Web-technology

Web technology is the tools and technique which enables two or more Computing devices to communicate over a network or internet. Web technology consists of two words. The web refers to the world & web generally known as a world wide web. WWW is the cyber space containing web pages, documents and any other resources which are identified and located with the help of their URL. Technology refers to the tools and technique that makes web browser to view contents of web, programming languages and framework for the development of website. Database is used to add data back end, protocols for communicating on the web, multimedia elements etc.

Web development is the process of designing and developing websites that are hosted on the internet or an intranet. This process ranges from creating simple static pages to complex web-based applications, social media sites, and e-commerce platforms. Web development includes web design, web content development, client-side scripting, server-side scripting, web engineering, and more. Since web development consists of several interrelated tasks, it can be accomplished by different types of developers who focus on various aspects of web creation.

Web Technology

- **Definition**: Web technology refers to the tools and techniques that enable communication between two or more computing devices over a network or the internet.
- **Components**:
    - **The Web (World Wide Web)**: A cyberspace containing web pages, documents, and other resources identified and located using URLs.
    - **Technology**: Involves the tools and techniques for:
        - Viewing web content (web browsers).
        - Developing websites (programming languages and frameworks).
        - Managing data back-end (databases).
        - Communicating on the web (protocols).
        - Integrating multimedia elements.

Web Development

- **Definition**: The process of designing and developing websites that are hosted on the internet or an intranet.
- **Scope**:
    - Ranges from creating simple static pages to complex web-based applications, social media sites, and e-commerce platforms.
- **Components**:
    - **Web Design**: The visual and functional design of websites.
    - **Web Content Development**: Creation and management of content for websites.
    - **Client-Side Scripting**: Code that runs in the user's web browser (e.g., HTML, CSS, JavaScript).
    - **Server-Side Scripting**: Code that runs on the web server (e.g., PHP, Python, Ruby).
    - **Web Engineering**: The systematic approach to web development, covering both technical and non-technical aspects.
- **Types of Developers**:

- **Front-End Developers**: Focus on client-side development, including design and user interaction.
- **Back-End Developers**: Focus on server-side development, including databases and server logic.
- **Full-Stack Developers**: Handle both front-end and back-end development tasks.

Server-side and client-side scripting

1. *Server-side scripting programming*

Server-side scripting, also known as back-end development, runs on the server where the application is hosted. The server side is used to serve content depending on user requests. The back end helps create dynamic web applications that allow users to interact and communicate with the application. Back-end languages also help connect the front end with the database, enabling users to store and retrieve data as needed. Some of the popular server-side scripting languages are ASP, Node.js, Perl, PHP, Ruby, and Python.

2. *Client-side scripting programming*

Client-side scripting is used to generate actions that run on the client side (front end) browser without needing server-side scripting. It is embedded within HTML documents. Client-side scripting can be used to layout the contents of a web page. For example, when a user makes a request through a web browser for a web page from the server, the server sends the HTML and CSS as plain text. The browser then interprets and renders the contents of the web page for the user.

Differences Between Server-Side Scripting and Client-Side Scripting

Server-Side Scripting	**Client-Side Scripting**
Executes on the web server.	Executes in the user's web browser.
Generally slower due to server processing and network latency.	Faster as it runs directly in the browser after the page is loaded.
Has access to server resources like databases, files, and system functions.	Limited to resources available in the browser and cannot directly access server resources.
More secure for sensitive operations as the code is not exposed to the user.	Less secure for sensitive operations as the code can be viewed and potentially manipulated by the user.
Handles tasks like form submissions, data retrieval, and user authentication.	Enhances user experience with dynamic content, animations, and interactive elements without reloading the page.
Common languages include PHP, Python, Ruby, Java, and Node.js.	Primarily uses JavaScript, along with HTML and CSS for structure and style.

Full stack developer

Full stack developer understands both front end and back end development process they can accomplish entire project alone they most have knowledge client site and server site scripting language and a great knowledge of integrating data base with application.

HTML

HTML is the backbone of web pages. It's like the blueprint for a website. Think of it as a recipe for a sandwich. Just like a recipe tells you what ingredients to use and how to arrange them to make a sandwich, HTML tells a web browser what elements to display on a webpage and how to structure them.

Basic HTML Page

```
<!DOCTYPE html>

<html>

        <head>

                <title>My First Page</title>

        </head>

        <body>

                <p>hello world</p>

        </body>

</html>
```

Quick Points

Html tag is parent of head & body tag.

Most of html elements have opening & closing tags with content in between.

Some tags have no content in between, eg - <br>

We can use inspect element/view page source to edit html.

HTML is NOT case sensitive

<p> = <P>

<html> = <HTML>

<head> = <HEAD>

<body> = <BODY>

HTML Elements and attributes

An HTML element is a fundamental building block of HTML (Hypertext Markup Language), which is used to structure and display content on the web. An HTML element typically consists of a start tag, content, and an end tag.

Structure of an HTML Element

1. Start Tag: This indicates the beginning of an HTML element. It is enclosed in angle brackets.
2. Content: This is the information or other HTML elements nested within the start and end tags.
3. End Tag: This indicates the end of an HTML element and is also enclosed in angle brackets but includes a forward slash before the element's name.

Common HTML Elements

- Headings: <h1>, <h2>, <h3>, etc. (define headings of different level)
- Paragraph: <p> (represents paragraph of text)
- Link: <a href="URL"> (created hyperlink to other pages)
- Image: <img src="URL" alt="description"> (embeds images on a webpage)
- List: <ul>, <ol>, <li> (used to create ordered and unordered list)
- Table: <table>, <tr>, <td>, <th> (constructs tables to organize data)

HTML attributes: HTML attributes provide additional information about HTML elements. Attributes are used to add more information to the tag. They are always included in the opening tag of an element and consist of a name and a value, separated by an equal's sign. Attributes help to define the properties and behavior of HTML elements.

Common HTML Attributes

1. id: Specifies a unique id for an HTML element.

   <div id="header"></div>

2. class: Specifies one or more class names for an element (used for CSS and JavaScript).

   <p class="intro">This is an introductory paragraph. </p>

3. style: Specifies an inline CSS style for an element.

   <h1 style="color: blue;">This is a blue heading</h1>

4. title: Provides extra information about an element (displayed as a tooltip when the mouse hovers over the element).

   <abbr title="HyperText Markup Language">HTML</abbr>

5. href: Specifies the URL of a link (used in <a> elements).

   <a href="https://www.example.com">Visit Example</a>

6. src: Specifies the URL of an image (used in <img> elements).

   <img src="image.jpg" alt="Description of image">

7. alt: Provides alternative text for an image if the image cannot be displayed (used in <img> elements).

```
<img src="image.jpg" alt="Description of image">
```

8. name: Specifies the name of an element (used in <input>, <select>, and <form> elements).

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

9. type: Specifies the type of an element (used in <input> elements).

   ```
   <input type="button" value="Click Me">
   ```

10. value: Specifies the value of an input element (used in <input> and <option> elements).

    ```
    <input type="text" value="Default Text">
    ```

11. placeholder: Provides a short hint that describes the expected value of an input field (used in <input> and <textarea> elements).

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

12. disabled: Specifies that an element should be disabled.

    ```
    <button disabled>Click Me</button>
    ```

13. readonly: Specifies that an input field is read-only.

    ```
    <input type="text" value="Cannot change this" readonly>
    ```

14. checked: Specifies that an input element should be pre-selected (used in <input> elements of type "checkbox" or "radio").

    ```
    <input type="checkbox" checked>
    ```

15. action: Specifies the URL where the form data should be sent (used in <form> elements).

    ```
    <form action="/submit-form" method="post">
    ```

16. method: Specifies the HTTP method to use when sending form data (used in <form> elements).

    ```
    <form action="/submit-form" method="post">
    ```

Some more important terms of technology

1. Web Server

A web server is a specialized high-end computer that hosts websites on the internet. Today, cloud services can also act as web servers. The basic objective of a web server is to store, process, and deliver web pages. Web pages mostly contain static content, including HTML documents, images, stylesheets, and JavaScript files.

2. URL (Uniform Resource Locator)

A URL, as the name suggests, provides a way to locate a resource on the web, the hypertext system that operates over the internet. The URL contains the name of the protocol to be used to access the resource and the resource name. The first part of the URL identifies the IP address or domain name where the resource is located.

3. Protocol

For two computers on a network to communicate successfully they must share a common set of rules about how to communicate. At a minimum such rules must include how to interpret signal, How to identify one self and other computers and network, how to insert, how to manage information exchange across the network medium.

4. Email

It is the fast, easy and inexpensive way to send and receive messages with other internet user around the world. E-mail stands for electronic mail. It is the most popular service provided by the internet.

5. Hyper Text Transfer Protocol (HTTP)

Web page are constructed according to a standard method called hypertext markup language. And HTML page is transmitted over web in a standard way and format known as HTTP. This protocol uses (IP/TCP) to manage the web transmissions in encrypted form to provide security for sensitive data.

6. DNS

The domain name system is a naming data base in which internet domain name are located and translated into internet protocol. The domain name system maps the name people used to locate a website to the Ip address that a computer uses to locate that website.

7. Ip address

An IP address is a unique address that identifies a device on the internet or a local network. IP stands for internet protocol which is set of the rules, governing the format of data sent via the internet or local network.

8. API (Application programming Interface)

A web API is an application programmer interface for the web. A browser API can extend the functionality of a web browser.

9. CMS (content management system)

A content management system is an application that is used to manage content, allowing multiple contributes to be created, edited as well as store digital Content. Content in a CMS is typically stored in a data base and displayed in a presentation based on a set of templates like a website.

The following are common Features of CMS: -

- Easy to use
- Easy to search for information.
- Easy to manage content

- Accessible from anywhere
- Allows multiple users
- Instant content updates
- Easy to Scale
- Easy to update

Web publishing and web hosting

Web publishing: It is the process of uploading files, updating web pages and posting block on the internet. It is actually the activity of making the websites this includes the designing the content additions, data base adding etc. Its main aim is to facilate communications, simply by adding Context through Style, and space. It is also known as Online publishing.

Web hosting: web hosting service types of internet hosting services that allows individuals and organizations, to make their own website accessible via the world wide web. Web host are companies that provide space on a server that own or lease for used by their Client as well as providing internet Connectivity, typically in a data center.

HTML Tag

A container for some content or other HTML tags. They are the component used to design the structure of websites. Tags are enclosed in angle brackets and usually come in pairs: a start tag and an end tag, although some tags are self-closing.

<p> This is a paragraph</p>

HTML tags can be broadly classified into two categories: container tags and empty tags.

- **Container Tags**

Container tags, also known as non-void elements, consist of an opening tag, content, and a closing tag. These tags define a section or block of content and can contain text, other HTML tags, or both.

Structure of Container Tags:

<start-tag>Content</end-tag>

Examples of Container Tags:

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

Division (<div>**)**:

```
<div>
   <p>This is a paragraph inside a div.</p>
</div>
```

Heading (<h1>**,** <h2>**, etc.)**:

```
<h1>This is a heading</h1>
```

List (<ul>**,** <ol>**, and** <li>**)**:

```
<ul>
   <li>List item 1</li>
   <li>List item 2</li>
</ul>
```

Table (<table>**,** <tr>**,** <td>**, etc.)**:

```
<table>
   <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
   </tr>
   <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
   </tr>
</table>
```

- **Empty Tags**

Empty tags, also known as void elements, do not have any content or closing tags. They are self-closing and are used to insert elements that do not wrap any content.

Structure of Empty Tags:

```
<tag-name >
```

Examples of Empty Tags:

Image (<img>**)**:

```
<img src="image.jpg" alt="Description of image" />
```

Line Break (<br>**)**:

```
This is a line break<br />
This is the next line.
```

Horizontal Rule (<hr>**)**:

```
<hr />
```

Meta (<meta>**)**:

<meta charset="UTF-8" />

Link (<link>**)**:

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

Input (<input>**)**:

<input type="text" placeholder="Enter your name" />

Source (<source>**)**:

<source src="video.mp4" type="video/mp4" />

Area (<area>**)**:

```
<map name="map">
    <area shape="rect" coords="34,44,270,350" alt="Description" href="example.html" />
</map>
```

Document Head

Information placed in this section essential to the inner working of documents and has nothing to do with the contents of documents that information placed within the <head></head> tag isn't display in the browser.

Document body

The tags are used to indicate the start and end of the main body of texture information.

<body>

……….

……….

</body>

The attributes that the body tag takes are:

- Bg color: is used to change the default background color.
- Background: is used to specify the name of file that will be used as the background of the documents.
- Text: is used to change the default body text color.
- Link: Define the color of the unvisited link in a document.
- a link: Define the color of link as it's being Clicked
- vlink: Define the color of the link after it has been visited.

- Left margin: Set the left margin
- Top margin: Set the top margin

Line break

when text need to start from a new line and not continue on the same line. Used to add next line (line breaks) to your page. The <br> tag is be used.

Comments in HTML

This is part of code that should not be parsed.

<! -- This is an HTML Comment -->

Heading Tag

Used to display headings in HTML

h1 (most important)

h3

h2

h4

h5

h6 (least important)

HTML automatically adds an extra blank after a heading. Align attributes is used to change the alignment of the text. There are 3 alignment (left, right and center).

Paragraph Tag

Used to add paragraphs in HTML

<p> This is a sample paragraph </p>

Drawing Lines

The tag <hr> draws the horizontal lines across the whole page whenever specified.

-The attributes to the <hr> tag are:

- Align: It aligns the line on the browser screen. Possible values are left, right and center. By default, aligned to the center of the screen.
- Size: Changes the size (thickness) of the rule.
- Width: Set the width of the rule. It can be set to the fixed number of pixels to a percentage of the available screen width.
- Color: changes the color of the line.
- No shade: Display a flats 2D effects line eliminate 3D line.

eg : <hr align="left"
width= 100
Size = "4" NOSHADE

Formatting text

The process of changing the appearance the HTML text.

Various text formatting tags are:

<b> Bold </b>

<i> Italic </i>

<u> Underline </u>

<em>emphasized text</em>

<pre>preformatted text</pre> (Used to display text as it is (without ignoring spaces & next line)

<sup>superscript text</sup>

<sub>subscript text</sub>

<big>big text</big>

<small>small text</small>

<ins>inserted text</ins>

Working with fonts:

All text specified within the tag <font> </font>will appear in the font, size and color as specified as attributes of the tag (<font>) The attributes are:

- Face: Set the font to the specified font name
- Size: Set the size of the text
- Color: Set the color of the text

List in HTML

Lists are used to represent real life list data.

Types of list:

1. **Unordered list (Bullets)**
   An unordered list starts with the tag <ul> and ends with </ul>.
   Each list items starts with the tag </li>
   The attributes that can be specified in <ul> tag is:
   type =" fill-round": It will give a Solid round black bullet.
   type = "Square": It will give a solid Square black bullet.
   type=" Circle": It will give a circle bullet.

<ul>

<li> Apple </li>

<li> Mango </li>

</ul>

2. Ordered list (Numbering)

An ordered list starts with the tag <ol> and ends with </ol>.

Each list items starts with the tag<li>

The attributes that can be specified in <ol> tags tare:

Type: controls the Controls the numbering Scheme to be

Type = "1" will give counting number (1,2 ,3---)

Type=” A” will give upper case letter (A, B, C---)

Type= "a" will give lower case letters. (a, b, c, ---)

Type = "I" will give upper-case roman numeral

Type = "i" will give lower case roman numerals

Start: change the numbering sequence. It is always numeric and is converted automatically according to the Type value.

Value: Change the numbering Sequence in the middle of ordered list. It is to be Specified with the <li> tag.

Ordered list

<ol>

<li> Apple </li>

<li> Mango </li>

</ol>

3. Definition list

In this kind of list, each item in the list has a term and a definition.

Definition list values appears within tag <dl>----</dl>

Definition list consists of 2 parts-

the Definition term: Appear after the tag. <dt>

Definition description: Appear after the tag <dd>

Anchor Tag

Used to add links to your page. Clicking on a section of a text or on image in one web page will open entire web page or an image. Every hyperlink text is underlined. The default is blue and can be set dynamically via html program if required.

The hyperlink text image is underlined. when the mouse cursor is placed over it the standard arrow shaped mouse change to the shape of hand. To change the default link colors there are three attributes: (ALINK, VLINK, LINK) that can be specified with the <body> tag. Links are created in a web page by using the <a>...</a> tag. Anything written between the <a> and </a> tag becomes hyperlinks.

Syntax:<a href="https://google.com"> Google </a>

Hyperlink can be of two types:

1. Links to an external document
2. Links to a specified place within the same document (generally done when web page has large text).

Marquee

The <marquee> tag is an HTML element used to create a scrolling effect for text or images.

It scrolls the content horizontally or vertically within a defined space on a web page.

Attributes used:

Width: specifies the width

Height: specifies the height

Behavior: Defines the scrolling behavior (scroll, slide, alternate).

Direction: Specifies the direction of the scroll (left, right, up, down).

Scrollamount: Sets the speed of the scrolling (in pixels).

Scrolldelay: Sets the delay between each scroll movement (in milliseconds).

Loop: Specifies the number of times the marquee will loop.

Image Tag

Used to add images to your page.

<img src="/image.png" alt="Random Image">

Attributes:

Src: set the part of the picture

Align: set the align for the picture to left, right or center.

Border: set the border for the picture

Height: set the height of image

Width: set the width of image

Alt: display text incase image is not displayed

Tables in HTML

Tables are used to represent real life table data.

<tr> used to display table row

<td> used to display table data

<th> used to display table header

A table is a two-dimensional matrix consisting of rows and columns. It is used for displaying the data in a webpage.

Table Attributes:

Bg color- Set the background color of the table.

Width: set the width to a specific number of pixels to a percentage of the available Screen width. If width is not specified, the data is adjusted based on the cell data value.

Border: Control the border to be placed around the table the border thickness is specified in pixels. The default is No Border.

Background: set the background images of the table

Border color: Set the Color out line to a border of a cell

Cell padding: Define the space between the edge of cell and its Content.

Cell spacing: Define the space between the adjacent cell in the table

Align: Determine the horizontal of the table possible values are left, center or right.

Table data and table header attributes:

- Colspan: specifies number of columns the cell is to occupy. (colspan="n" used to create cells which spans over multiple columns
- Rowspan: specifies the number of rows the cell is to occupy.
- Align: align the cell data to top, middle or bottom.
- Bg color: Set background color at the back of a cell.
- Background: Set background image the back a cell.
- valign: It align cell data to top, middle or bottom.

Caption in Tables

<caption> Student Data </caption>

thead & tbody in Tables

<thead> to wrap table head

<tbody> to wrap table body

Caption acts as a tittle for the table which gives the reader of contexts of the information in the table.

- The table caption can be made to appear above or below the table structure with the help of attributes ALIGN:

Align = "bottom": place the caption immediately below the table.

Align="top": place the caption immediately above the table.

Example:

```
</table>

<th> Name </th>

<table>

<tr>

<th> Roll No </th>

</tr>

<td> Shradha </th>

<tr>

<th> 1664 </th>

</tr>
```

Div Tag

Div is a container used for other HTML elements. A web page can be divided into segments or division called div. Each segment starts with <div> and ends with </div> tag. The div tag has position attribute that can take one of the two values absolute or Relative.

Absolute position the segment with respect to the top/left edge of the browser window.

Relative position the segment in relation to the other elements on the page.

Frames

When the browser screen is divided into more than one unique html recognizable sections, each such section is called frames.

Each frame can be loaded with different documents and hence, multiple document can be seen concurrently.

Once the browser screen is divided into rows and column, each unique section define can be loaded with different html document. This is accomplished by using <FRAME> tag which takes in the following attributes.

1. Src="URL": Indicates the URL of the documents to be loaded into the frame.
2. Marginheight="n": Specifies the amount of white space to be left at the top and bottom of the frame.

3. Marginwidth="n": Specifies the amount of white space to be left along the side of the frame
4. Name="name": Gives frame a unique name so it can be targeted by other documents. The name given must begin with alphanumeric characters.
5. Noresize: Disable the frames resizing capability.
6. Scrolling: Control of the appearance of the horizontal and vertical scroll bar in a frame.
   possible values are as below:
   Yes: Always display the Scrollbar
   No: Never
   Auto: Browser will decide based on frame contain
7. Bgcolor Border color = Allow choosing a Color for the frame border color.
8. Frame Borden Specifies whether or not the frame has a visible border, the default value 1, tells the browser to draw a border between the frame and all adjoining frame. The values indicate that no border Should be drawn.

<noframe> tag

A frameset document has no <body> tag. It must not have one because the browser ignores any content within <body> tags if it finds <body> content before it encounters the first <frameset> tag. The <noframes> tag contains content that should only be rendered when frames are not displayed. <noframes> is typically used in a frameset document to provide alternative content for browsers that do not support frames or have frames disabled.

Page Layout Techniques

using Semantic tags for layout

<header>

- <main>

<footer>

Section Tag

<section>For a section on your page

Article Tag

<article>For an article on your page

Aside Tag

<aside>For content aside main content(ads)

Revisiting Anchor Tag

<a href="https://google.com" target="_main"> Google </a>

<a href="https://google.com"> <img src="link"> </a>

Span Tag

Span is also a container used for other HTML elements

Inline Element (takes width as per size)

Form in HTML

Forms are used to collect data from the user

Eg- sign up/login/help requests/contact me

```
<form>
form content
</form>
```

Action in Form

```
<form action="/action.php" >
```

Action attribute is used to define what action needs to be performed when a form is submitted

Form Element: Input

```
<input type="text" placeholder="Enter Name">
```

Label

```
<label for="id1">
<input type="radio" value="class X" name="class" id="id1">
</label>
<label for="id2">
<input type="radio" value="class X" name="class" id="id2">
</label>
```

Checkbox

```
<label for="id1">
<input type="checkbox" value="class X" name="class" id="id1">
</label>
<label for="id2">
<input type="checkbox" value="class X" name="class" id="id2">
</label>
```

Textarea

```
<textarea name="feedback" id="feedback" placeholder="Please add Feedback">
```

```
</textarea>
```

Select

```
<select name="city" id="city">
<option value="Delhi"> Delhi </option>
<option value="Mumbai"> Delhi </option>
<option value="Bangalore"> Delhi </option>
</select>
```

iframe Tag

website inside website

```
<iframe src="link"> Link </option>
```

Video Tag

```
<video src="myVid.mp4"> My Video </video>
```

Unit 2: Cascading Style Sheet

CSS (Cascading Style Sheet)

CSS is a language that define the design and layout of web pages. In other word CSS controls how, web pages look when loaded in web browser. We can decide design and layout of the "styl e" of Page. CSS the standard language for styling and typically works in conjunction with HTML. CSS stands for cascading style sheets. Style sheets refers to CSS document itself and Cascading refers to how styles rules are applied to page element.

Difference between HTML and CSS

HTML and CSS go hand in hand in building the web pages. HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two fundamental technologies for building web pages. HTML is used to structure the content of a webpage by defining elements such as headings, paragraphs, images, and links. It provides the semantic meaning of the content. CSS, on the other hand, is used to style and layout web pages. It controls the visual presentation, including colors, fonts, spacing, and positioning of elements. While HTML focuses on the content and its organization, CSS focuses on the appearance and design, allowing for the separation of content from presentation.

Benefits of CSS

1. Less coding: CSS reduces the amount of repetitive code by allowing the reuse of styles across multiple elements and pages.
2. Most styling options: CSS provides a wide range of styling options, including fonts, colors, layouts, and animations, giving developers extensive control over the presentation.
3. Standardization: CSS ensures consistent styling and layout across different browsers and devices, promoting a standardized look and feel.
4. Better performance: By using external stylesheets, CSS can improve web page load times and performance, as the stylesheets are cached by the browser.
5. Separation of Content and Design: CSS allows for the separation of content (HTML) from its presentation, making it easier to manage and maintain both.
6. Consistent Design: CSS enables consistent styling across multiple web pages, ensuring a uniform look and feel.
7. Improved Load Times: By using external stylesheets, web pages can load faster since the CSS file is cached by the browser after the first load.
8. Enhanced Flexibility: CSS offers more design flexibility and control over the layout, such as positioning, spacing, and animations.
9. Accessibility: CSS helps improve accessibility by allowing more control over how content is presented to different devices and screen readers.
10. Responsive Design: CSS enables responsive design, allowing web pages to adapt to different screen sizes and devices.
11. Simplified HTML: By offloading styling to CSS, HTML code becomes cleaner and more semantic, improving readability and maintainability.

CSS Advantages

1. **Less Coding**: Reduces redundancy by reusing styles across multiple elements and pages.
2. **Consistency**: Ensures a uniform look and feel across all web pages.

3. **Performance**: Improves load times with external stylesheets that are cached by browsers.
4. **Flexibility**: Offers extensive styling options, including fonts, colors, layouts, and animations.
5. **Responsive Design**: Facilitates the creation of designs that adapt to various screen sizes and devices.
6. **Separation of Concerns**: Separates content structure (HTML) from presentation (CSS), making maintenance easier.
7. **Accessibility**: Enhances accessibility by allowing tailored presentations for different devices and screen readers.

Disadvantages of CSS

1. **Browser Compatibility**: Different browsers may render CSS differently, requiring additional testing and adjustments.
2. **Complexity**: Advanced CSS can become complex and difficult to manage, especially in large projects.
3. **Learning Curve**: Requires learning and understanding of various properties, selectors, and concepts.
4. **Cascading Issues**: Inheritance and specificity can sometimes lead to unexpected results, making debugging challenging.
5. **Maintenance**: Managing large stylesheets can become cumbersome without proper organization.
6. **Lack of Variables (Older Versions)**: Older CSS versions did not support variables, making repetitive values harder to manage (this is less of an issue with modern CSS using custom properties).
7. **No Logic or Conditions**: CSS lacks logical statements or conditions, limiting dynamic styling based on user interactions or data (this is mitigated with the use of JavaScript).

CSS animation

Css Animations allows you to create complex, timed animations with key frame.Key frames define specific stage of the animations and their associated styles. Animations are defined using the animation property in CSS. They can be control with property like animation name, animation durations, animation timing function animation delay and animation iteration count.

CSS variable

CSS variable is also known as custom properties are a powerful Features in modern CSS that allows you to define and reuse values in your style sheets. They provide more flexibility and maintain ability in your CSS code by enabling you to define values and use them throughout your stylesheet.

CSS flexbox

CSS Flexbox, or the Flexible Box Layout, is a layout model that simplifies the design and alignment of elements within a container. It allows you to distribute space and align items efficiently, making it a powerful tool for creating responsive and complex layouts. Here is a brief overview:

1. Container and Items:

Flexbox works by dividing a container into a flex container and its child items, which become flex items. The container is defined with display: flex; or display: inline-flex; This establishes a new flex formatting context.

2. Main Axis and Cross Axis:
Flexbox introduces two axes: the main axis and the cross axis. The main axis is defined by the flex-direction property, and the cross axis is perpendicular to it. The main axis determines how flex items are laid out, and the cross axis is used for alignment.
3. Alignment:
Flexbox provides properties like justify-content (for aligning items along the main axis) and align-items (for aligning items along the cross axis). Additional properties like align-self allow you to override alignment for individual items.
4. Flexibility:

Flex items can expand and shrink to fill available space using the flex property. This value can be adjusted to distribute space properly.

5. Ordering:
Flex items can be reordered using the order property. Items with a lower order value appear first.
6. Wrapping:
Flex items can wrap to new rows or columns when they don't fit within the container using the flex-wrap property. The flex-flow property combines flex-direction and flex-wrap in one declaration.

Grid layout

CSS Grid Layout is a powerful two-dimensional layout system in CSS that allows you to create complex grid-based layouts with ease. It is designed to handle both rows and columns, making it an excellent tool for structuring web content. Here is a brief overview:

1. Grid Container and Items:
   - Grid layout is applied to a container element, which becomes a grid container.
   - The child elements of the grid container are called grid items.
2. Placement of Grid Items:
   - Grid items can be explicitly placed using properties like grid-column and grid-row.
   - The grid-area property allows you to assign a name to a grid item and place it by referring to that name.
3. Defining the Grid:
   - You define the grid structure by specifying the number of rows and columns using properties like grid-template-rows and grid-template-columns.
   - Grid template properties can include values like fr (fractional units), px (pixels), or named grid areas.
4. Alignment and Spacing:
   - CSS Grid provides properties like justify-items, align-items, justify-content, and align-content for aligning grid items within the grid.
   - You can also control the gaps between grid cells using the grid-gap property (or row-gap and column-gap).
5. Responsiveness:
   - Grid layout can be made responsive by using media queries and adjusting the grid structure or item placement based on the viewport size.

6. CSS selector

CSS selectors allow you to select and manipulate HTML elements. CSS selectors are used to find HTML elements based on their ID, class, type, attribute, and more.

Types of Selectors

1. **Element Selector**:

   The element selector selects elements based on the element name.

   Syntax:

   ```
   p {
   text-align: center;
   color: red;
   }
   ```

2. **ID Selector:**

   The ID selector uses the ID attribute of an HTML element to select a specific element. An ID should be unique within a page, so the ID selector is used to select a single, unique element.

   To select an element with a specific ID, write a hash character (#) followed by the ID of the element.

   Syntax:

   ```
   #para {
     text-align: center;
     color: red;
   }
   ```

3. **Class Selector:**

   The class selector selects elements with a specific class attribute. To select elements with a specific class, write a period (.) character followed by the class name. Here, all elements with class="center" will be red and center-aligned.

   Syntax:

   ```
   .center {
     text-align: center;
     color: red;
   }
   ```

4. **Grouping Selector:**

    If you have elements with the same style definitions, you can group selectors to minimize code repetition.

    Syntax:

    ```
    h1, h2, p {
      text-align: center;
      color: red;
    }
    ```

In this example, the h1, h2, and p elements will all have their text centered and colored red.

2.1 How CSS fits with an HTML page?

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in HTML. It controls the layout of multiple web pages all at once. By separating content (HTML) from presentation (CSS), CSS allows for more flexibility and control in the specification of web page characteristics.

- **HTML** defines the structure of a web page.
- CSS defines how HTML elements are to be displayed.

2.2 Inline, Internal, and External CSS

CSS can be applied to HTML documents in three ways:

1. **Inline CSS**:
    - Uses the style attribute inside HTML elements.
    - Example:

    ```
    <p style="color: red;">This is a red paragraph.</p>
    ```

2. **Internal CSS**:
    - Defined within the <style> tag inside the <head> section of an HTML page.
    - Example:

    ```
    <head>
      <style>
        p {
          color: blue;
        }
      </style>
    </head>
    <body>
      <p>This is a blue paragraph.</p>
    </body>
    ```

3. **External CSS**:
    - Stored in an external file with a .css extension.
    - Linked to an HTML document using the <link> tag within the <head> section.
    - Example:

```
html

<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <p>This paragraph is styled with an external CSS file.</p>
</body>

Css
P{
color:blue;
}
```

2.3 CSS Selectors

CSS selectors are used to select the HTML elements you want to style. Common selectors include:

- **Element Selector**:
    - Targets elements by their name.
    - Example:

```
p {
  color: green;
}
```

- **ID Selector**:
    - Targets an element based on its id attribute.
    - Example:

```
#uniqueElement {
  color: red;
}
```

- **Class Selector**:
    - Targets elements based on their class attribute.
    - Example:

```
.className {
  color: blue;
}
```

- **Attribute Selector**:
    - Targets elements based on an attribute and its value.
    - Example:

```
[type="text"] {
  border: 1px solid black;
}
```

- **Group Selector**:
  - Targets multiple elements.
  - Example:

```
h1, h2, h3 {
  color: purple;
}
```

2.4 CSS Properties for Text, List, Table, Background, and Link Formatting

- **Text Properties**:
  - color: Sets the color of the text.
  - font-size: Sets the size of the font.
  - font-family: Specifies the font family for the text.
  - text-align: Aligns the text (left, right, center, justify).
- **List Properties**:
  - list-style-type: Specifies the type of list item marker (disc, circle, square, none).
  - list-style-position: Specifies the position of list-item markers (inside, outside).
- **Table Properties**:
  - border: Sets the border of the table and cells.
  - border-collapse: Specifies whether table borders should collapse into a single border or be separated.
  - padding: Sets the padding inside the table cells.
  - text-align: Aligns the text inside the cells.
- **Background Properties**:
  - background-color: Sets the background color.
  - background-image: Sets the background image.
  - background-repeat: Defines if/how a background image will be repeated.
  - background-position: Sets the starting position of a background image.
- **Link Formatting**:
  - a:link: Normal, unvisited link.
  - a:visited: Link the user has visited.
  - a:hover: Link when the user mouses over it.
  - a:active: Link the moment it is clicked.

Example:

```
a:link {
  color: blue;
}
a:visited {
  color: purple;
}
a:hover {
  color: red;
}
a:active {
  color: yellow;
}
```

2.5 Pseudo-classes

Pseudo-classes are used to define the special states of an element. They can style an element when a user interacts with it or based on its position in the document.

- :before: Inserts content before an element.

  ```
  p:before {
    content: "Note: ";
  }
  ```

- :after: Inserts content after an element.

  ```
  p:after {
    content: " End of note.";
  }
  ```

- :first-line: Applies style to the first line of a block-level element.

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

- :first-letter: Applies style to the first letter of a block-level element.

  ```
  p:first-letter {
    font-size: 2em;
    color: red;
  }
  ```

- :hover: Applies style when the user hovers over an element.

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

- :focus: Applies style when an element is focused.

  ```
  input:focus {
    background-color: yellow;
  }
  ```

- :active: Applies style when an element is activated.

  ```
  a:active {
    color: orange;
  }
  ```

2.6 Custom List Numbering using Content Property

You can use the content property along with pseudo-elements to create custom list numbering.

Example:

```
ol {
  list-style: none;
  counter-reset: item;
}
li {
  counter-increment: item;
}
li:before {
  content: vcounter(item) ". ";
  font-weight: bold;
}
```

2.7 CSS Box Model: Margin, Padding, and Border

The CSS box model describes the rectangular boxes generated for elements in the document tree and consists of:

- **Content**: The actual content of the box, where text and images appear.
- **Padding**: Clears an area around the content. It is transparent.

  ```
  padding: 10px;
  ```

- **Border**: A border that goes around the padding (if any) and content.

  ```
  border: 1px solid black;
  ```

- **Margin**: Clears an area outside the border. It is also transparent.

  ```
  margin: 10px;
  ```

2.8 Creating Layouts with Display, Position, and Float Property

- **Display**:
  - block: The element is displayed as a block element.
  - inline: The element is displayed as an inline element.
  - inline-block: The element is formatted as an inline element but can have a width and height.
  - none: The element will not be displayed.
  - Example:

    ```
    .block {
      display: block;
    }
    ```

- **Position**:
  - static: Default value. Elements are positioned according to the normal flow of the document.
  - relative: Positioned relative to its normal position.
  - absolute: Positioned relative to the nearest positioned ancestor.
  - fixed: Positioned relative to the browser window.
  - Example:

```
.relative {
  position: relative;
  top: 10px;
  left: 20px;
}
```

- **Float**:
  - Floats an element to the left or right, allowing text and inline elements to wrap around it.
  - Example:

```
.float-left {
  float: left;
}
.float-right {
  float: right;
}
```

2.9 Fixed and Liquid Design of the Page

- **Fixed Design**:
  - Uses fixed-width layout for the web page.
  - Measurements are typically in pixels.
  - Example:

```
.fixed-container {
  width: 960px;
  margin: 0 auto;
}
```

- **Liquid Design**:
  - Uses relative measurements such as percentages.
  - Example:

```
.liquid-container {
  width: 80%;
  margin: 0 auto;
}
```

Unit 3: Client Side Programming with JavaScript

Java script

JavaScript is an object-oriented programming language that allows the creation of interactive web pages. It offers several advantages to web developers, such as a short development cycle, ease of learning, small script size, platform independence, and more. JavaScript can be easily and quickly used to extend the functionality of HTML pages.

Adding JavaScript to an HTML page is done by inserting the JavaScript code within the <script> tag.

Syntax:

```
<script type="text/javascript">

  // JavaScript code

</script>
```

JavaScript Fundamentals

1. Variables

Variables are used to store values that can be used in other parts of the program.

Rules for Naming Variables: a) Must start with an alphabet (A to Z or a to z), an underscore (_), or a dollar sign ($). b) The remaining characters can be letters, underscores, dollar signs, or digits (0-9). c) Variable names are case-sensitive.

2. Data Types

JavaScript supports several primitive data types:

Number: Consists of integers, floating-point numbers, and the special NaN (Not a Number) value.

Boolean: Consists of logical values true and false. Boolean values are automatically converted into 1 and 0 in numerical expressions.

String: Consists of a sequence of characters enclosed in single or double quotes.

NULL: Represents a null value, indicating the absence of any object value.

3. Operators

An operator is a symbol that is used to perform operations on operands and produce a single resultant value.

Arithmetic Operators: Perform mathematical calculations.

Logical Operators: Used for logical operations. &&.||,!

Comparison Operators: Used to compare values (e.g., == (equal), === (strictly equal), = (not equal), == (strictly not equal)). Note that == and! = perform type conversions before testing for equality, whereas === and! == do not.

String Operator: + (string concatenation). For example, "pq" + "rs" produces "pqrs".

Assignment Operators: Used to assign values to variables (e.g., =, +=, -=, *=, /=, %=, etc.).

Conditional (Ternary) Operator: Takes three operands. The syntax is condition? value1: value2. For example, x = (5 > 7)? 5 + 7: 5 - 7 assigns -2 to x because the condition is false.

4. Special Operators

delete Operator: Used to delete a property of an object or an element of an array. For example, delete myArray [5];

new Operator: Used to create an instance of an object. For example, to create a new array, you use myArray = new Array ();

3.1 How JavaScript Fits into a Web Page

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and much more. It is an essential technology of the web, alongside HTML and CSS.

- **HTML**: Provides the structure of the web page.
- **CSS**: Provides the style and layout of the web page.
- **JavaScript**: Provides the dynamic behavior and interactivity of the web page.

JavaScript can be included in an HTML document in three ways:

1. **Inline JavaScript**:

   ```
   <button onclick="alert('Hello World!')">Click me</button>
   ```

2. **Internal JavaScript**:

   ```
   <head>
     <script>
       function showAlert() {
         alert('Hello World!');
       }
     </script>
   </head>
   <body>
     <button onclick="showAlert()">Click me</button>
   </body>
   ```

3. **External JavaScript**:

```
<head>
  <script src="script.js"></script>
</head>
<body>
  <button onclick="showAlert()">Click me</button>
</body>
```

- In script.js:

```
function showAlert() {
  alert('Hello World!');
}
```

3.2 JavaScript Basics: Variables and Operators

- **Variables**: Containers for storing data values.
    - Declared using var, let, or const.
    - Example:

```
var x = 5;
let y = 6;
const z = 7;
```

- **Operators**: Symbols used to perform operations on variables and values.
    - **Arithmetic Operators**: +, -, *, /, %.
    - **Assignment Operators**: =, +=, -=, *=, /=, %=.
    - **Comparison Operators**: ==, ===, !=, !==, >, <, >=, <=.
    - **Logical Operators**: &&, ||, !.

Example:

```
let a = 10;
let b = 20;
let sum = a + b;  // sum is 30
let isEqual = (a == b);  // isEqual is false
```

3.3 Understanding the Document Object Model (DOM)

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content.

DOM Tree: The hierarchical representation of an HTML document.

Nodes: The building blocks of the DOM tree (element nodes, text nodes, etc.).

The DOM is a programming interface for web documents. It represents the structure of a document as a tree of objects. Each element, attribute, and piece of text in the HTML document is represented as a node in this tree. JavaScript can interact with and manipulate these nodes to change the content, structure, and style of a web page dynamically.

Accessing Nodes: Methods like getElementById(), getElementsByClassName(), getElementsByTagName() are used to access nodes.

What is the DOM?

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a document as a tree of objects. Each node in the tree corresponds to a part of the document, such as an element, attribute, or text.

- **HTML Example**:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample Page</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

In the DOM, this HTML document would be represented as a tree structure:

```
- Document
  - html
    - head
      - title
        - "Sample Page"
    - body
      - h1
        - "Hello, World!"
      - p
        - "This is a paragraph."
```

Core Concepts of the DOM

- **Node**: The basic building block of the DOM tree. Everything in the DOM is a node.
  - **Element Node**: Represents an HTML element (e.g., <body>, <h1>, <p>).
  - **Text Node**: Represents the text content inside an element.
  - **Attribute Node**: Represents an attribute of an element (e.g., class, id).
- **Document Object**: The root node of the DOM tree, representing the entire HTML document.
  - Example:

    ```
    console.log(document);
    ```

- **Element Object**: Represents an element in the HTML document.

  - Example:

```
let header = document.querySelector('h1');
console.log(header);
```

Traversing the DOM

You can navigate through the DOM tree to access elements, attributes, and text nodes.

- **Accessing Child Nodes**:

```
let body = document.body;
let firstChild = body.firstChild;
console.log(firstChild);  // Could be a text node or element node
```

- **Accessing Parent Node**:

```
let paragraph = document.querySelector('p');
let parent = paragraph.parentNode;
console.log(parent);  // Outputs the body element
```

- **Accessing Sibling Nodes**:

```
let header = document.querySelector('h1');
let nextSibling = header.nextSibling;
console.log(nextSibling);  // Could be a text node or element node
```

Manipulating the DOM

JavaScript can be used to dynamically change the content, structure, and style of the document.

- **Changing Content**:

```
let header = document.querySelector('h1');
header.textContent = "New Heading";
```

- **Changing Attributes**:

```
let paragraph = document.querySelector('p');
paragraph.setAttribute('id', 'new-id');
console.log(paragraph.getAttribute('id'));  // Outputs 'new-id'
```

- **Changing Styles**:

```
let paragraph = document.querySelector('p');
paragraph.style.color = 'blue';
paragraph.style.fontSize = '20px';
```

- **Creating and Adding Elements**:

```
let newElement = document.createElement('div');
newElement.textContent = 'This is a new div';
document.body.appendChild(newElement);
```

- **Removing Elements**:

```
let paragraph = document.querySelector('p');
paragraph.remove();
```

Event Handling in the DOM

JavaScript can be used to add event listeners to DOM elements, allowing interaction with the user.

- **Adding Event Listeners**:

```
let button = document.querySelector('button');
button.addEventListener('click', function() {
  alert('Button was clicked!');
});
```

- **Removing Event Listeners**:

```
function showAlert() {
  alert('Button was clicked!');
}

let button = document.querySelector('button');
button.addEventListener('click', showAlert);

// To remove the event listener
button.removeEventListener('click', showAlert);
```

Working with Forms

Manipulating form elements using the DOM is essential for tasks like form validation and submission.

- **Accessing Form Elements**:

```
let form = document.querySelector('form');
let input = form.querySelector('input[name="username"]');
console.log(input.value);
```

- **Form Validation**:

```
let form = document.querySelector('form');
form.addEventListener('submit', function(event) {
  let input = form.querySelector('input[name="username"]');
  if (input.value === '') {
    alert('Username cannot be empty');
  event.preventDefault();  // Prevents form from submitting
 }
});
```

3.4 Accessing HTML Elements

- **getElementById()**:
    - Returns the element with the specified ID.
    - Example:

      ```
      let element = document.getElementById('myElement');
      ```

- **getElementsByClassName()**:
    - Returns a collection of all elements with the specified class name.
    - Example:

      ```
      let elements = document.getElementsByClassName('myClass');
      ```

- **getElementsByName()**:
    - Returns a collection of all elements with the specified name attribute.
    - Example:

      ```
      let elements = document.getElementsByName('myName');
      ```

- **getElementsByTagName()**:
    - Returns a collection of all elements with the specified tag name.
    - Example:

      ```
      let elements = document.getElementsByTagName('p');
      ```

3.5 JavaScript Objects

JavaScript objects are collections of properties, and a property is an association between a name (or key) and a value.

- **window**: Represents the browser's window.

  ```
  console.log(window.innerWidth);  // Outputs the width of the window
  ```

- **document**: Represents the HTML document.

  ```
  console.log(document.title);  // Outputs the title of the document
  ```

- **Array**: Used to store multiple values in a single variable.

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

- **String**: Used to store and manipulate text.

```
let text = "Hello World";
console.log(text.length);  // Outputs the length of the string
```

- **Math**: Provides mathematical constants and functions.

```
console.log(Math.PI);  // Outputs 3.141592653589793
console.log(Math.sqrt(16));  // Outputs 4
```

- **Date**: Provides methods for working with dates and times.

```
let date = new Date();
console.log(date.getFullYear());  // Outputs the current year
```

3.6 Writing Scripts to Handle Events

Events are actions that occur when a user interacts with a web page. JavaScript can handle events such as clicks, form submissions, and keyboard inputs.

- **Event Handlers**:
    - onclick: Executes JavaScript when an element is clicked.
    - onload: Executes JavaScript when a page or an image is fully loaded.
    - onchange: Executes JavaScript when an input element's value is changed.

  Example:

```
<button onclick="handleClick()">Click me</button>
<script>
  function handleClick() {
    alert('Button clicked!');
  }
</script>
```

3.7 Using JavaScript to Validate User Inputs

JavaScript can be used to validate form inputs before they are submitted to the server. This ensures that the data entered by the user is correct and complete.

- **Example of Form Validation**:

```
<form onsubmit="return validateForm()">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Submit">
</form>
<script>
  function validateForm() {
    let name = document.getElementById('name').value;
    if (name == "") {
      alert("Name must be filled out");
      return false;
    }
    return true;
  }
</script>
```

Unit 4: Server Side Programming with PHP

4.1 How PHP Fits into a Web Page

PHP (Hypertext Preprocessor) is a popular server-side scripting language designed specifically for web development. It is embedded within HTML and is used to manage dynamic content, session tracking, databases, and more.

- **HTML**: Provides the structure of the web page.
- **CSS**: Provides the style and layout of the web page.
- **JavaScript**: Provides the dynamic behavior and interactivity of the web page.
- **PHP**: Handles the server-side logic, such as processing form data, managing sessions, and interacting with databases.

PHP code is executed on the server, and the result is returned to the browser as plain HTML.

4.2 Variables and Constants

- **Variables**: In PHP, variables start with a dollar sign ($), followed by the name of the variable.
  - Example:

    ```
    <?php
    $text = "Hello, World!";
    $number = 42;
    echo $text;  // Outputs "Hello, World!"
    ?>
    ```

- **Constants**: Constants are like variables, except that once they are defined, they cannot be changed or undefined. They are defined using the define() function.
  - Example:

    ```
    <?php
    define("SITE_NAME", "My Website");
    echo SITE_NAME;  // Outputs "My Website"
    ?>
    ```

4.3 Operators

PHP supports a wide range of operators, similar to other programming languages.

- **Arithmetic Operators**: +, -, *, /, %
  - Example:

    ```
    <?php
    $x = 10;
    $y = 5;
    $sum = $x + $y;  // $sum is 15
    ?>
    ```

- **Assignment Operators**: =, +=, -=, *=, /=, %=

  - Example:

    ```
    <?php
    $x = 10;
    $x += 5;  // $x is now 15
    ?>
    ```

- **Comparison Operators**: ==, ===, !=, !==, >, <, >=, <=
  - Example:

    ```
    <?php
    $a = 10;
    $b = 20;
    $isEqual = ($a == $b);  // $isEqual is false
    ?>
    ```

- **Logical Operators**: &&, ||, !
  - Example:

    ```
    <?php
    $x = true;
    $y = false;
    $result = $x && $y;  // $result is false
    ?>
    ```

4.4 Working with Text and Numbers

- **String Concatenation**: Use the . operator to concatenate strings.
  - Example:

    ```
    <?php
    $firstName = "John";
    $lastName = "Doe";
    $fullName = $firstName . " " . $lastName;  // $fullName is "John Doe"
    ?>
    ```

- **String Functions**:
  - strlen(): Returns the length of a string.

    ```
    <?php
    echo strlen("Hello, World!");  // Outputs 13
    ?>
    ```

  - strpos(): Finds the position of the first occurrence of a substring in a string.

    ```
    <?php
    echo strpos("Hello, World!", "World");  // Outputs 7
    ?>
    ```

- **Number Functions**:
  - abs(): Returns the absolute value of a number.

    ```
    <?php
    echo abs(-5);  // Outputs 5
    ?>
    ```

  - round(): Rounds a floating-point number to its nearest integer.

    ```
    <?php
    echo round(3.6);  // Outputs 4
    ?>
    ```

4.5 Making Decisions with Control Statements

- **if Statement**: Executes some code if one condition is true.
  - Example:

    ```
    <?php
    $age = 20;
    if ($age >= 18) {
      echo "You are an adult.";
    }
    ?>
    ```

- **if...else Statement**: Executes some code if a condition is true and another code if that condition is false.
  - Example:

    ```
    <?php
    $age = 17;
    if ($age >= 18) {
      echo "You are an adult.";
    } else {
      echo "You are not an adult.";
    }
    ?>
    ```

- **if...elseif...else Statement**: Executes different codes for more than two conditions.
  - Example:

    ```
    <?php
    $score = 85;
    if ($score >= 90) {
      echo "Grade: A";
    } elseif ($score >= 80) {
      echo "Grade: B";
    } else {
      echo "Grade: C";
    }
    ```

```
?>
```

- **switch Statement**: Selects one of many blocks of code to be executed.
    - Example:

```
<?php
$day = "Monday";
switch ($day) {
  case "Monday":
    echo "Start of the work week.";
    break;
  case "Friday":
    echo "End of the work week.";
    break;
  default:
    echo "Midweek days.";
}
?>
```

- **Loop Statements**: Used to execute the same block of code a specified number of times.
    - **while Loop**: Loops through a block of code as long as a specified condition is true.

```
<?php
$x = 1;
while ($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

- **for Loop**: Loops through a block of code a specified number of times.

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

- **foreach Loop**: Loops through a block of code for each element in an array.

```
<?php
$colors = array("red", "green", "blue");
foreach ($colors as $color) {
  echo "$color <br>";
}
?>
```

4.6 Working with Arrays, Strings, DateTime, and Files

- **Arrays**:
  - Indexed Arrays:

    ```
    <?php
    $fruits = array("Apple", "Banana", "Mango");
    echo $fruits[0];  // Outputs "Apple"
    ?>
    ```

  - Associative Arrays:

    ```
    <?php
    $ages = array("Peter" => 35, "Ben" => 37, "Joe" => 43);
    echo $ages['Peter'];  // Outputs 35
    ?>
    ```

  - Multidimensional Arrays:

    ```
    <?php
    $cars = array(
      array("Volvo", 22, 18),
      array("BMW", 15, 13),
      array("Saab", 5, 2),
      array("Land Rover", 17, 15)
    );
    echo $cars[0][0];  // Outputs "Volvo"
    ?>
    ```

- **String Functions**:
  - str_replace(): Replaces some characters in a string with some other characters.

    ```
    <?php
    echo str_replace("world", "Dolly", "Hello world!");  // Outputs "Hello Dolly!"
    ?>
    ```

  - substr(): Returns a part of a string.

    ```
    <?php
    echo substr("Hello world", 0, 5);  // Outputs "Hello"
    ?>
    ```

- **DateTime Functions**:
  - date(): Formats a local date and time.

    ```
    <?php
    echo date("Y/m/d");  // Outputs current date in Y/m/d format
    ?>
    ```

    - strtotime(): Parses an English textual datetime into a Unix timestamp.

      ```
      <?php
      $d = strtotime("tomorrow");
      echo date("Y-m-d", $d);  // Outputs the date for tomorrow
      ?>
      ```

- **File Functions**:
    - fopen(): Opens a file or URL.

      ```
      <?php
      $file = fopen("test.txt", "r");
      ?>
      ```

    - fread(): Reads from an open file.

      ```
      <?php
      $file = fopen("test.txt", "r");
      echo fread($file, filesize("test.txt"));
      fclose($file);
      ?>
      ```

    - fwrite(): Writes to an open file.

      ```
      <?php
      $file = fopen("test.txt", "w");
      fwrite($file, "Hello, World!");
      fclose($file);
      ?>
      ```

4.7 Functions

Functions are blocks of code that can be reused to perform a specific task.

- **Defining a Function**:

  ```
  <?php
  function sayHello() {
    echo "Hello, World!";
  }
  ?>
  ```

- **Calling a Function**:

  ```
  <?php
  sayHello();  // Outputs "Hello, World!"
  ?>
  ```

- **Function with Parameters**:

  ```
  <?php
  function greet($name) {
    echo "Hello, $name!";
  }
  greet("Alice");  // Outputs "Hello, Alice!"
  ?>
  ```

- **Function with Return Value**:

  ```
  <?php
  function add($a, $b) {
    return $a + $b;
  }
  echo add(5, 3);  // Outputs 8
  ?>
  ```

Unit 5: Working with Web Forms

5.1 Creating Simple Web Forms

Web forms are used to collect user input. HTML provides various form elements that allow users to enter data and submit it to the server for processing.

- **HTML Form Syntax**:

  ```
  <form action="submit.php" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <input type="submit" value="Submit">
  </form>
  ```

  - action: Specifies the URL to which the form data will be sent for processing.
  - method: Specifies the HTTP method to be used when sending form data (get or post).
- **Form Elements**:
  - **Text Input**:

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

  - **Password Input**:

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

  - **Radio Buttons**:

    ```
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">Male</label>
    ```

```
<input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label>
```

- **Checkboxes**:

```
<input type="checkbox" id="subscribe" name="subscribe" value="newsletter">
<label for="subscribe">Subscribe to newsletter</label>
```

- **Submit Button**:

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

5.2 Creating Multipage Web Forms

Multipage forms break a long form into multiple steps or pages. This improves user experience by making the form less intimidating.

- **Example of a Multipage Form**:

```
<!-- Page 1 -->
<form action="page2.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Next">
</form>
<!-- Page 2 -->
<form action="submit.php" method="post">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
  <input type="hidden" name="name" value="<?php echo $_POST['name']; ?>">
  <input type="submit" value="Submit">
</form>
```

  - Hidden fields can be used to pass data from one page to another.

5.3 Retrieving Form Data Using POST and GET Methods

- **GET Method**:
  - Appends form data to the URL in name/value pairs.
  - Not suitable for sensitive data because it is visible in the URL.
  - Example:

```
<form action="submit.php" method="get">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Submit">
</form>
```

- Retrieving data:

```
<?php
$name = $_GET['name'];
echo "Name: " . $name;
?>
```

- **POST Method**:
    - Sends form data as an HTTP post transaction.
    - More secure than GET because data is not displayed in the URL.
    - Example:

```
<form action="submit.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <input type="submit" value="Submit">
</form>
```

    - Retrieving data:

```
<?php
$name = $_POST['name'];
echo "Name: " . $name;
?>
```

5.4 Storing Form Data to CSV File

CSV (Comma Separated Values) files are simple text files used to store tabular data. PHP can be used to write form data to a CSV file.

- **Example of Storing Form Data in a CSV File**:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = $_POST['name'];
  $email = $_POST['email'];

  $file = fopen("data.csv", "a");
  fputcsv($file, array($name, $email));
  fclose($file);

  echo "Data stored successfully.";
}
?>
<form action="" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
  <input type="submit" value="Submit">
```

```
</form>
```

5.5 Reading CSV File and Displaying Content as HTML Table

PHP can read data from a CSV file and display it in an HTML table.

- **Example of Reading and Displaying CSV Data**:

```php
<?php
echo "<table border='1'>";
$file = fopen("data.csv", "r");

while (($data = fgetcsv($file)) !== FALSE) {
  echo "<tr>";
  foreach ($data as $field) {
    echo "<td>" . htmlspecialchars($field) . "</td>";
  }
  echo "</tr>";
}

fclose($file);
echo "</table>";
?>
```

Unit 6: Database and PHP

6.1 Introduction to Database

A database is an organized collection of structured data, typically stored and accessed electronically. Databases are essential for storing large amounts of data and enabling efficient retrieval, updating, and management.

- **Relational Database**: A type of database that stores data in tables with rows and columns. Each table is identified by a unique name, and relationships can be established between tables using keys.
- **SQL (Structured Query Language)**: The standard language for relational database management systems (RDBMS) to perform tasks such as querying data, updating records, and managing schema.

6.2 Create, Retrieve, Update, and Delete O peration in Database

These operations, often referred to as CRUD, are the basic operations that can be performed on data in a database.

- **Create**: Inserting new records into a table.
  - Example SQL:

    ```
    INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com');
    ```

- **Retrieve**: Selecting records from a table.
  - Example SQL:

    ```
    SELECT * FROM users WHERE email = 'john@example.com';
    ```

- **Update**: Modifying existing records in a table.
  - Example SQL:

    ```
    UPDATE users SET email = 'john.doe@example.com' WHERE name = 'John Doe';
    ```

- **Delete**: Removing records from a table.
  - Example SQL:

    ```
    DELETE FROM users WHERE name = 'John Doe';
    ```

6.3 Connecting to Database through PHP mysqli_connect()

To interact with a database in PHP, the mysqli extension is commonly used. mysqli_connect() function is used to establish a connection to the database.

- **Example**:

  ```
  <?php
  ```

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "database";

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

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

6.4 Executing Queries with mysqli_query()

mysqli_query() function is used to execute SQL queries on the connected database.

- **Example**:

```
<?php
$sql = "SELECT id, name, email 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["name"]. " - Email: " . $row["email"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

6.5 Fetching Data with mysqli_fetch_assoc() and mysqli_fetch_array()

These functions are used to fetch rows from a result set obtained by mysqli_query().

- mysqli_fetch_assoc(): Fetches a result row as an associative array.
    - Example:

```
<?php
while($row = mysqli_fetch_assoc($result)) {
  echo "id: " . $row["id"]. " - Name: " . $row["name"]. " - Email: " . $row["email"].
"<br>";
}
?>
```

- mysqli_fetch_array(): Fetches a result row as an associative array, a numeric array, or both.
  - Example:

```
<?php
while($row = mysqli_fetch_array($result, MYSQLI_ASSOC)) {
  echo "id: " . $row["id"]. " - Name: " . $row["name"]. " - Email: " . $row["email"].
"<br>";
}
?>
```

6.6 Creating User Registration and Login Feature

To create a user registration and login system, you need to handle user input, store user data securely, and validate user credentials.

- **User Registration**:
  - HTML Form:

```
<form action="register.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
  <label for="password">Password:</label>
  <input type="password" id="password" name="password">
  <input type="submit" value="Register">
</form>
```

  - PHP Code (register.php):

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = $_POST['name'];
  $email = $_POST['email'];
  $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

  $sql = "INSERT INTO users (name, email, password) VALUES ('$name', '$email',
'$password')";

  if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
  } else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
  }

  mysqli_close($conn);
}
?>
```

- **User Login**:
    - HTML Form:

```
<form action="login.php" method="post">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email">
  <label for="password">Password:</label>
  <input type="password" id="password" name="password">
  <input type="submit" value="Login">
</form>
```

    - PHP Code (login.php):

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $email = $_POST['email'];
  $password = $_POST['password'];

  $sql = "SELECT * FROM users WHERE email = '$email'";
  $result = mysqli_query($conn, $sql);

  if (mysqli_num_rows($result) > 0) {
    $row = mysqli_fetch_assoc($result);
    if (password_verify($password, $row['password'])) {
      echo "Login successful";
    } else {
      echo "Invalid password";
    }
  } else {
    echo "No user found with that email";
  }

  mysqli_close($conn);
}
?>
```

6.7 Remembering Users with Cookies and Session

- **Sessions**: Store user information for the duration of their visit. Data is stored on the server.
    - Start a session:

```
<?php
session_start();
$_SESSION["username"] = "JohnDoe";
?>
```

    - Access session data:

```
<?php
session_start();
```

```
echo "Welcome, " . $_SESSION["username"];
?>
```

- **Cookies**: Store user information in the user's browser for a specified period.
  - Set a cookie:

```
<?php
setcookie("username", "JohnDoe", time() + (86400 * 30), "/");  // 86400 = 1 day
?>
```

  - Access a cookie:

```
<?php
if(isset($_COOKIE["username"])) {
  echo "Welcome back, " . $_COOKIE["username"];
} else {
  echo "Welcome, new user!";
  }
?>
```

6.8 Converting Database Table to CSV File Using fputcsv()

PHP can be used to export data from a database to a CSV file.

- **Example**:

```
<?php
$filename = "export.csv";
$file = fopen($filename, "w");

$sql = "SELECT * FROM users";
$result = mysqli_query($conn, $sql);

while ($row = mysqli_fetch_assoc($result)) {
  fputcsv($file, $row);
}

fclose($file);
echo "Data exported to " . $filename;
?>
```

6.9 Reading CSV File and Reflecting the Contents in Database

PHP can read data from a CSV file and insert it into a database table.

- **Example**:

```
<?php
$file = fopen("import.csv", "r");
```

```
while (($data = fgetcsv($file)) !== FALSE) {
  $sql = "INSERT INTO users (name, email) VALUES ('$data[0]', '$data[1]')";
  mysqli_query($conn, $sql);
}

fclose($file);
echo "Data imported successfully.";
?>
```