: 100px; /* Sets the height of the element */
    padding: 20px; /* Adds padding inside the element */
    border: 2px solid #ccc; /* Adds a border around the element */
    margin: 10px; /* Creates space outside the element */
}
```

6. DROP-DOWN:

```
.dropdown {
    position: relative;
    display: inline-block;
}
.dropbtn {background-color: #4CAF50;
    color: white;
    padding: 10px;
    font-size: 16px;
    border: none;
    cursor: pointer}
.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}
.dropdown:hover .dropdown-content {
    display: block;
}
.dropdown-content a {
   color:  black;
   padding: 12px 16px;
    text-decoration: none;
    display: block;
```

```
}

.dropdown-content a:hover {

    background-color: #f1f1f1;

}
```

PHP

PHP (Hypertext Preprocessor) is a widely-used open-source server-side scripting language designed primarily for web development. It's embedded into HTML and executed on the server, generating dynamic content that can be sent to the client's web browser.

```php
<!DOCTYPE html>
<html>
<head>
    <title>PHP Example</title>
</head>
<body>

<?php
    // PHP code starts with <?php and ends with ?>
    $name = "John Doe";
    $age = 30;
    echo "<h1>Hello, $name!</h1>";
    echo "<p>You are $age years old.</p>";
?>

</body>
</html>
```

KEY POINTS

1. **Variables in PHP:**

- Variables in PHP start with a `$` sign followed by the variable name.

- They are case-sensitive and can store various types of data.

- Example: `$name = "John";`

2.**Loop Statements in PHP:**

- **`for` loop**: Executes a block of code a specified number of times.

- **`while` loop**: Executes a block of code as long as a specified condition is true.

- **`do-while` loop**: Similar to `while` loop but executes the code block once before checking the condition.

- **`foreach` loop**: Iterates over arrays or objects.

3. **Data Types in PHP:**

- **Integer**: Whole numbers, e.g., `42`.

- **Float (Double)**: Numbers with a decimal point, e.g., `3.14`.

- **String**: Sequence of characters, e.g., `"Hello World"`.

- **Boolean**: Represents `true` or `false`.

- **Array**: Holds multiple values in a single variable.

- **Object**: Instances of user-defined classes.

- **NULL**: Represents a variable with no value assigned.

4. Print Statement in PHP:

- Used to output text or variables to the screen.

- `echo` and `print` are commonly used print statements.

- Example: `echo "Hello, World!";`

5. Operators in PHP:

- **Arithmetic Operators**: `+`, `-`, `*`, `/`, `%` (modulo).

- **Assignment Operators**: `=`, `+=`, `-=` etc.

- **Comparison Operators**: `==`, `!=`, `<`, `>`, `<=`, `>=`.

- **Logical Operators**: `&&` (and), `||` (or), `!` (not).

- **String Operators**: `.` (concatenation).

- **Increment/Decrement Operators**: `++`, `--`.

PHP WITH MYSQL

Using PHP with MySQL allows developers to create dynamic web applications that interact with databases to store, retrieve, and manipulate data. Here's an overview of how PHP and MySQL are typically used together:

Connecting to MySQL Database:

1. **Connecting to MySQL Server:**
    - Use mysqli_connect() or PDO (PHP Data Objects) to establish a connection to the MySQL database server.
    - Example with mysqli_connect()

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "database_name";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

2. Querying MySQL Database:

Executing SQL Queries: Use mysqli_query() or PDO to execute SQL queries (SELECT, INSERT, UPDATE, DELETE) on the database.

```php
<?php
$sql = "SELECT id, firstname, lastname FROM users";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // Output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["la
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

3. Fetching and Displaying Data:

- Use mysqli_fetch_assoc(), mysqli_fetch_array(), or PDO methods to fetch data from query results.

BOOTSTRAP

Bootstrap is a popular front-end framework for building responsive and mobile-first websites and web applications. It provides a set of CSS and JavaScript components and utilities that simplify the process of designing consistent and visually appealing layouts across different devices. Here's an overview of Bootstrap:

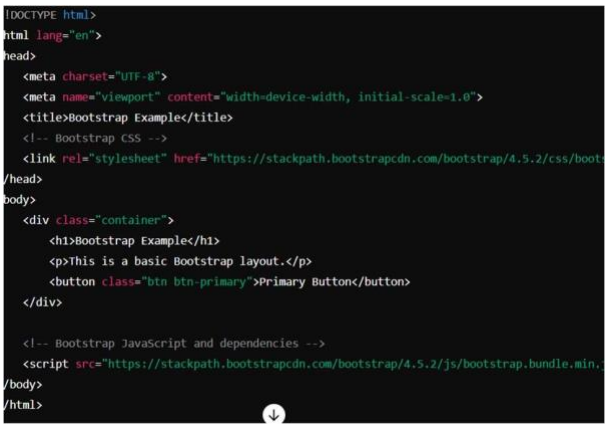
Key Features of Bootstrap:

1. **Responsive Grid System:**
   - Bootstrap uses a responsive, mobile-first fluid grid layout system with 12 columns.
   - Grid classes (col-, col-sm-, col-md-, etc.) allow developers to create responsive layouts easily.
2. **Pre-styled Components:**
   - Bootstrap offers a wide range of pre-styled components such as buttons, forms, navigation bars, dropdowns, alerts, modals, and more.
   - These components are designed to be customizable and reusable.
3. **Bootstrap Themes and Templates:**
   - Bootstrap comes with built-in themes and templates that can be customized or used as-is to accelerate development.
   - Third-party themes and templates are also available, offering a variety of styles and designs.
4. **JavaScript Plugins:**
   - Bootstrap includes JavaScript plugins such as carousel, modal, collapse, tooltip, popover, and scrollspy.
   - These plugins enhance user experience and add interactive functionalities to websites.
5. **Customizable with Sass:**
   - Bootstrap can be customized using Sass (Syntactically Awesome Style Sheets), allowing developers to modify variables, mixins, and functions to create custom themes.
6. **Browser Compatibility:**
   - Bootstrap ensures compatibility with popular browsers and devices, ensuring consistent appearance and functionality across different platforms.

Getting Started with Bootstrap:

To use Bootstrap in your project, include the Bootstrap CSS and JavaScript files in your HTML document, either by downloading them or linking to a CDN (Content Delivery Network).

Example HTML Structure with Bootstrap:



```
!DOCTYPE html>
tml lang="en">
ead>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Bootstrap Example</title>
   <!-- Bootstrap CSS -->
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boot
/head>
body>
   <div class="container">
      <h1>Bootstrap Example</h1>
      <p>This is a basic Bootstrap layout.</p>
      <button class="btn btn-primary">Primary Button</button>
   </div>

   <!-- Bootstrap JavaScript and dependencies -->
   <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.bundle.min.
/body>
/html>
```

JAVASCRIPT

JavaScript is a versatile programming language primarily used for creating interactive and dynamic web content. Here's an overview of JavaScript, its features, and how it's commonly used:

Features of JavaScript:

1. **Client-Side Scripting:** JavaScript runs on the client-side (in the user's web browser), enabling dynamic interactions with web pages without requiring server-side processing.
2. **Cross-platform:** JavaScript is supported by all modern web browsers, making it a universal scripting language for web development.
3. **Dynamic Typing:** Variables in JavaScript are dynamically typed, meaning they can hold values of any data type.
4. **Object-Oriented:** JavaScript supports object-oriented programming concepts like objects, classes (with ES6), inheritance, and encapsulation.
5. **Event-Driven:** JavaScript facilitates event-driven programming, where actions (such as clicks or keystrokes) trigger specific functions or behaviors.
6. **Asynchronous Execution:** Supports asynchronous programming through callbacks, promises (ES6+), and async/await (ES8+), enabling non-blocking operations.
7. **Built-in APIs:** Provides built-in APIs (Application Programming Interfaces) for manipulating the DOM, handling events, performing AJAX requests, etc.

Example Use Cases:

- **DOM Manipulation:** JavaScript is commonly used to manipulate HTML elements and update the DOM dynamically based on user interactions or data changes.
- **Form Validation:** Validate user inputs on forms in real-time using JavaScript to ensure data integrity before submission.

Example JavaScript Code Snippet:

```
document.addEventListener('DOMContentLoaded', function() {
    const button = document.getElementById('myButton');
    const output = document.getElementById('output');

    button.addEventListener('click', function() {
        output.innerHTML = 'Button clicked!';
    });
});
```

Conclusion

i. **HTML Basics**: Markup language for structuring web content; tags define structure, headings, paragraphs, lists, links, and more.

ii. **CSS Basics**: Stylesheet language for design; selectors target HTML elements to apply styles like colors, fonts, layout, and responsiveness.

iii. **JavaScript Basics**: Client-side scripting language for web interactivity; manipulates DOM, handles events, asynchronous operations, and enhances user experience dynamically.

iv. **PHP Basics**: Server-side scripting language for web development; integrates with databases (like MySQL), handles forms, sessions, and dynamic content generation.

v. **Bootstrap**: Front-end framework for responsive web design; provides CSS styles, components (buttons, forms), grid system, and JavaScript plugins for UI development.

vi. **MySQL with PHP**: Database management system used with PHP for storing, retrieving, and manipulating data; supports SQL queries and secure data handling.

vii. **JavaScript with HTML**: Integrates client-side scripting in HTML; enables dynamic content, event handling, DOM manipulation, and enhances user interface interactivity.