

# **Web Technology Notes**

## **HTML NOTES**

=====

### **1)WHAT IS WORLD WIDE WEB(WWW)?**

world wide web commonly referred as www/w3/web. It is a system interconnected public web pages accessible through internet.

-->If we want to create a web we need at least 2 components i.e, client computer and a web server.

-->with the help of internet only we are able to connect to the server, without it won't be possible.

### **INTERNET:(Interconnected Network)**

=====

The internet is a global network of interconnected computer networks that use a standardized set of protocols (such as TCP/IP) to facilitate the exchange of data and information. It is a vast infrastructure that connects millions of devices including computers, servers, routers and other network-enabled devices across the world.

**TCP:** Transmission control protocol

**IP:** Internet protocol

(or)

The internet enables communication, information sharing and access to various online services and resources. It provides a platform for individuals, organizations and governments to connect, collaborate and exchange information regardless of geographical boundaries.

### **What is Web Browser?**

A web browser is a software application that allows users to access, view and interact with others on the world wide web. It serves as a user interface for navigating and displaying web pages, websites and web-based applications and also it helps to render the HTML, CSS, JavaScript etc.

### **what is web server?**

A web server is a software and hardware that uses http and other protocols to respond to client requests made over the www.

**http:** hyper text transfer protocol

**url:** uniform resource locator

### **What is website?**

website is the collection of web pages different multimedia content such as text images and videos which can be accessed by the url, which you can see in the address bar of the browser.

for example: <https://www.google.com>

### **How to access websites?**

when we type a certain url in a browser search bar, the browser requests the page from the webserver and the web server returns the required web page and its content to the browser. Now it differs how the server returns the required information in the case of static and dynamic websites.

### **TYPES OF WEBSITES.....**

- 1) static website
- 2) dynamic website

#### **1) Static Website**

web pages are returned by the server which are prebuilt source code files built using simple languages such as html, css, js. There is no processing of the content on the server (according to the user) in static

website.web pages are returned by the server with no change."therefore,static websites are fast,also they are less costly.

Note: Static does not mean that it will not respond to user actions,these websites are called static because these cannot be manipulated on the server.

## 2)Dynamic Website

web pages are returned by the server which are processed during runtime means they are not prebuilt web pages but they are build during runtime according to the user's demand with the help of server-side scripting languages such as php,nodejs,asp.net and many more supported by the server. so they are slower than static websites but updates and intaction with database are possible.

Dynamic websites are used over static websites are updates can be done very easily as compared to static website(Where altrng in every page is required) but in dynamic websites it is possible to do common changes once and it will reflect in all the web pages.

### Difference between Static website and Dynamic website?

Static website	Dynamic website:
1)content of web pages can't be change at runtime. 2)It is faster to load as compared to dynamic website 3)cheaper development costs 4)html,css,js is used for developing the websites 5)same content is delevered everytime the page is loaded. examples: any educations websites	1)content of web pages can be changed. 2)It is slower than static website. 3)More development costs. 4)server side languages such as PHP,Nodejs etc are used. 5)content may changes everytime the page is loaded.

### installation process for vscode:

#### step 1:

open google chrome and search "visual studio code for windows" download and install.

#### step 2:

open vs code and open extensions section and search live server and install.

**step 3:**

create a folder on desktop and name it as WEB Technology.

**step 4:**

open that folder and right click and select the open with vs code

or

open that folder in address bar type "cmd" press enter. then it will open command prompt . in command prompt type "code ." press enter.

**How to create a html file in vs code**

**step 1:**

in vs code top left side corner we can see one file icon click on it , give your file name with .html file extension. press enter

**step 2:**

press shift+1 you will get ! and press enter or type html:5 press enter.

**what is html tag?**

HTML tags are the predefined keywords on a web page that define how your web browser must format and display your web page.

Almost all tags contains two parts

1.opening tag

2.closing tag

ex: <html></html>

## what is an html elements?

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

ex: <tagname> content goes here .....</tagname>

## what is head tag?

the head element is a container for metadata.(metadat is data about the html document).the head element contains machine readable information about the document like style sheet ,title, script other metadata.

Note:Head tag primarily holds information for machine processing, not for the human readability.

## what is body tag?

the body tag defines the document's body. the body element contains all the content of an html document, such as headings,tables,lists etc.

Note: only one body tag should be in a single html document.

## HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

## horizontal line:

The <hr> tag in HTML stands for "**horizontal rule**." It is a self-closing tag

that is used to create a thematic break or horizontal line in a webpage.

The `<hr>` tag is primarily used to separate content or sections within a

document, providing visual clarity and structure to the page.

The `<hr>` element is used to separate content (or define a change) in an HTML page

**Thematic Break:** It is often used to denote a thematic break between different sections of content within a webpage. For example, you might use `<hr>` to separate different topics in an article, distinguish sections of a form, or divide content within a blog post.

The `<hr>` tag is an empty tag, which means that it has no end tag.

## HTML Line Breaks

The HTML `<br>` element defines a line break.

Use `<br>` if you want a line break (a new line) without starting a new paragraph.

The `<br>` tag is an empty tag, which means that it has no end tag.

**`<p>`** Defines a paragraph

**`<hr>`** Defines a horizontal rule in the content

**`<br>`** Inserts a single line break

**<pre>** Defines pre-formatted text

## HTML Entities

Reserved characters in HTML must be replaced with entities:

< (less than) = &lt;

> (greater than) = &gt;

### Formatting tags in html:

**<b>** - Bold text

**<strong>** - Important text

**<i>** - Italic text

**<em>** - Emphasized text

**<mark>** - Marked text

**<del>** - Deleted text

**<sub>** - Subscript text

**<sup>** - Superscript text

**<abbr>** - abbreviation text

**<u>** - underline the text

### LIST IN HTML:

HTML lists allow web developers to group a set of related items in lists.

### Types of list in html

#### 1) Ordered HTML list

The HTML <ol> tag defines an ordered list. An ordered list can be numerical or alphabetical.

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.

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

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 number
type="i"	The list items will be numbered with lowercase roman number

By default, an ordered list will start counting from 1. If you want to start counting from a specified number, you can use the start attribute.

example: <ol start="10">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

## **2)Unordered list**

The HTML <ul> tag defines an unordered (bulleted) 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.

The CSS list-style-type property is used to define the style of the list item marker. It can have one of the following values.

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

example:

```
<ul style="list-style-type:disc;">
```

```
<li>Coffee</li>
```

```
<li>Tea</li>
```

```
<li>Milk</li>
```

```
</ul>
```

### 3) Description List:

A description list is a list of terms, with a description of each term.

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term.

Use the HTML <dl> element to define a description list

Use the HTML <dt> element to define the description term

Use the HTML <dd> element to describe the term in a description list

example:

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

#### **4) Nested List**

Lists can be nested (list inside list):

example:

Using HTML5, produce the following nested lists exactly as illustrated in Figure 1. It includes a header, ordered, unordered and definition nested lists.

## List Example

- I. List Item 1
    - a. Nested item 1.1
    - b. Nested item 1.2
  - II. List Item 2
    - 1. Nested item 2.1
    - 2. Nested item 2.2
      - Nested item 2.2.1
      - Nested item 2.2.2
        - Nested item 2.2.2.1
        - Nested item 2.2.2.2
      - Nested item 2.2.3
    - 3. Nested item 2.3
  - III. List Item 3
    - Nested item 3.1
    - Nested item 3.1
    - Nested item 3.1
- COMP 249  
Object-Oriented Programming II.  
Soen287  
Web Programming.

This is a <sup>5</sup>test<sub>9</sub> for Assignment 1.

**Figure 1.** List illustration in HTML

## HTML Tables:

HTML tables allow web developers to arrange data into rows and columns.

`<table></table>` tag is used to define a table in html.

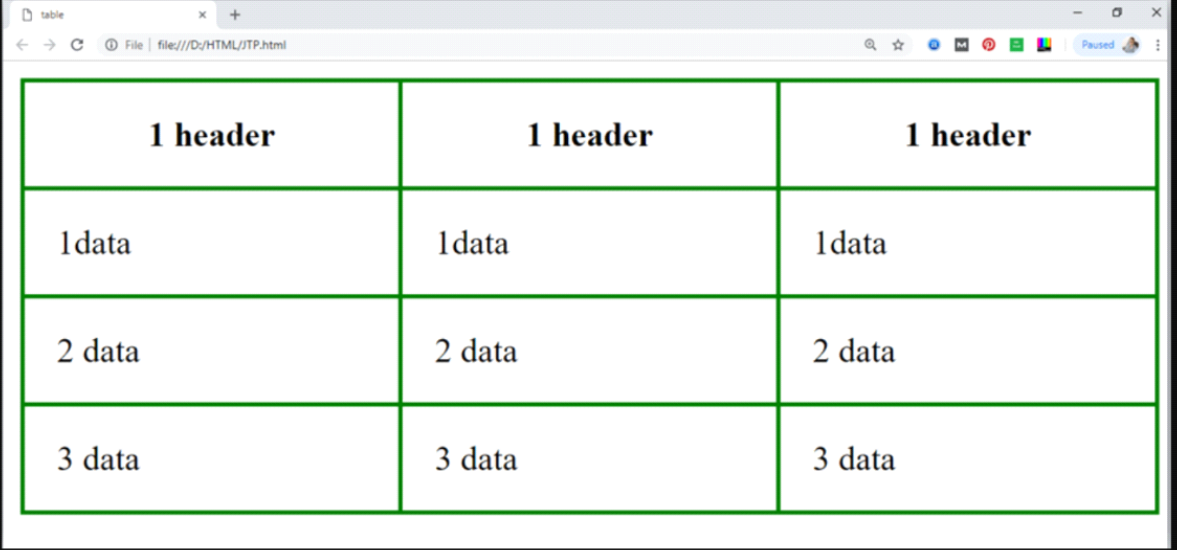
Each table row starts with a `<tr>` and ends with a `</tr>` tag.

The `<th>` tag defines a header cell in an HTML table.

The `<td>` tag defines a standard data cell in an HTML table.

In HTML tables are created in row by row.

Example:



1 header	1 header	1 header
1 data	1 data	1 data
2 data	2 data	2 data
3 data	3 data	3 data

The `<thead>` tag is used to group header content in an HTML table.

The `<tbody>` tag is used to group the body content in an HTML table.

The `<tfoot>` tag is used to group footer content in an HTML table.

The `<caption>` tag defines a table caption.

The `<caption>` tag must be inserted immediately after the `<table>` tag.

To make a cell span over multiple columns, use the `colspan` attribute.

To make a cell span over multiple rows, use the `rowspan` attribute.

The `<colgroup>` tag specifies a group of one or more columns in a table for formatting.

The `<colgroup>` tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.

The `<col>` tag specifies column properties for each column with `<colgroup>` element.

The `<col>` tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.

Attributes in col tag:

<a href="#">span</a>	<i>number</i>	Specifies the number of columns a <code>&lt;col&gt;</code> element should span
----------------------	---------------	--

### HTML `<a></a>` (anchor) tag:

The `<a>` tag defines a hyperlink, which is used to link from one page to another.

The most important attribute of the `<a>` element is the `href attribute`, which indicates the link's destination.

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

**NOTE:** If the `<a>` tag has no `href` attribute, it is only a placeholder for a hyperlink.

Attributes for a tag:

1)href

2)target

3)download

### 1)href attribute

The **href** attribute specifies the URL of the page the link goes to.

If the **href** attribute is not present, the **<a>** tag will not be a hyperlink.

#### Syntax

```
<a href="URL"></a>
```

Example:

```
<a href="https://www.google.com">Visit Google</a>
```

### 2)target attribute

The **target** attribute specifies where to open the linked document.

Value	Description
_blank	Opens the linked document in a new window or tab
_self	Opens the linked document in the same frame as it was clicked (this is default)
_parent	Opens the linked document in the parent frame
_top	Opens the linked document in the full body of the window

### 3)download attribute

The **download** attribute specifies that the target (the file specified in the **href** attribute) will be downloaded when a user clicks on the hyperlink.

The optional value of the **download** attribute will be the new name of the file after it is downloaded. There are no restrictions on allowed values, and the browser

will automatically detect the correct file extension and add it to the file (.img, .pdf, .txt, .html, etc.).

If the value is omitted, the original filename is used.

Examples:

```
<a href="/images/schoolimage.jpg" download>click</a>
```

```
<a href="/images/schoolimage.jpg" download="old_picture">click here</a>
```

```
<a href="#top"></a> it helps to move the top section of the web page.
```

## HTML **<img>** Tag:

The `<img>` tag is used to embed an image in an HTML page.

Images are not technically inserted into a web page; images are linked to web pages. The `<img>` tag creates a holding space for the referenced image.

The `<img>` tag has two required attributes:

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

**Note:** Also, always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads.

## HTML **<audio>** Tag:

The `<audio>` tag is used to embed sound content in a document, such as music or other audio streams.

The text between the `<audio>` and `</audio>` tags will only be displayed in browsers that do not support the `<audio>` element.

There are three supported audio formats in HTML: MP3, WAV, and OGG.

Attribute	Value	Description
<a href="#">autoplay</a>	autoplay	Specifies that the audio will start playing as soon as it is ready
<a href="#">controls</a>	controls	Specifies that audio controls should be displayed (such as a play/pause button)
<a href="#">loop</a>	loop	Specifies that the audio will start over again, every time it is finished
<a href="#">muted</a>	muted	Specifies that the audio output should be muted

HTML **<video>** Tag:

Same as audio tag



### **The <figure> tag:**

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

### **The <figcaption> tag:**

The <figcaption> tag defines a caption for a <figure> element.

The <figcaption> element can be placed as the first or last child of the <figure> element.

### **The <details> tag:**

The <details> tag specifies additional details that the user can open and close on demand.

The <details> tag is often used to create an interactive widget that the user can open and close. By default, the widget is closed. When open, it expands, and displays the content within.

### **The <summary> tag:**

The <summary> tag defines a visible heading for the <details> element. The heading can be clicked to view/hide the details.

Note: The <summary> element should be the first child element of the <details> element.

### **The <form> tag :**

The <form> tag is used to create an HTML form for user input.

### **The <input> tag:**

The <input> tag specifies an input field where the user can enter data.

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the type attribute.

The different input types are as follows:

<input type="text"> (default value)

<input type="email">

<input type="number">

<input type="password">

<input type="tel">

<input type="date">

<input type="checkbox">

<input type="radio">

<input type="range">

<input type="submit">

<input type="search">

<input type="reset">

<input type="url">

<input type="button">

**HTML tags are classified into two types.**

- **Semantic**
- **Non-Semantic**

## **Semantic Elements:**

Semantic elements have meaningful names that tell about the type of content. For example header, footer, table, ... etc. HTML5 introduces many semantic elements as mentioned below which make the code easier to write and understand for the developer as well as instruct the browser on how to treat them.

- **<article>**  
It contains independent content which doesn't require any other context blog Post, Newspaper Article, etc.
- **<aside>**  
It is used to place content in a sidebar i.e. aside from the existing content. It is related to surrounding content.
- **<details>**  
"details" defines additional details that the user can hide or view. "summary" defines a visible heading for a "details" element.
- **<figcaption>**  
These are used to add an image to a web page with a small description.
- **<figure>**  
These are used to add an image to a web page with a small description.
- **<footer>**  
Footer located at the bottom of any article or document, they can contain contact details, copyright information etc. There can be multiple footers on a page.
- **<header>**  
As the name suggests, it is for the header of a section introductory of a page. There can be multiple headers on a page.
- **<main>**  
It defines the main content of the document. The content inside the main tag should be unique.
- **<nav>**  
It is used to define a set of navigation links in the form of a navigation bar or nav menu.

- **<section>**  
A page can be split into sections like Introduction, Contact Information, Details, etc and each of these sections can be in a different section tag.

## Non-Semantic Elements:

Tags like div, and span fall under the Non-Semantic categories as their names don't tell anything about what kind of content is present inside them.

## CSS

**CSS stands for Cascading Style Sheet.**

Used to give styling to the web pages which is structured by the HTML language.

There are three ways in CSS to Style the Web pages

1. Inline CSS
2. Internal CSS
3. External CSS
4. Import.

1. **Inline CSS:** Inline CSS code is written in the opening tag of the HTML element by using style attribute.

Syntax:           <p style=" "> </p>

2. **Internal CSS:** Internal CSS code is written in the head section of an HTML element using <style> tag.

Syntax: <head>

```
        <style>
            tag_name{
                property: value;
            }
        </style>
    </head>
```

3. **External CSS:** External CSS styling can be done by creating an external CSS file with file\_name.css extension and providing the link between that CSS file to HTML file by <link> tag in the head section of HTML element.

Syntax: <head>

```
        <link rel="stylesheet" href="file_name.css">
    </head>
```

4. @import: This styling can be done by linking one CSS file to another CSS file.

Syntax: @import url(file\_name.css)

Import should be written in the very first line of the CSS file which has to be linked.

NOTE: The first priority is inline CSS

But the priorities of the Internal and External CSS vary accordingly depending on the declared position

```
if,      <link rel="stylesheet" href="index.css">
        <style> </style>
```

Here, the first priority is for Internal CSS.

```
if,      <style> </style>
```

```
<link rel="stylesheet" href="index.css">
```

Here, the first priority is for External CSS.

## **SELECTOR**

Selectors are used to select the particular HTML element and to style them.

There are five types of Selectors

1. Simple Selectors
2. Combinator Selectors
3. Attribute Selectors
4. Pseudo Element Selectors.
5. Pseudo Classes Selectors.

### **Simple Selectors**

Simple selectors style the HTML element in five ways:

1. Id Selector.
2. Class Selector.

3. Tagname.
4. Groupname.
5. Universe.

#### **Id selectors:**

Id selector targets only individual element in the HTML document.

Prefix symbol used for id selector is # (hash)

Example:      #demo{  
                 color: red;  
                 background: yellow;  
                 }

#### **Class Selector**

Class selector targets multiple elements in the HTML document.

Prefix symbol used for id selector is . (dot)

Example:      .test{  
                 color: red;  
                 background: yellow;  
                 }

#### **Tagname Selector**

Tagname selector targets the HTML elements by tag name.

There is no symbol used in tagname selectors.

Example:      h1{  
                 color: magenta;  
                 background: green;

```
}  
h2{  
    color: red;  
    background: black;  
}  
h3{  
    color: yellow;  
    background: orange;  
}
```

### Grouping Selector

Groupname selector targets a group of HTML elements by tagname separated by commas.

Example:

```
p, div, h4{  
    color: magenta;  
    background: green;  
}
```

### Universal Selector

Universal Selectors targets every HTML element.

The symbol of Universal Selector is \*

There is no selector is declared in css.

Example:

```
*{  
    color: yellow;  
    background: magenta;  
}
```



**Color Property :** Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

- An RGB color value represents RED, GREEN, and BLUE light sources.
- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color. `rgba(red, green, blue, alpha)`. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)
- HEX, A hexadecimal color is specified with: `#RRGGBB`, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color. Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255). Sometimes you will see a 3-digit hex code in the CSS source. The 3-digit hex code is a shorthand for some 6-digit hex codes.
- HSL stands for hue, saturation, and lightness. `hsl(hue, saturation, lightness)`. Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all). `hsla(hue, saturation, lightness, alpha)`

Note: CSS/HTML support [140 standard HYPERLINK](https://www.w3schools.com/colors/colors_names.asp)  
["https://www.w3schools.com/colors/colors\\_names.asp"color HYPERLINK](https://www.w3schools.com/colors/colors_names.asp)  
["https://www.w3schools.com/colors/colors\\_names.asp" names.](https://www.w3schools.com/colors/colors_names.asp)

Example: color: `color-name | rgb() | hex | hsl () | rgba() | hsla();`

## CSS background properties

The CSS background properties are used to add background effects for elements.

- **background-color**
  - The **background-color** property specifies the background color of an element.

- Example: `background-color: lightblue;`
- `background-image`
  - The `background-image` property specifies an image to use as the background of an element.
  - By default, the image is repeated so it covers the entire element
- `background-repeat: repeat-x | repeat-y | no-repeat ;`
- `background-position : right top | center | right bottom`
- The `background-attachment` property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):
  - `background-attachment: fixed | scroll;`
- `background-size: cover | contain;`

7. `background`, Use the shorthand property to set the background properties in one declaration.

Example:

```
background: #ffffff url("img_tree.png") no-repeat right top;
```

## BOX Model

Box Model is essentially a box that wraps around every HTML element. Box Model is used to design and layout.

The CSS Box Model consists of Margin, Border, Padding and the actual Content.

**Content:** It is the content of the HTML element.

**Padding:** It is the space after the content.

**Border:** It is the Border of the HTML element.

**Margin:** It is the space after the Border of an HTML element.

Padding and Margin allow up to four values.

## Combinator Selector

Combinator Selectors are used to style the combination of HTML elements.

There are four Combinator Selectors:

1. Descendent Selectors
2. Child Selectors
3. Adjacent Sibling Selectors
4. General Sibling Selectors

**1. Descendent Selectors:** This Selector targets both direct and indirect children of a parent tag. The symbol is (space)

Syntax: `parent_tag child_tag{property: value;}`

**2. Child Selectors:** This Selector targets the only the direct children of the parent tag. The symbol is > (greater than)

Syntax: `parent_tag > child_tag{property: value;}`

**3. Adjacent Selectors:** This Selector targets the element which is the first sibling of the targeted HTML tag or element.

(or)

This Selector targets the element which is exactly adjacent to the targeted HTML tag or element.

The symbol is + (plus)

Syntax: `parent_tag + target_tag{property: value;}`

**4. General Sibling Selectors:** This Selector targets all the siblings of the target HTML tag or element.

(or)

This Selector targets all the adjacent tags of the target HTML tag or element. The symbol is ~ (tilde)

Syntax: `parent_tag ~ target_tag{property: value;}`

### **CSS Pseudo Classes Selectors.**

A Pseudo class in CSS is used to define the special state of an element. It can be combined with a CSS selector to add an effect to existing elements based on their states. For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.

Note that pseudo-class names are not case-sensitive.

Syntax:

```
selector: pseudo-class{  
    property: value;  
}
```

There are many Pseudo-classes in CSS but the ones that are most commonly used are as follows:

- 1):hover
- 2):active
- 3):focus
- 4):visited
- 5):first-child
- 6):nth-child(n)

**1) :hover** :This pseudo-class is used to add a special effect to an element when our mouse pointer is over it.

ex:

```
<style>
```

```
.box {  
    background-color: yellow;  
    width: 300px;  
    height: 200px;  
    margin: auto;  
    font-size: 40px;  
    text-align: center;  
}
```

```
.box:hover {  
    background-color: orange;  
}
```

```
</style>
```

```
<body>
```

```
<div class="box">
```

```
    My color changes if you hover over me!
```

```
</div>
```

```
</body>
```

**2):active** Pseudo-class: This pseudo-class is used to select an element that is activated when the user clicks on it.

ex:

```
<style>
.box{
    background-color: yellow;
    width: 300px;
    height: 200px;
    margin: auto;
    font-size: 40px;
    text-align: center;
}
.box:active{
    background-color: orange;
}

</style>
<body>
<div class="box">
    My color changes if you hover over me!
</div>
</body>
```

**3):focus** Pseudo-class: This pseudo-class is used to select an element that is currently focused by the user. It works on user input elements used in forms and is triggered as soon as the user clicks on it.

ex:

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

**4):visited** Pseudo-class: This pseudo-class is used to select the links which have been already visited by the user.

```
ex: a:visited{  
    color: red;  
}
```

**5) :first-child** pseudo-class matches a specified element that is the first child of another element.

```
ex:p:first-child {  
    color: blue;  
}
```

**6) :nth-child(n)** selector matches every element that is the nth child of its parent.

n can be a number, a keyword (odd or even), or a formula (like  $an + b$ ).

```
ex:tr:nth-child(odd) {  
    background: red;  
}
```

```
tr:nth-child(even) {  
    background: lightgreen;  
}
```

### **CSS Pseudo Element Selectors.**

A CSS pseudo-element is a keyword added to a selector that lets you style a specific part of the selected elements. For Example, Styling the first letter or line of an element, and Inserting content before or after the content of an element. All of these can be done using Pseudo Elements in CSS.

Note that in contrast to pseudo-elements, pseudo-classes can be used to style an element based on its state.

Syntax:

```
selector::pseudo-element {  
    property: value;  
}
```

There are many Pseudo Elements in CSS but the ones which are most commonly used are as follows:

- 1)::first-line
- 2)::first-letter
- 3)::before
- 4)::after
- 5)::selection

**1)::first-line** Pseudo-element applies styles to the first line of a block-level element. Note that the length of the first line depends on many factors, including the width of the element, the width of the document, and the font size of the text. Note that only a few properties are applied for first-line pseudo-element like font properties, color properties, background properties, word-spacing, letter-spacing, text-decoration, vertical-align, text-transform, line-height, clear, etc.

ex:

<style>

```
body {  
    background-color: whitesmoke;  
    color: green;  
    font-size: large;  
    text-align: center;
```



```
}

p::first-line {
    color: Red;
    font-weight: bold;
}

</style>

<body>

<p>
    This is a paragraph using first-line
    pseudo-element to style first line of
    the paragraph. Content in the first
    line turns red and becomes bold.
</p>
</body>
```

**2)::first-letter** Pseudo-element applies styles to the first letter of the first line of a block-level element, but only when not preceded by other content (such as images or inline tables). Note that only a few properties are applied for first-line pseudo-element like font properties, color properties, background properties, word-spacing, letter-spacing, text-decoration, vertical-align, text-transform, line-height, clear, etc.

```
ex: <style>

    body {

        background-color: whitesmoke;

        color: green;

        font-size: large;

        text-align: center;

    }
```

```
p::first-letter {  
    color: Red;  
    font-size: 30px;  
    font-weight: bold;  
}  
</style>  
  
<body>  
  
<p>  
    This is a paragraph using first-letter  
    pseudo-element to style first letter  
    of the paragraph. first-letter element  
    turned the first letter of this paragraph  
    to red and made it bold.  
</p>  
</body>
```

**3)::before** Pseudo-element creates a pseudo-element that is the first child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

ex:

```
p::before {  
    content: "";  
    color: red;  
    font-size: 30px;  
}
```

**4)::after** Pseudo-element creates a pseudo-element that is the last child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

ex:

```
p::after {  
    content: "";  
    color: red;  
    font-size: 30px;  
}
```

**5)::selection** Pseudo-element applies styles to the part of a document that has been highlighted by the user such as clicking and dragging the mouse across the text.

ex:

```
p::selection {  
    color: red;  
    background-color: green;  
    font-size: 30px;  
}
```

```
::selection {  
    color: green;  
    background-color: red;  
    font-size: 30px;  
}
```

## CSS [attribute] Selector

The `[attribute]` selector is used to select elements with a specified attribute.

Example:

```
a[target] {  
    background-color: yellow;  
}
```

## CSS [attribute="value"] Selector

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

Example:

```
a[target="_blank"] {  
  background-color: yellow;  
}
```

## CSS [attribute~="value"] Selector

The `[attribute~="value"]` selector is used to select elements with an attribute value containing a specified word.

Example:

```
[title~="flower"] {  
  border: 5px solid yellow;  
}
```

## CSS [attribute|= "value"] Selector

The `[attribute|= "value"]` selector is used to select elements with the specified attribute, whose value can be exactly the specified value, or the specified value followed by a hyphen (-).

**Note:** The value has to be a whole word, either alone, like `class="top"`, or followed by a hyphen( - ), like `class="top-text"`.

Example:

```
[class|= "top"] {  
  background: yellow;  
}
```

## CSS [attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements with the specified attribute, whose value starts with the specified value.

**Note:** The value does not have to be a whole word!

Example:

```
[class^="top"] {  
  background: yellow;  
}
```

## CSS [attribute\$="value"] Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

**Note:** The value does not have to be a whole word!

Example:

```
[class$="test"] {  
  background: yellow;  
}
```

## CSS [attribute\*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

**Note:** The value does not have to be a whole word!

Example:

```
[class*="te"] {  
  background: yellow;  
}
```

Note: The attribute selectors can be useful for styling forms without class or ID.

Example:

```
input[type="text"] {  
  width: 150px;  
  background-color: yellow;  
}  
  
input[type="button"] {  
  width: 120px;  
  background-color: lightblue;  
}
```

## The display Property

The `display` property is used to specify how an element is shown on a web page.

Every HTML element has a default display value, depending on what type of element it is. The default display value for most elements is `block` or `inline`.

The `display` property is used to change the default display behavior of HTML elements.

### Block-level Elements:

A block-level element ALWAYS starts on a new line and takes up the full width available .

Examples of block-level elements:

- `<div>`
- `<h1>` to `<h6>`
- `<p>`
- `<form>`
- `<header>`
- `<footer>`
- `<section>`
- `<li>`

- <figure>
- <pre>
- <nav>
- <hr>
- <ul>
- <ol>
- <table>

## Inline Elements:

An inline element DOES NOT start on a new line and only takes up as much width as necessary.

Examples of inline elements:

- <span>
- <a>
- <button>
- <select>
- <textarea>
- All formatting tags

## The display Property Values

Value	Description
<ul style="list-style-type: none"> <li>• <b>Inline</b></li> <li>• Block</li> <li>• inline-block</li> <li>• <b>none</b></li> <li>• flex</li> </ul>	<ul style="list-style-type: none"> <li>• Displays an element as an inline element</li> <li>• Displays an element as a block element</li> <li>• Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values</li> <li>• The element is completely removed.</li> <li>• Displays an element as a block-level flex container</li> </ul>

<ul style="list-style-type: none"> <li>• <code>grid</code></li> </ul>	<ul style="list-style-type: none"> <li>• Displays an element as a block-level grid container</li> </ul>
---	---

## Inline Block Elements

The display value of inline-block works similarly to inline with one exception: the height and width of that element become modifiable.

**visibility:hidden;** also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

### Difference between display:none and visibility: hidden

**visibility:hidden** hides the element, but it still takes up space in the layout.

**display:none** removes the element from the document. It does not take up any space.

### display:flex;

The **flexbox** or flexible box model in CSS is a one-dimensional layout model that has flexible and efficient layouts with distributed spaces among items to control their alignment structure ie., it is a layout model that provides an easy and clean way to arrange items within a container. Flexbox can be useful for creating small-scale layouts & is responsive and mobile-friendly

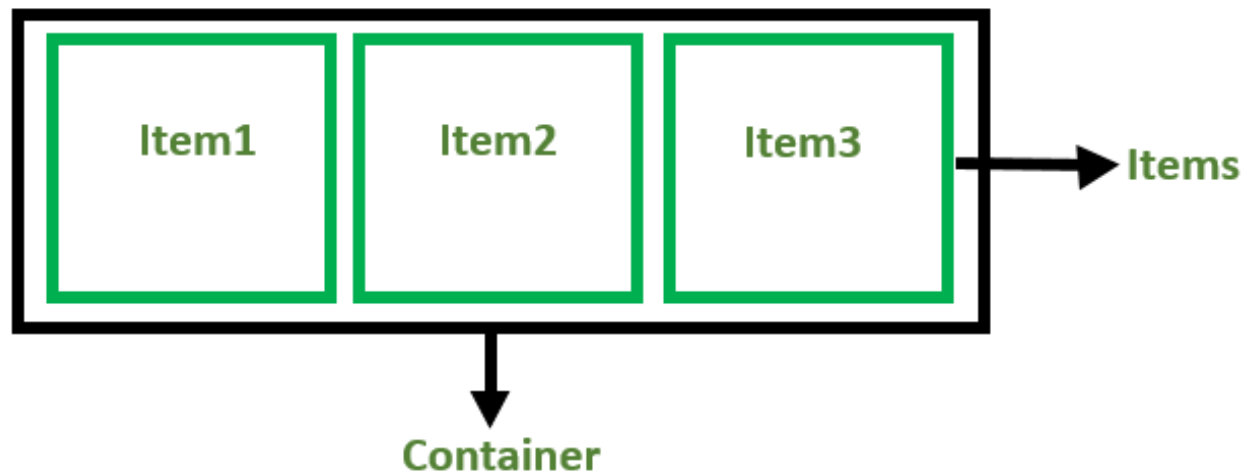
#### Features of flexbox:

- A lot of flexibility is given.
- Arrangement & alignment of items.
- Proper spacing
- Order & Sequencing of items.
- Bootstrap 4 is built on top of the flex layout.

There are 2 main components of the Flexbox:

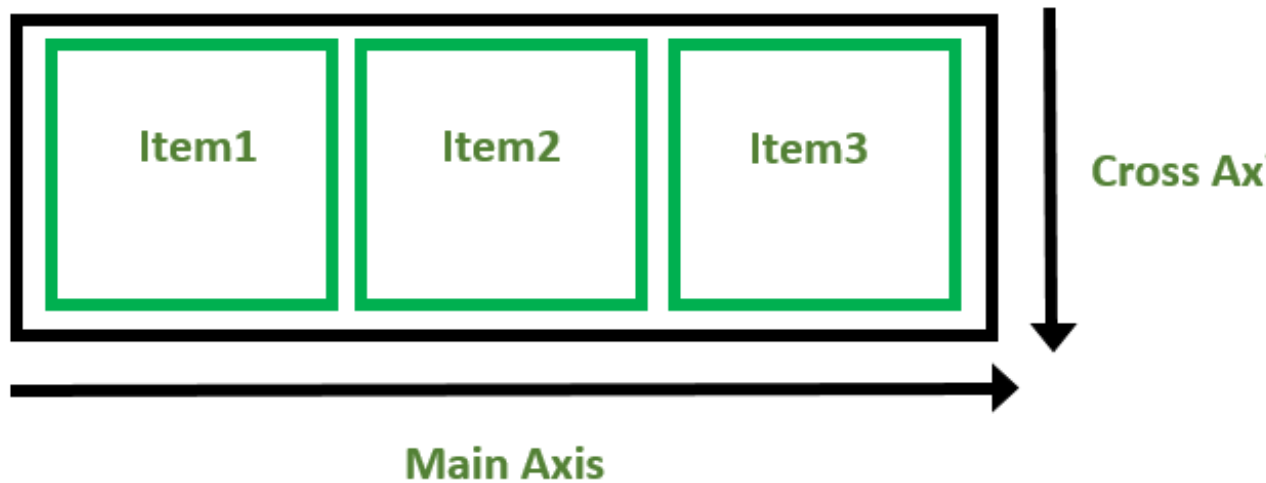


- **Flex Container:** The parent “div” which contains various divisions is called a flex container.
- **Flex Items:** The items inside the container “div” are flex items.



**Flexbox Axes:** While working with Flexbox, we deal with 2 axes:

- Main Axis
- Cross Axis



### Properties of flexbox:

Parent properties:

- **display** defines a flex container – flex formatting context.
- **flex-direction** defines the main axis inside the container.
- **flex-wrap** allows flex items to wrap onto more than one line.
- **justify-content** allows items to align along main axis.
- **align-items** aligns multiple lines of items on the cross axis.

### Children/Flex-items properties:

- **flex-grow** allows an item to fill up the available free space.
- **flex-shrink** allows an item to shrink if there is no enough free space available.
- **flex-basis** defines the size of an item before space is distributed.

## CSS Positions

The **position** property in CSS tells about the method of positioning for an element or an HTML entity and the positioning of an element can be done using the top, right, bottom, and left properties.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

**Note:** Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

### position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

### position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

## position: sticky;

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like `position:fixed`).

## position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

## position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like `fixed`).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

**Note:** Absolute positioned elements are removed from the normal flow, and can overlap elements.

## CSS z-index Property

The `z-index` property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

**Note:** `z-index` only works on positioned elements (`position: absolute`, `position: relative`, `position: fixed`, or `position: sticky`) and flex items (elements that are direct children of `display: flex` elements).

**Note:** If two positioned elements overlap without a `z-index` specified, the element positioned last in the HTML code will be shown on top.

## CSS Overflow

The `overflow` property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The `overflow` property has the following values:

- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`, but it adds scrollbars only when necessary

**Note:** The `overflow` property only works for block elements with a specified height.

## overflow-x and overflow-y

The `overflow-x` and `overflow-y` properties specify whether to change the overflow of content just horizontally or vertically (or both):

`overflow-x` specifies what to do with the left/right edges of the content.

`overflow-y` specifies what to do with the top/bottom edges of the content.

# JAVASCRIPT

## **What is JavaScript?**

Javascript is a high-level , object based ,programming language. Javascript is a interpreted, user friendly ,client-side scripting language.

## **History of javascript:**

javascript developed by Brenden Eich-1995

First name of javascript --> MOCHA

next --> LIVESCRIPT

next--> JAVASCRIPT

## **FEATURES OF JAVASCRIPT:**

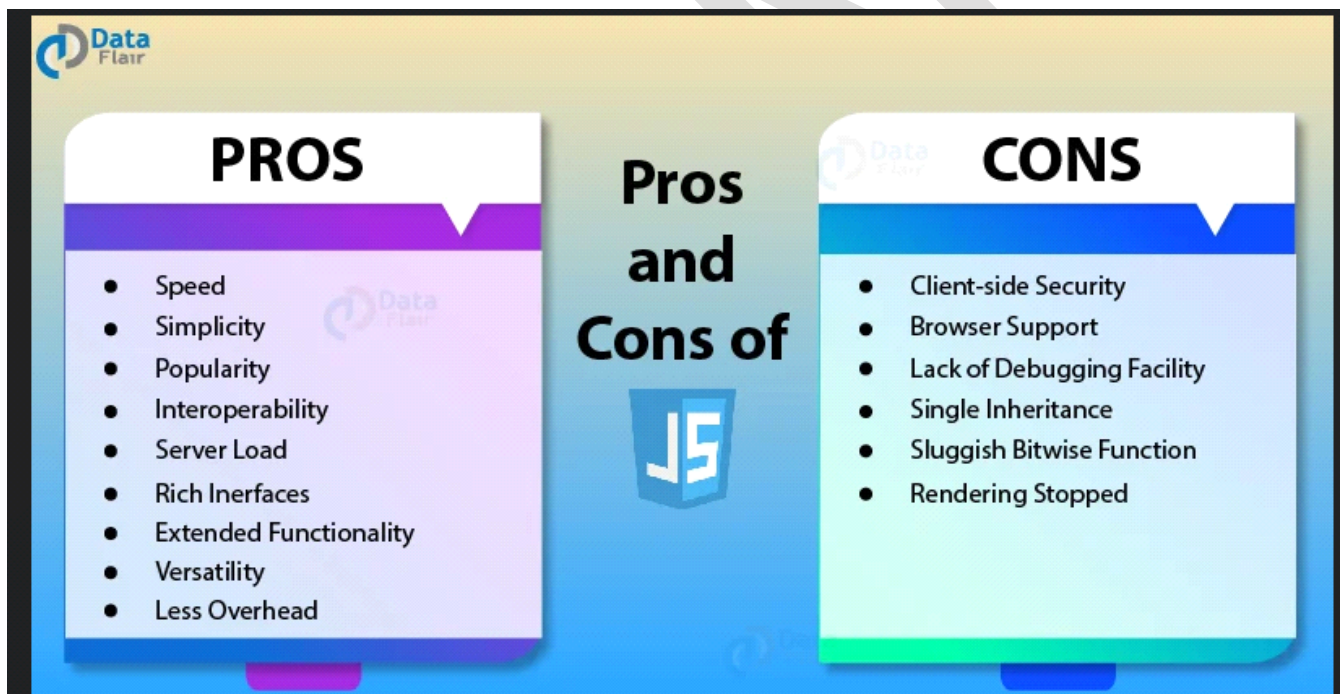
- 1)Light Weight
- 2)intepreted language
- 3)open source
- 4)cross platform
- 5)Dynamically typed language
- 6)Weakley typed language
- 7)client-side scripting language
- 8)Synchronous language

## Javascript characteristics:

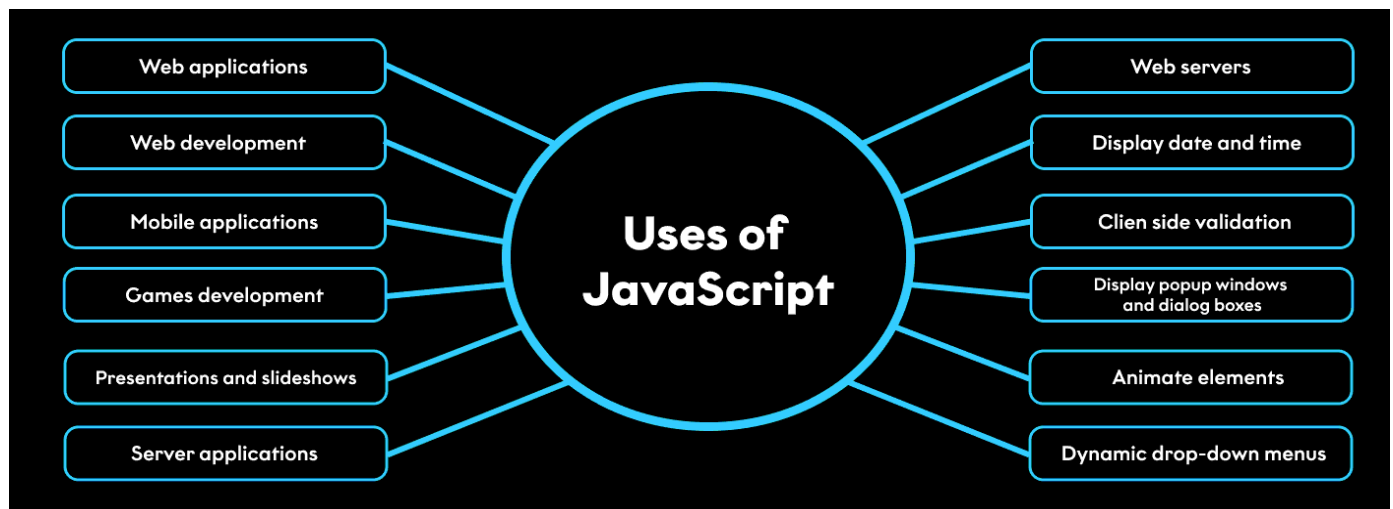
- Client-side-Scripting language - no compiler and without help of server logic we can update the data
- High-level language - user-friendly
- Interpreted language – line by line execution

- Single threaded – one call stack and one heap area
- Loosely typed – no strong syntax
- Dynamic language – can change the datatype during the runtime
- Object-based language – In JavaScript everything was object

### Advantages and disadvantages of JavaScript:



### Uses of JavaScript:



## Difference between Java and JavaScript

### Java

### javascript

Java is a strongly typed language and variables must be declared first to use in the program. In Java, the type of a variable is checked at compile-time.	JavaScript is a loosely typed language and has a more relaxed syntax and rules.
Java is an object-oriented programming language primarily used for developing complex logic applications.	JavaScript is a <a href="#">scripting language</a> used for creating interactive and dynamic web pages.
Java applications can run in any virtual machine(JVM) or browser.	JavaScript code used to run only in the browser, but now it can run on the server via Node.js.
Objects of Java are class-based even we can't make any program in java without creating a class.	JavaScript Objects are prototype-based.
Java program has the file extension ".Java" and translates source code into bytecodes which are executed by JVM(Java Virtual Machine).	JavaScript file has the file extension ".js" and it is interpreted but not compiled, every browser has the Javascript interpreter to execute JS code.
Java supports multithreading, which allows multiple threads of execution to run concurrently within a single program.	JavaScript does not support multithreading, although it can simulate it through the use of web workers.
Java has a rich set of libraries and frameworks for building enterprise applications, such as Spring, Hibernate	JavaScript has a wide variety of libraries and frameworks for building web applications, such as React, Angular, and Vue.
Java is mainly used for backend.	Javascript is used for the frontend and backend both.



Java is statically typed, which means that data types are determined at compile time. Java uses more memory	JavaScript is dynamically typed, which means that data types are determined at runtime. Javascript uses less memory.
---	--

**Note:** The file extension for the javascript is **.js**

### **Attaching the javascript file to html file:**

We can attach the js file with html file in 2 ways.

#### **1.Internal Way**

#### **2.External Way**

##### **1.Internal Way**

By writing the code of javascript inside `<script></script>` tag

#### **Syntax:**

```

<body>
    <h1>hello world</h1>
    <script>
        // javascript code.....
    </script>
</body>

```

##### **2. External Way**

- To write javascript externally to html file, we need to create an external javascript file with extension as **.js** (ex: index.js)
- After that link that javascript file to the html by using script tag.

#### **Example:**

```

<body>
    <script src = "./index.js "></script>
</body>

```

## Printing Statements in javascript:

In javascript we have 2 types of printing statements

1.document.write();

2.console.log();

### 1.document.write()

document.write() is a printing statement which is used to see the output in the web browser (client purpose)

### 2.console.log()

Console.log() is a printing statement which is used to see the output in the console window (developer view)

Here, console. is an object , dot is a period (or) access operator and log is a function and member of console. that accepts argument as data to print in console.

## TOKENS:

In JavaScript, a token is the smallest unit of a program that is meaningful to the interpreter. JavaScript code is made up of various types of tokens, including

- **Keywords:**

Reserved words that have special meanings in the language. Examples include **var**, **if**, **else**, **for**, **while**, etc.

NOTE: Every keyword must be in the lower case and it is not used as Identifiers.

### 2. Identifiers:

These are names that are used to identify variables, functions, and other objects. For example, the identifier `myVariable` could be used to identify a variable that stores the value 5.

Rulers: -an identifiers can't start with number. -We can't use keywords as identifiers. - Except \$, \_ no other Special Characteristics are allowed.

### 3.Literals:

Data which is used in the js programming is called as literals. For example, the literal 5 is a numeric literal, and the literal "Hello, world!" is a string literal.

#### 4. Operators:

These are symbols that are used to perform operations on operands. For example, the operator + is used to add two numbers together.

### JavaScript Variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

- Name must start with a letter (a to z or A to Z), underscore( \_ ), or dollar( \$ ) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

#### Correct JavaScript variables

- var x = 10;
- var \_value="sonoo";

#### Incorrect JavaScript variables

- var 123=30;
- var \*aa=320;

### JavaScript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

- Primitive data type

- Non-primitive (reference) data type

## JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all
Operator	Description

## JavaScript non-primitive data types

The non-primitive data types are as follows:

Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

# JavaScript Operators

Operators are special symbols in JavaScript that are used to perform operations on values. There are many different types of operators, including arithmetic operators, comparison operators, logical operators, and assignment operators.

## JavaScript Arithmetic Operators

**Arithmetic Operators** are used to perform arithmetic on numbers:

+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation ( <a href="#">ES2016</a> )
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

# JavaScript Assignment

The **Assignment Operator** (=) assigns a value to a variable:

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

## JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10===20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!== "20" = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

## JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
	Logical OR	(10==20    20==33) = false
!	Logical Not	!(10==20) = true