UNIT- I

HTML Tags and JavaScript

Unit -1 Web Design using HTML

Origins and evolution of HTML :

Web design is the process of creating and designing websites that are displayed on the internet. It encompasses a variety of skills and disciplines, including graphic design, user interface design, coding, and content creation.

HTML (HyperText Markup Language) is the language used to create and design web pages. It was first introduced in **1990** by **Tim Berners-Lee**, a computer scientist at CERN (European Organization for Nuclear Research).

The first version of HTML was very basic and included only a few elements such as headings, paragraphs, and links. It was designed to be a simple way to create and share scientific documents between researchers.

Over the years, HTML has evolved and expanded to include new features and capabilities.

- **HTML 2.0** was released in **1995** and added support for tables, images, and forms.
- **HTML 3.2** was released in **1997** and introduced support for style sheets, which allowed web designers to specify the layout and appearance of web pages separately from the content.

- **HTML 4.01** was released in **1999** and included many new features, such as support for frames, multimedia, and scripting. It was the most widely used version of HTML for many years and remained in use until HTML5 was released.
- **HTML5** was introduced in **2014** and is the latest version of HTML. It includes many new features and improvements, such as new semantic elements, improved support for multimedia, and new form controls. HTML5 also includes support for mobile devices, making it easier to create websites that are optimized for smartphones and tablets.

In addition to HTML, web designers also use **CSS** (Cascading Style Sheets) and **JavaScript** to create and design websites. CSS is used to control the layout, typography, and color of web pages, while JavaScript is used to create interactive and dynamic web content.

Overall, the evolution of HTML has played a significant role in the development of web design. It has enabled web designers to create more complex and interactive websites, and has contributed to the growth and evolution of the internet as a whole.

Basic syntax of HTML :

The basic syntax of HTML (Hypertext Markup Language) consists of a set of tags, which are enclosed in angle brackets (< >) and define the structure and content of a web page. The tags are used to create elements such as headings, paragraphs, links, images, forms, and more.

Here's an example of basic HTML syntax:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Each elements / tags in the previous example :

- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

An HTML element is defined by a start tag, some content, and an end tag:
Eg: <tagname> Content goes here... </tagname>

The HTML element is everything from the start tag to the end tag:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

Basic text markup :

Comment: Used to add comments to the HTML code that are not displayed in the web page.
Syntax: <!-- This is a comment -->

Conditional Comment: A special type of comment used to target specific versions of Internet Explorer.
Syntax: <!--[if IE]> This is a conditional comment <![endif]-->

Document Type Declaration:
A declaration at the beginning of an HTML document that specifies the version of HTML being used.
Syntax: <!DOCTYPE html>

Anchor Tag: Used to create hyperlinks to other web pages or resources.
Syntax: <a href="https://www.example.com">Link Text</a>

Article Tag: Used to define an independent, self-contained content.
Syntax: <article>Content</article>

Aside Tag: Used to define a section of content that is not the main content of the page.
Syntax: <aside>Content</aside>

Bold Tag: Used to make text bold.
Syntax: <b>Bold Text</b>

Body Tag: Used to define the main content of a web page.
Syntax: <body>Content</body>

Line Break Tag: Used to insert a line break.
Syntax: <br>

Form Button Tag: Used to create a button within a form.
Syntax: <button>Button Text</button>

Image Tag : `<img>` tag is used to embed an image in an HTML page. Below are the mainly used attributes of <img> tags

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed
- height - Specifies height of the image
- width - Specifies width of the image

Center Tag: Used to center content on a web page.
Syntax: <center>Centered Content</center>

Div Tag: Used to group and format content.
Syntax: <div>Content</div>

Definition List (**dl**), Definition Term (**dt**), and Definition Description (**dd**) Tags: Used to create a list of terms and definitions.

Syntax of <dl> <dt> <dd> :

```
<dl>
        <dt>Term</dt>
        <dd>Definition of the term</dd>
</dl>
```

Emphasis Tag: Used to emphasize text.
Syntax: <em>Emphasized Text</em>

Fieldset Tag: Used to group related form elements.
The legend tag is used to define a caption for a fieldset element.

Syntax:

```
<fieldset>
        <legend>Fieldset Title</legend>
        Form Elements
</fieldset>
```

Figure Tag: Used to group and format content such as images and captions.

Syntax:

```
<figure>
        <img src="image.jpg" alt="Image">
        <figcaption>Image Caption</figcaption>
</figure>
```

Font Tag: Used to set the font size, color, and family of text.
Note: this tag is deprecated and removed from HTML5.

Syntax: `<font size="3" color="red" face="Arial">Text</font>`

Footer Tag: Used to define the footer of a web page.

Syntax:
```
<footer>Content</footer>
```

Form Tag: Used to create a form on a web page.

Syntax:
```
<form action="process-form.php" method="POST">
        Form Elements
</form>
```

Headings (h1 - h6) Tags: Used to create headings of various sizes.

Syntax:
```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

head tag: The head tag is used to define the header section of an HTML document, which contains metadata such as the title of the document, links to stylesheets and scripts, and other information that is not displayed directly in the browser.

Title tag: The title tag is used to define the title of an HTML document.
Syntax :

```
<head>
    <title>Document Title</title>
    <link rel="stylesheet" href="style.css">
    <script src="script.js"></script>
</head>
```

header tag: The header tag is used to define a header section of a document or section of a page.
nav tag: The nav tag is used to define a section of a document that contains navigation links.
Syntax:

```
<header>
    <h1>Page Title</h1>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
</header>
```

html tag: The html tag is used to define an HTML document.

italic tag: The italic tag is used to make the text italicized.

Syntax:
```
<i>Italic Text</i>
```

iframe tag: The iframe tag is used to embed another HTML document inside the current document.

Syntax:
```
<iframe src="https://www.example.com"></iframe>
```

input tag: The input tag is used to create various form controls such as text input fields, checkboxes, radio buttons, and more.

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

label tag: The label tag is used to associate a label with an input element.

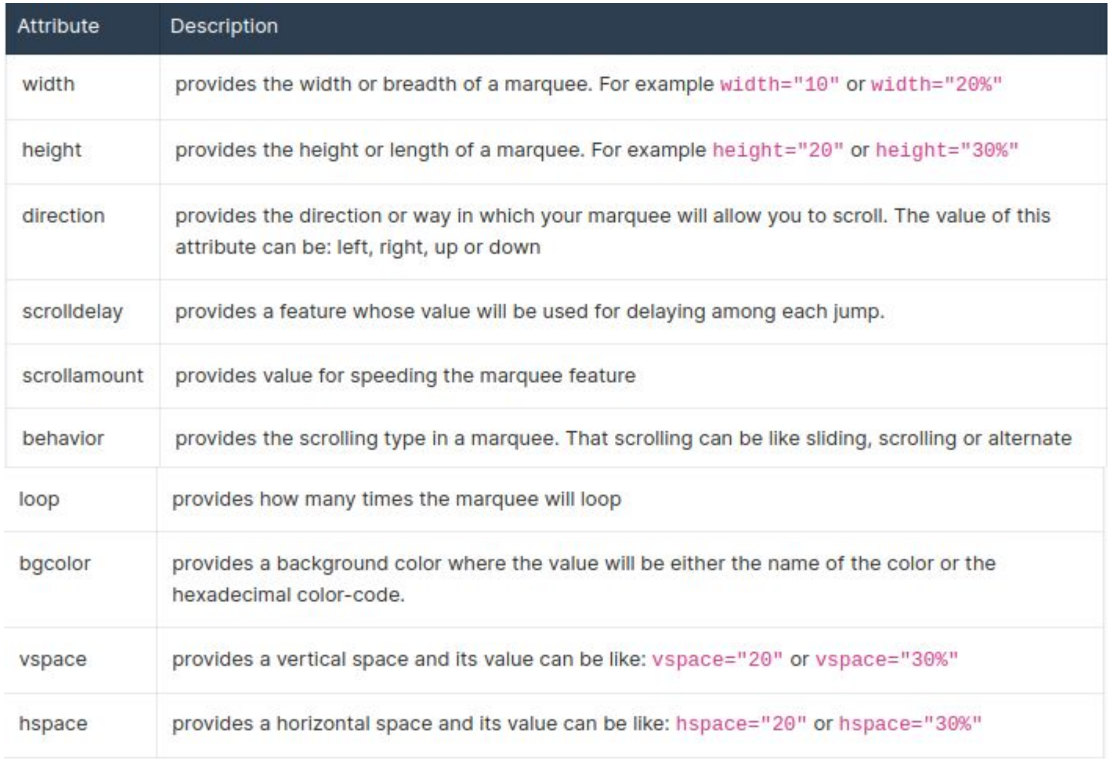
Syntax:
```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

link tag: The link tag is used to link an HTML document to an external resource such as a stylesheet or script.

Syntax :
```
<link rel="stylesheet" href="style.css">
```

The `Marquee Tag`

The `<marquee>` tag is a container tag of HTML is implemented for creating scrollable text or images within a web page from either left to right or vice versa, or top to bottom or vice versa. But this tag has been deprecated in HTML5.
The attributes of `<marquee>` tag are:



Attribute	Description
width	provides the width or breadth of a marquee. For example `width="10"` or `width="20%"`
height	provides the height or length of a marquee. For example `height="20"` or `height="30%"`
direction	provides the direction or way in which your marquee will allow you to scroll. The value of this attribute can be: left, right, up or down
scrolldelay	provides a feature whose value will be used for delaying among each jump.
scrollamount	provides value for speeding the marquee feature
behavior	provides the scrolling type in a marquee. That scrolling can be like sliding, scrolling or alternate
loop	provides how many times the marquee will loop
bgcolor	provides a background color where the value will be either the name of the color or the hexadecimal color-code.
vspace	provides a vertical space and its value can be like: `vspace="20"` or `vspace="30%"`
hspace	provides a horizontal space and its value can be like: `hspace="20"` or `hspace="30%"`

Lists in HTML :

1) Unordered HTML List

An unordered list starts with the `<ul>` tag. Each list item starts with the `<li>` tag. The list items will be marked with **bullets** (small black **circles**) by default:

```
<ul style="list-style-type:disc;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

The CSS **`list-style-type`** property is used to define the style of the list item marker. Or type attribute can be used with below values :

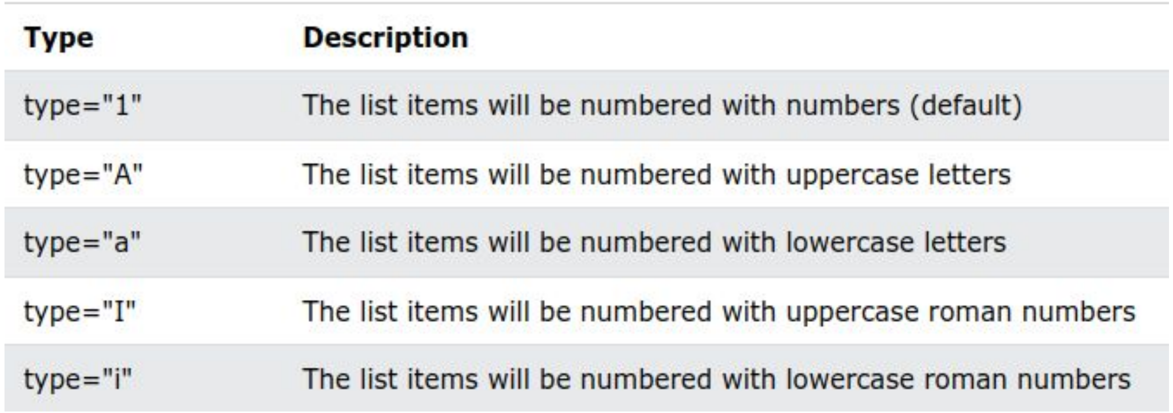
Value	**Description**
disc	Sets the list item marker to a bullet (default)
circle	Sets the list item marker to a circle
square	Sets the list item marker to a square
none	The list items will not be marked

2) Ordered HTML List

An ordered list starts with the `<ol>` tag. Each list item starts with the `<li>` tag. The list items will be marked with numbers by default:

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

The `type` attribute of the `<ol>` tag, defines the type of the list item marker:



Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Paragraph (<p>) tag: The paragraph tag is used to create a new paragraph.

```
Syntax :    <p>This is a new paragraph.</p>
```

Script tag: The script tag is used to define a client-side script, such as JavaScript.

```
Syntax :    <script>
            // JavaScript code goes here
            </script>
```

Section tag: The section tag is used to define a section of a document, such as a chapter, header, or footer.

```
Syntax :    <section>
                <h2>Section Title</h2>
                <p>Section content goes here.</p>
            </section>
```

Select tag: The select tag is used to create a dropdown list of options.

```
Syntax :    <select>
                <option value="option1">Option 1</option>
                <option value="option2">Option 2</option>
                <option value="option3">Option 3</option>
            </select>
```

Span tag: The span tag is used to group inline elements, such as text or images, for styling purposes.

Syntax : `<span>This is some text.</span>`

Style tag: The style tag is used to define style information for an HTML document.

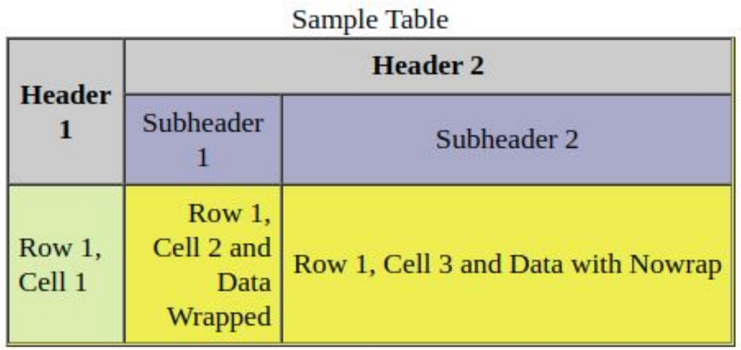
Syntax : `<style> /* CSS code goes here */ </style>`

Table tag and related tags: The table tag is used to create a table in HTML, and the related tags are used to define the structure and content of the table. Some tags and attributes of table as follows :

Define Table	`<table>..... </table>`	Add Caption	`<caption>.....</caption>`
Table row	`<tr>...........</tr>`	Table header	`<th>.............</th>`
Table Data in a cell	`<td>..........</td>`	Cell spacing	`<table cellspacing="2px">...</table>`
Cell padding	`<table cellpadding="2px">...</table>`	Table border	`<table border="1">...</table>`
Cell color	`<table bgcolor="#fff">...</table>`	Width of the tabel	`<table width=”60%”>..</table>`
Align content	`<tr align=”center”>...</tr>`	No line breaks	`<td nowrap>...</td>`
Column span in td	`<td colspan="2">...</td>`	Row span in td	`<td rowspan="2">...</td>`

Sample Example of Table

```
<!DOCTYPE html>
<html>
<body>
<table border="1"  cellpadding="5"  cellspacing="0"   width="50%"   bgcolor="#ee3">
        <caption>Sample Table</caption>
        <tr  align="center"  bgcolor="#ccc">
                <th  rowspan="2">Header 1</th>
                <th  colspan="2" >Header 2</th>
        </tr>
        <tr align="center" bgcolor="#aac">
                <td>Subheader 1</td>
                <td>Subheader 2</td>
        </tr>
        <tr>
                <td bgcolor="#dea">Row 1, Cell 1</td>
                <td align="right" >Row 1, Cell 2 and Data Wrapped</td>
                <td nowrap >Row 1, Cell 3 and Data with Nowrap</td>
        </tr>
</table>
</body>
</html>
```


Sample Table

Header 1	Header 2	
	Subheader 1	Subheader 2
Row 1, Cell 1	Row 1, Cell 2 and Data Wrapped	Row 1, Cell 3 and Data with Nowrap

Time tag: The time tag is used to define a date or time.

Syntax : `<time datetime="2023-04-29">April 29th, 2023</time>`

The <audio> and <video> tags in HTML are used to embed multimedia content, such as audio or video files, into a web page. Here's the syntax for each tag:

The <audio> tag: `<audio src="audio_file.mp3" controls></audio>`

● **src** attribute	specifies the URL of the audio file to be played.
● **controls** attribute	adds a set of playback controls (play, pause, volume, etc.) to the audio player.

The <video> tag: `<video src="video_file.mp4" controls width="640" height="480"></video>`

● **src** attribute	specifies the URL of the video file to be played.
● **controls** attribute	adds a set of playback controls (play, pause, volume, etc.) to the video player.
● **width** attribute	specifies the width of the video player in pixels.
● **height** attribute	specifies the height of the video player in pixels.

Both tags also support several other attributes for controlling playback, such as autoplay, loop, muted, and preload.

Frames

HTML allows programmers to divide a single browser display into multiple window sections, where each section can load individual URLs. This concept of HTML providing multiple frames at one browser display is called **frameset**, and all the **<frame>** tags are used within the container tag **`<frameset>`**. So the entire separation of HTML pages is possible using the concept of frames.

Attributes of `<frameset>`:

- **cols:** Specifies the width of each frame in a horizontally divided frameset. You can specify multiple values separated by commas to divide the space among multiple frames.
- **rows:** similar to "cols", but used for vertically divided framesets.
- **order:** specifies the width of the border around each frame in pixels.
- **framespacing:** specifies the space between each frame in pixels.
- **frameborder:** specifies whether or not each frame should display a border. This can be set to "1" (to display the border) or "0" (to hide the border).
- **marginwidth and marginheight:** specify the width and height of the margin around each frame in pixels.

Attributes of <frame> :

- **src**: is implemented for fetching the HTML file that needs to be loaded in one of the frames. It takes the value as filename.html or filename.htm within double-quotes.
- **name**: facilitates you in giving a name to your frame, and hence you can indicate which frame(s) you are supposed to load into your page.
- **frameborder**: is used for specifying if the borders are being shown in the frame you are using, and you can assign values either: 1 (yes) or 0 (no) for it.
- **marginwidth**: facilitates specifying the frame borders width spacing on the left and right sides. It takes the value in pixels.
- **marginheight**: facilitates specifying the frame borders height spacing on top and bottom sides. It also takes the value in pixels.
- **noresize**: It is generally possible to resize your frame by clicking and dragging the frame borders. But this attribute helps users stop resizing the frames. It is written something like: `noresize="noresize"`.
- **scrolling**: is used for activating and deactivating the scroll-bar appearance in your frame and takes either yes, no, or auto as values to be assigned to it within double-quotes.

Example of Frameset to divide the page into different sections:

```
<!DOCTYPE html>
<html>
<head>
<title>Frame Demo</title>
</head>
<frameset  rows="15,85"  frameborder="yes" border="5px"  bordercolor="red">
        <frame  src="row1.html"  name="row1" scrolling="no" noresize="noresize" />
        <frameset cols="30,70">
                <frame src="col1.html" name="col1" scrolling="no" />
                <frame src="col2.html" name="col2" />
        </frameset>
</frameset>
</html>
```

Overview and features of HTML5

HTML5 is the latest version of the Hypertext Markup Language, which is used to create content on the web. It was released in 2014 and introduced many new features and improvements over previous versions . Some of the key features of HTML5 include:

1.Improved Semantics:

HTML5 introduces many new semantic elements that make it easier to create web pages that are meaningful and easier to understand for both humans and machines. Some of these elements include:

<header>	: Defines a header for a section or page
<footer>	: Defines a footer for a section or page
<nav>	: Defines a set of navigation links
<article>	: Defines an independent, self-contained piece of content
<section>	: Defines a section within a document

Using these new elements makes it easier for search engines and other tools to understand the structure and meaning of your content, which can lead to better search engine rankings and a better user experience.

2.Multimedia Support:

HTML5 includes native support for audio and video playback, without the need for third-party plugins like Flash. This makes it much easier to embed multimedia content on your web pages, and also provides better performance and compatibility across different devices and platforms.
The **<audio>** and **<video>** elements allow you to embed audio and video content directly into your web pages, with support for a wide range of formats and codecs. You can also control playback using JavaScript, and customize the player with CSS.

3.Improved Forms:

HTML5 introduces many new input types and attributes that make it easier to create and validate forms. Some of these new input types include:

<input type="email">	: Validates that the input is a valid email address
<input type="date">	: Allows users to select a date from a calendar widget
<input type="time">	: Allows users to select a time from a dropdown menu
<input type="search">	: Provides a search input field with a clear button
<input type="color">	: Provides a a color picker widget that allows the user to select a color
<input type="number">	: Allows users to enter only numeric value

HTML5 also introduces several new attributes that can be used to customize the behavior of form elements. Some of these new attributes include "required", "placeholder", "min", "max", and "pattern".

4.Canvas and SVG:

HTML5 includes support for two new graphics elements: <canvas> and <svg>.
The <canvas> element allows you to create dynamic, scriptable rendering of 2D graphics using JavaScript.
This can be used for things like creating charts and graphs, drawing animations, or creating games.

The <svg> element allows you to create vector graphics using XML markup. (i.e., visual images are created directly from geometric shapes such as points, lines, curves and polygons) This allows for crisp, scalable graphics that can be easily styled with CSS.

5.Offline Support:
HTML5 introduces the ability to store web application data offline, allowing for the creation of web apps that can be used even without an internet connection. This is achieved using the "localStorage" and "sessionStorage" APIs, which allow you to store data locally in the user's browser.

You can use this feature to create web apps that are more like traditional desktop or mobile apps, with fast performance and offline capabilities.

Overall, these features and many others make HTML5 a powerful and flexible tool for creating rich, interactive web content.

JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an ***interpreted programming language*** with ***object-oriented capabilitie***s.

JavaScript was first known as **LiveScript**, but *Netscape* changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in **1995** with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Client-Side JavaScript

- Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an ***HTML document*** for the code to be interpreted by the browser.
- It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
- The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.
- The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.
- JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

For example, here is a simple JavaScript object that represents a person:

```
let person = {
            name: "John",
            age: 30,
            occupation: "Developer"
            };
```

In this example, the *person* object has three properties: *name, age*, and *occupation*. These properties are accessed using dot notation, like so:

```
console.log(person.name);   // outputs "John"
console.log(person.age);    // outputs 30
```

<u>How to add JavaScript to html :</u>

JavaScript, also known as **JS**. The term "**script**" is used to refer to the languages that are not standalone in nature and here it refers to JavaScript which run on the client machine.The term scripting is used for languages that require the support of another language to get executed. For example, JavaScript programs cannot get executed without the help of ***HTML*** or without integrated into HTML code.

JavaScript is used in several ways in web pages such as generate warning messages, build image galleries, DOM manipulation, form validation, and more.

There are **three ways** in which users can add JavaScript to HTML pages.

1. **Embedding code**
2. **Inline code**
3. **External file**

I. Embedding code:-

To add the JavaScript code into the HTML pages, we can use the **<script>.....</script>** tag of the HTML that wrap around JavaScript code inside the HTML program. Users can also define JavaScript code in the <body> tag (or we can say body section) or <head> tag because it completely depends on the structure of the web page that the users use.

```
<!DOCTYPE html >
<html>
<head>
        <title> page title</title>
        <script>
        document.write("Welcome to Javatpoint");
        </script>
</head>
<body>
        <p>Inthis example we saw how to add JavaScript in the head section </p>
</body>
</html>
```

II. Inline code:-

Generally, this method is used when we have to call a function in the HTML event attributes. There are many cases (or events) in which we have to add JavaScript code directly eg., OnMouseOver event, OnClick, etc. Here we can add JavaScript directly in the html ***without using the <script>.... </script> tag***.

(Eg: **inline javascript**)

```
<!DOCTYPE html >
<html>
<head>
        <title> page title</title>
</head>
<body>
        <p>
                <a href="#" onClick="alert('Welcome !');">Click Me</a>
        </p>
        <p> in this example we saw how to use inline JavaScript or directly in an HTML tag. </p>
</body>
</html>
```

III. External file:-

We can also create a separate file to hold the code of JavaScript with the **(.js) extension** and later incorporate/include it into our HTML document using the **src** attribute of the <script> tag. It becomes very helpful if we want to use the same code in multiple HTML documents. It also saves us from the task of writing the same code over and over again and makes it easier to maintain web pages.

I.e., : `<script type="text/javascript" src="Sample.js"></script>`

Here "**type**" attribute Specifies the media type of the script , which is optional and "**src**" Specifies the URL of an external script file

(Eg : external js)

```
<html>
<head>
        <title>Including a External JavaScript File</title>
</head>
<body>
        <form>
                <input type="button" value="Result" onclick="display()"/>
        </form>
        <script src="hello.js"></script>
</body>
</html>
```

And the external **hello.js** file will contain :

```
function display()
{
        alert("Hello World!");
}
```

The external javascript file will not contain <script> tag, as similar to external css file which will not contain <style> tag. The file extension itself specifies which type of file it is.

General syntactic characteristics of Javascript :

The General Syntactic Characteristics of JavaScript refer to the way that the language is structured and how code is written. Here are some of the main syntactic characteristics of JavaScript:

1. Statements: JavaScript code is made up of statements, which are instructions that tell the computer what to do. A statement typically ***ends with a semicolon (;)***.

For example: `let x = 5;` is a statement that declares a variable named x and assigns it the value 5.

2. Comments: Comments are used to add notes to code that explain what it does or how it works. JavaScript supports both ***single-line comments***, which begin with ***//***, and **multi-line comments**, which are enclosed in ***/* and */***.

For example:

```
//  This is a single-line comment
/*  This is a multi-line comment
    that spans multiple lines       */
```

3. Variables: JavaScript uses variables to store data values. Variables can be declared using the **let, const,** or **var** keywords.

For example:

```
let     x  =  5;         // declares a variable named x and assigns it the value 5
const y  =  "hello";     // declares a constant named y and assigns it the value "hello"
var     z  =  true;      // declares a variable named z and assigns it the value true
```

4. Functions: JavaScript uses functions to group statements together and perform a specific task. Functions are declared using the ***function*** keyword and can be called later in the code.

For example:

```
function addNumbers(x, y) {
        return x + y;
}

let sum = addNumbers(2, 3);
// calls the addNumbers function with arguments 2 and 3 and assigns the result to the sum variable
```

5.Conditional Statements: JavaScript uses conditional statements to execute different code based on different conditions. The most common conditional statement is the if statement.

For example:

```
let x = 5;
if (x < 10) {
        console.log("x is less than 10");
} else {
        console.log("x is greater than or equal to 10");
}
```

6. Loops: JavaScript uses loops to execute the same code multiple times. The most common loop statements are for **loops** and **while loops , do while loops**.

For example:

```
for (let i = 0; i < 5; i++) {
        console.log(i);
}
```

```
let i = 0;
while (i < 5) {
        console.log(i);
         I++;   }
```

7.Arrays: JavaScript uses arrays to store collections of data. An array is declared using square brackets and can contain any type of data.

For example:

```
let numbers = [1, 2, 3, 4, 5];
let names = ["John", "Jane", "Bob"];
```

Arrays can be accessed and modified using index notation, like so:

```
console.log(numbers[0]);        // prints the first element of the numbers array (1)
numbers[3] = 10;                // changes the fourth element of the numbers array to 10
```

8. Objects: JavaScript uses objects to store collections of key-value pairs. An object is declared using ***curly braces*** and can contain any type of data.

For example:

```
let person = {
        name: "John",
        age: 30,
        occupation: "Developer"
};
```

Objects can be accessed and modified using **dot notation** or **bracket notation**, like so:

```
console.log(person.name);          // prints the value of the name property ("John")
person.age = 35;                   // changes the value of the age property to 35
```

9. Classes: JavaScript supports object-oriented programming using classes. A class is a blueprint for creating objects, and it can contain properties and methods.

For example:

```
class Person {

        constructor(name, age) {
                this.name = name;
                this.age = age;
        }

        greet() {
                console.log(`Hello, ${this.name}  is ${this.age} year old` );
        }
}

let john = new Person("John", 30);
john.greet();                      // prints "Hello, John is 30 year old" to the console
```

In this example, the Person class is used to create a new object named john. The greet method is called on the john object to print a greeting to the console.

Javascript Primitives, Operations, and Expressions

In JavaScript, ***a primitive is a basic data type*** that represents a single value. There are six primitive data types:

1.Number: The number primitive represents numeric data, including integers and floating-point numbers. Numbers can be positive, negative, or zero. Here are some examples:

```
const integer = 5;
const floatingPoint = 3.14;
```

2.String: The string primitive represents textual data. Strings are enclosed in either single quotes ('...') or double quotes ("..."). Here are some examples:

```
const greeting = "Hello, world!";
const name = 'John Doe';
```

3.Boolean: The boolean primitive represents a logical value that can either be true or false.
Here are some examples:

```
const isRaining = true;
const hasCar = false;
```

4.Null: The null primitive represents a deliberate non-value. It is often used to indicate the absence of a value. It is not a space, nor is it a zero; it is simply nothing. Here is an example:

```
const user = null;
```

5.Undefined: The undefined primitive represents a variable or property that has been declared but has not been assigned a value. It can also represent a non-existent property or variable. Here is an example:

```
let variable;
console.log(variable);    // logs undefined
```

6.Symbol: The symbol primitive represents a ***unique identifier***. Symbols are often used as property keys in objects.
Here is an example:

```
const sym1 = Symbol('foo');
const sym2 = Symbol('foo');
console.log(sym1 === sym2); // logs false
```

<u>Operators: In JavaScript</u> <u>:</u> Operators are symbols that perform operations on values. There are several types of operators, including:

- **Arithmetic operators:** perform mathematical operations on numbers.
  For example: + (addition), - (subtraction), * (multiplication), / (division), % (modulus).
- **Comparison operators:** used to compare two values, and return a boolean value.
  For example: == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to).
- **Logical operators:** used to compare two conditional statements (or to operate on one statement) to determine if the result is true and to proceed accordingly. They use symbols such as :
  && (returns true if the statements on both sides of the operator are true)
  || (returns true if a statement on either side of the operator is true).
- **Bitwise operators:** These are logical operators that work at the bit level (ones and zeros). They use
  symbols like << (for left-shifting bits) and >> (for right-shifting bits).
- **Assignment operators:** These operators are used to assign new values to variables. Some assignment operators like = , += ,-= etc
- **Unary operators:** operate on a single operand. For example: ++ (increment), -- (decrement).

In JavaScript, an expression is any valid unit of code that can be evaluated to a value. An expression can consist of one or more operands (values or variables) and operators (symbols that perform an operation). Here are some examples of expressions:

```
5 + 3 /                 //  evaluates to 8
"Hello, " + "world!"    //  evaluates to "Hello, world!"
x = 10                  // assigns the value 10 to the variable x and evaluates to 10
```

Functions in Javascript :

A function is basically a little script within a larger script. Its purpose is to perform a single task or a series of tasks.Functions help organize the various parts of a script into the different tasks that must be accomplished.Functions can be used more than once within a script to perform their task. Rather than rewriting the entire block of code, you can simply call the function again.

A function in JavaScript can be defined using the *function* keyword, followed by the function name, a list of parameters (optional), and the function body enclosed in curly braces.

For example:

```
function greet(name) {
  console.log("Hello, " + name + "!");
}
```

This function takes one parameter, name, and logs a greeting message to the console. To call this function, you simply need to pass a value for the name parameter, like so:

```
greet("John"); // Output: "Hello, John!"
```

Naming Functions :
As with variables, functions need to be named carefully to avoid problems with your scripts. The same basic rules that applied to variables apply to the naming of functions: ***case sensitivity, using allowed characters, avoiding reserved words, and giving functions memorable and meaningful names.***

<u>Anonymous Functions :</u>
An anonymous function is a function that *doesn't have a name.* It can be defined using the *function* keyword, but without a function name. Anonymous functions are often used *as callback functions or as immediate function* invocations.

Sample Callback function :

```
let numbers = [1, 2, 3, 4, 5];
numbers.forEach (  function(number)  {
        console.log(number);
});
```

Immediate function invocation :

```
( function() {
        alert("Hello, World!");
}) ();
```

Anonymous functions are quite useful when dealing with JavaScript events. For example, to react to a user clicking the mouse while on a Web page, you could write a simple function for a click event on the document and then call it, as in the following code:

```
document.onclick = function() {
        alert("Do not click on my page!");
};
```

Event Handlers in Java Scripts

An event handler is a predefined JavaScript property of an object that is used to handle an event on a Web page.An event is something that happens when the viewer of the page performs some sort of action, such as clicking a mouse button, clicking a button on the page, changing the contents of a form element, or moving the mouse over a link on the page. Events can also occur simply by the page loading or other similar actions.

Event handlers can be used in a number of locations. They can be used directly within HTML elements by adding special attributes to those elements. They can also be used within the <script> and </script> tags or in an external JavaScript file.
For Example :

```
<input type="button"  id=”b1”  value="Click Me!" onclick="alert(‘You clicked Me’);" />
```

Using the same in script tag :

```
<input  type="button"  id=”b1”   value="Click Me!" onclick="msg()" />
<script> function msg(){
                alert(“You clicked Me”); }
</script>
```

Other Ways to Register Events inside script tag is :-

- **addEventListener() Method :**
  Syntax: element.addEventListener('event_type', function_name, true_or_false);
  Eg: b1.addEventListener('click' , function() { alert("Clicked"); });
- **attachEvent() Method :**
  Syntax: element.attachEvent(event_handler, function_name);
  Eg: b1.attachEvent(onclick, function() { alert("Clicked");   });

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Screen Outputs and Keyboard Inputs in Javascript :

There are several ways to perform screen output, including:

- Using console.log() , document.write()
- Manipulating the HTML DOM
- Displaying alerts and confirmations

Using console.log(), document.write()

console.log() is a built-in JavaScript method that allows you to output text to the console, which can be viewed in the developer tools of most modern browsers. Where as document.write will be printed on the Html page . Here's an example:

```
console.log("Hello, world!");      // will be printed on the console
document.write("Hello Script");  // will be printed on the HTML Page
```

Manipulating the HTML DOM

In JavaScript, you can use the **Document Object Model (DOM)** to manipulate the content of an HTML page. You can create new elements, change existing elements, and add or remove elements from the page. Here's an example:

```
<!DOCTYPE html>
<html>
<head>  <title>DOM Manipulation Example</title> </head>
<body>
  <h1 id="myHeading">Welcome to my website!</h1>
  <button onclick="changeText()">  Click me to change the heading text </button>
  <script>
  function changeText() {
          let heading = document.getElementById("myHeading");
          heading.innerHTML = "Hello, world!";
  }
  </script>
</body>
</html>
```

Displaying alerts and confirmations

You can also use the **alert()** and **confirm()** methods to display messages to the user in a dialog box.
Here's an example: alert("Hello, world!"); // a dialog box with the message "Hello, world!".

The **confirm()** method displays a dialog box with a message and two buttons, typically ***"OK" and "Cancel".***
The method returns ***true*** if the user clicks "OK" and ***false*** if the user clicks "Cancel".
Here's an example:

```
let result = confirm("Are you sure you want to delete this item?");
if (result) {
        // Perform the deletion
}
```

Keyboard Inputs : We can detect keyboard inputs using the $keydown$ and $keyup$ events. These events are fired when a key on the keyboard is pressed down and released, respectively. You can attach event listeners to these events using the addEventListener method.

Here's an example of detecting a keyboard input using the keydown event:

```
document.addEventListener("keydown", function(event) {
        alert("Key pressed: " + event.key);
});
```

The prompt method is a built-in method in JavaScript that displays a dialog box that prompts the user to enter some input. The method takes a string argument that represents the message to display to the user.

Eg:
```
<script>
        let name = prompt("Enter your name:");
        alert("Hello, " + name );
</script>
```

Some Useful Javascript methods:

charAt(index)	It provides the char value present at the specified index.
indexOf('char')	It provides the position of a char value present in the given string.
substring(start, end)	It returns the extracted part in a new string from start index to end index
substr(start, length)	It is used to fetch the part of the given string on the basis of the specified starting position and length.
toLowerCase()	It converts the given string into lowercase letter.
trim()	It trims the white space from the left and right side of the string.