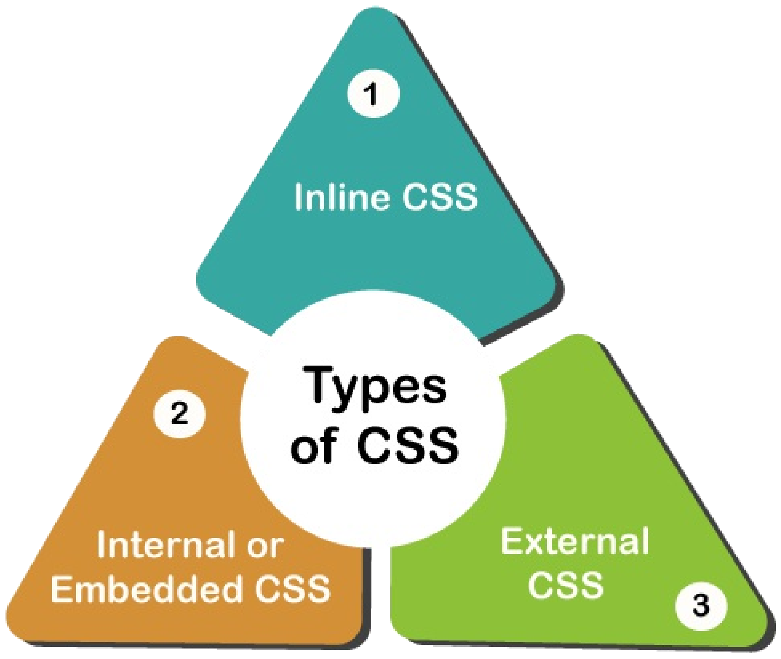
CSS

CSS(Cascading style sheet)

- CSS is the language we use to style and align our HTML elements.
- CSS was introduced in 1994 by Haykon wium lie.
- First version of CSS was introduced in 1996(CSS1)
- CSS2-1998
- Current version of CSS is CSS3-1999
- It is style sheet language which is used to describe the look and formatting of documents written in markup language.
- It is generally used with html to change the style of web pages and UI.
- Commenting in CSS:-- /*-----*/

How to add CSS to HTML



1.Inline style sheet :

- We will use style attribute in the tagline itself

<p style : " color: red">Hello CSS</p>

2.Internal style sheet :

- We have to use <style > tag in the <head> section of HTML documnet

Example:

<style>

p{color : value}

</style>

3.External style sheet :

- We have to create external file with .css extension and we need to link that CSS file to HTML file using <link> tag

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

@import

- @import rule is used to link one CSS within another CSS..

Syntax:- @import url("cssfilepath")

Selectors

- Selectors are used to target the HTML elements
- A CSS selector selects the HTML element(s) you want to style.
- Syntax :

Selector{

property: value;

property: value;

}

Types of CSS Selectors

1. Simple Selector
2. Combinator Selector
3. Attribute Selector
4. Pseudo class Selector
5. Pseudo Element Selector

1.Simple Selectors

1. Id selector (#)
2. Class selector (.)
3. Universal Selector (*)
4. Element Selector (tag)
5. Grouping Selector(,)

1.ID SELECTOR (#)

- Unique id attribute within the page is selected
- Core attribute selector
- Selected by symbol "#" followed by id name
- SYNTAX: #id_name{

css properties

}

2. CLASS SELECTOR (.)

- Core attribute selector ,with specific class.
- The character to access in css file " . "
- Multiple class name can be called using comma

SYNTAX: . class _ name{

/* css properties*/

}

3. UNIVERSAL SELECTOR(*)

- Select all elements in html page.
- All the elements within a body tag.
- Symbol of selector: " * "

SYNTAX:

*{

properties

}

4. TYPE/ ELEMENT SELECTOR(tagname)

- Selects particular element with specified tagname.
- Call by type of tag.

SYNTAX:

element name{

properties

}

5. GROUPING SELECTOR(,)

- We can group the multiple selectors and we can do styling

#heading,.para,button{

Properties

}

NOTE: PRIORITY ORDER =====

{ " ID > CLASS > TYPE/ELEMENT >UNIVERSAL "}

Combinator Selectors

- We will combine and target the elements

Types of combinator selectors

1. Descendent selector(space)
2. Child selector(>)
3. Adjacent sibling selector(+)
4. General sibling selector(~)

1.DESCENDENT SELECTOR (space)

- A CSS selector can contain more than one simple selector. Between the simple selector we can use combinator
- Combines two selectors are such that elements are matched by second selector are selected if they have ancestor.
- (parent,parents's parent, parent's parents's parent)
- Selector utilities a descendents combinator are called descendent selectors.

Syntax: Selector1_Selector2{property declaration}

Div p{ prop:val;}

2.Child selector(>)

- The child selector selects all the elements that are the children of a specified element
- It is placed between 2 CSS selectors,matches only those elements matched by second selector and direct child.

Syntax:

selector 1 > selector 2 { properties }

Ex:

Div>p{prop : val}

3. Adjacent sibling selector (+)

- The adjacent sibling selector is used to select an element that is directly after another specific element.
- Sibling elements must have the same parent element, and "adjacent" means "immediately following".

Syntax :

former_element + target_element {style properties }

4. General sibling selector(~)

- The general sibling selector selects all elements that are next sibling of a specified element.
- The general sibling combinator (~) separates two selectors.

Syntax :

former_element ~ target_element {

style properties

}

Pseudo classes

A CSS pseudo-class is a keyword added to a selector that specify a special state of the selected element.

For Example: It can be used to

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus
- Symbol for pseudo class selector is ':'

Selector:pseudo-class{

property:value;

}

In pseudo class selectors we have 2 types

1.structural pseudo class selectors

- Link: used to style the unvisited link
- Visited: used to style the visited link
- Active: used to style the active link
- Focus: used to style the text filed when we focus that
- Hover: used to style the element when mouse hover on the element

a:hover - style the element when mouse hover on the element

a:link - style the unvisited link

a:visited - style the unvisited link

a:active – style the active link

input:focus – style the focused text field

2.dynamic pseudo class selectors

- first-child
- last-child
- nth-child

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

The :last-child pseudo-class matches a specified element that is the last child of another element.

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

We can target only even children using nth-child() by passing even or 2n in parenthesis

We can target only odd children using nth-child() by passing odd or 2n+1 in parenthesis

Selector:nth-child(even){ Selector:nth-child(even){

Property Property

} }

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

Pseudo Elements

A CSS pseudo-element is used to style specified parts of an element.

Symbol for pseudo element selector is '::'

For example, it can be used to:

-
    - Style the first letter, or line, of an element
    - Insert content before, or after, the content of an element

selector::pseudo-element{

property:value;

}

:: first-line : styles the first letter of the element

:: first-letter: styles the first line of the element

::before : adds the content before the actual content

::after: adds the content after the actual content

::marker: style the list style type

::selection : style the selected content on the webpage

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user

Attribute selector

- Attribute provides extra information to the tag.
- In this attribute selector we are targeting the elements based on attributes

Targeting using attribute name

Selector[attribute]{

property}:value

}

Targeting using attribute name and value

Selector[attribute="value"]{

Property:value

}

Text Properties in CSS

color : property is used to set the color of the text

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb (255,0,0)"
- An RGB color value represents red, green, blue.
- Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
- For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.
- To display black, set all color parameters to 0, like this: rgb(0, 0, 0).
- To display white, set all color parameters to 255, like this: rgb(255, 255, 255).
- In rgba (red,green,blue,alpha).alpha means opacity of the color.
- The alpha value is a number between 0.0 means full transparent and 1.0 not transparent.
- Color:rgb(255,255,0)
- Color:rgba(160,200,100,0.4)

text-align : property is used to set the horizontal alignment of a text.

- A text can be left or right aligned, centered, or justified.

text-transform: property is used to specify uppercase and lowercase letters in a text.

- It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

text-shadow: property adds shadow to text

- In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px) (text-shadow: 2px 2px;)
- add a color (red) to the shadow (text-shadow: 2px 2px red;)
- add a blur effect (5px) to the shadow(text-shadow: 2px 2px 5px red)

letter-spacing: property is used to specify the space between the characters in a text.

word-spacing: property is used to specify the space between the words in a text.

text-indention: property is used to specify the indentation of the first line of a text

text-decoration :

- text-decoration-line : property is used to add a decoration line to text.(overline,underline,line-through)
- text-decoration-color: property is used to set the color of the decoration line(red).
- text-decoration-style: property is used to set the style of the decoration line(solid,dashed,double,dotted,wavy).
- text-decoration-thickness: property is used to set the thickness of the decoration line(2px).
- text-decoration: It is a shorthand property for all these four properties

text-decoration-line (required)

text-decoration-color (optional)

text-decoration-style (optional)

text-decoration-thickness (optional)

line-height**:-** The line-height property specifies the height of a line. we pass values in px, deg, decimal.

box-shadow**:-** The CSS box-shadow property is used to apply one or more shadows to an element.

This property will accept five values. h-shadow v-shadow blur-radius spread-radius color of the shadow.

*Default shadow color is text-color. we can pass negative values too.

Background Property: -

background-color: property sets the background color of an element.

- background-image : url (“image.jpg”)
  - property sets one or more background images for an element.
  - By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.
- background-repeat : no-repeat/repeat-x/Y
- property sets if/how a background image will be repeated.
- By default, a background-image is repeated both vertically and horizontally.
- background-size : cover/100%
- property specifies the size of the background images.
- background-Position : right/left/center
- property sets the starting position of a background image.
- background-attachment : scroll/fixed
- property sets whether a background image scrolls with the rest of the page, or is fixed.

Gradient

CSS gradients is used to display smooth transitions between two or more specified colors.

1. **Linear-gradient(direction,color-stop1,color-stop2);**
2. **Radial-gradient(shape size at position, start-color,…,last-color);**

Height and Width Property:-

The height and width properties are used to set the height and width of an element.

The height and width properties may have the following values:

- auto - This is default.
- length - The height/width.
- % - The height/width in percent of the containing block
- initial – Automatically set the height/width to its default value
- inherit - The height/width will be inherited from its parent

max-width:px, max-height:px, min-width:px, min-height:px, width:px, height:px

Font Properties

CSS fonts control how text appears on a webpage.

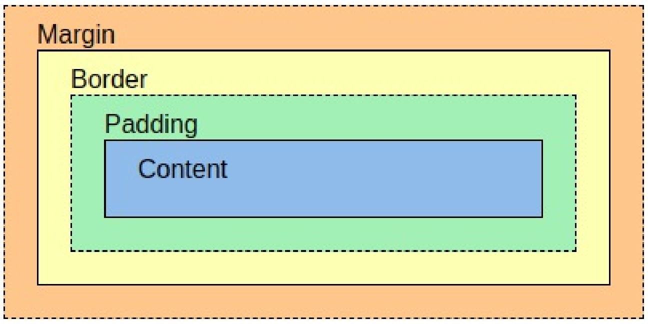
Font-size: large/smaller/medium/small/x-small/xx-small/x-large/xx-large/px/deg.

Font-weight: bold/bolder/lighter/normal/100 to 900

Font-style: italic/normal/oblique

Font-family: font styles

Box Model:-



• The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image illustrates the box model:

• Content - The content of the box, where text and images appear

• Padding - Clears an area around the content. The padding is transparent no negative values.

• Border - A border that goes around the padding and content

• Margin - Clears an area outside the border. The margin is transparent.negative values will be accepted.

• Margin-style

• Margin-top

• Margin-bottom

• Margin-left

• Margin-right

• Padding-style

• Padding-left

• Padding-right

• Padding-top

• Padding-bottom

• Border-style

• Border-color: Color of the border. Default color will be text color(this property is optional)

• Border-width: Thickness of border. Default thickness will be 2px(this property is optional)

• Border-style: dotted/double/groove/dashed/solid(this property is mandatory)

• Border-radius: Convert sharp edges to curve shape.

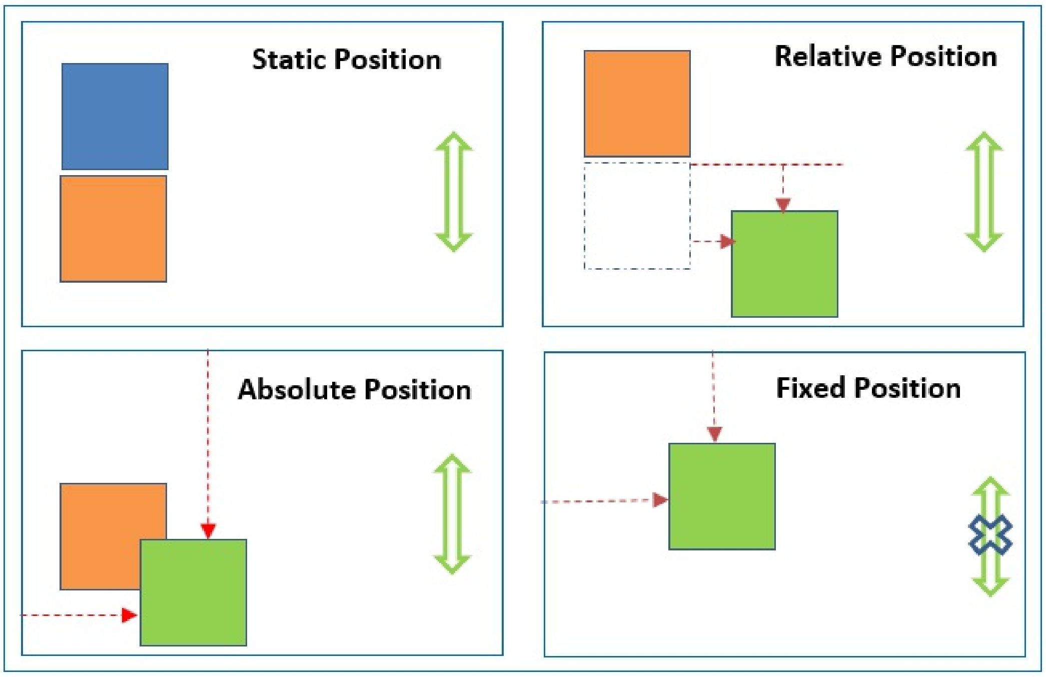
Ex:-border-radius:px/deg.

Position Property:-

The position property specifies the type of positioning method used for an element.

You must specify at least one of top, right, bottom or left for positioning to work.

- Static:-Default value for position property
- Relative:-Moves the element from its current position
- Fixed:-It will be fixed in particular place. which means it always stays in the same place even if the page is scrolled
- Absolute:- Is positioned relative to the nearest positioned ancestor.if no ancestor then it uses document body.
- Sticky:- It is positioned based on the user's scroll. Toggles between relative and fixed.



Display properties:-

Inline :- Displays an element as an inline element (like <span>). Any height and width properties will have no effect

Block :- Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width

Inline-block :-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

Flex :-Displays an element as a block-level flex container

Flex-box properties

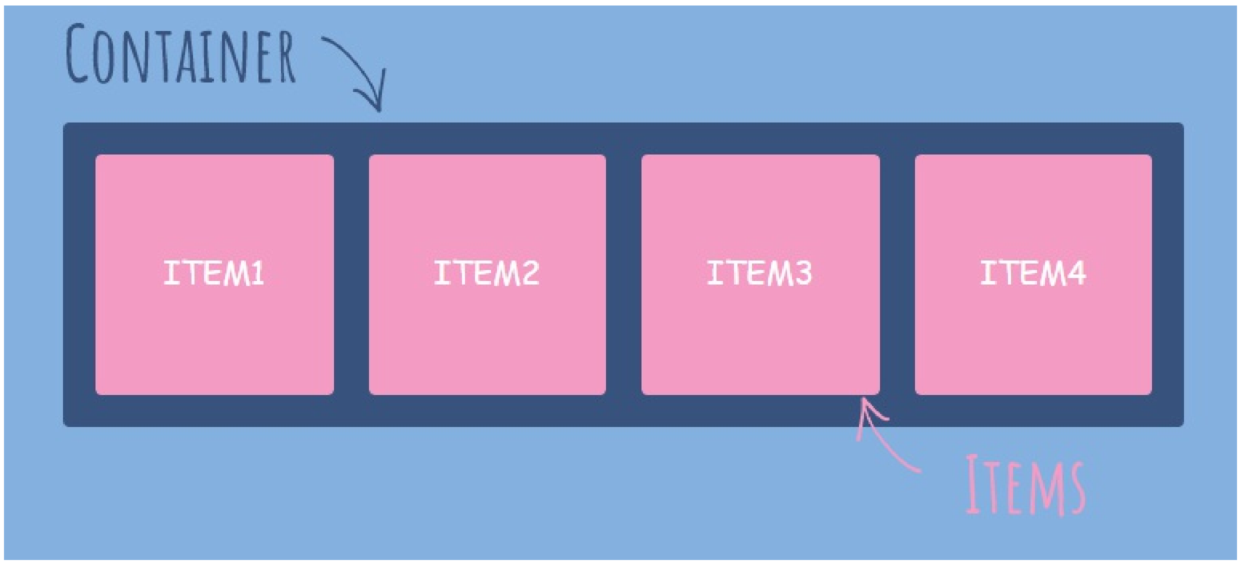
Display :- flex

Flex-direction:- specifies the display-direction of flex items in the flex container. Values are row/column/row-reverse/colomn-reverse

Flex-wrap:- specifies whether the flex items should wrap or not. values are wrap/no-wrap/wrap-reverse

Justify-content:- property is used to align the flex items in horizontally. values are flex-start/flex-end/center/space-around/space-between/baseline

Align-items:- property is used to align the flex items in vertically. values are flex-start/flex-end/center



CSS Grid Layout

The Grid Layout Module offers a grid-based layout system, with rows and columns.

The Grid Layout Module allows developers to easily create complex web layouts.

Grid Container

The rows and columns of a grid is defined with the grid-template-rows and the grid-template-columns properties (or the shorthand grid or grid-template properties).

grid-template-columns: property defines the number of columns in your grid layout, and it can define the width of each column.

grid-template-rows: property defines the height of each row.

Grid Items

grid-column: property defines on which column(s) to place an item.

You define where the item will start, and where the item will end.

grid-row: property defines on which row to place an item.

You define where the item will start, and where the item will end.

grid-area: property can be used as a shorthand property for the grid-row-start, grid-column-start, grid-row-end and the grid-column-end properties.

Transition:

- Transition allows you to change property value smoothly, over a given duration.
- **Transition-property**: The CSS property you want to add an effect
- **Transition-duration:** The duration of the effect
- **Transition-timing-function:** speed curve of the transition effect.
- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- **Transition-delay** : The transition-delay property specifies a delay (in seconds) for the transition effect.
- **Transition(shorthand property)**
- transition: width 2s linear 1s

2D and 3D Transforms

Transforms allow you to move, rotate, scale, skew, element.

- The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.
- It is used to change behavior of an element

2D transform :

translate()

rotate()

scaleX()

scaleY()

scale()

skewX()

skewY()

skew()

3D transform :

rotateX()

rotateY()

rotateZ()

1.translate()-moves an element from its current position.

2.rotate()-rotate an element clockwise/counter-clockwise according to degree.

Using negative values will rotate the element counter-clockwise.

3.scale()- increases/decreases the size of an element.

4.scaleX()-increases or decreases the width of an element.

5.scaleY()- increases or decreases the height of an element.

6.skew()-skews an element along the X and Y-axis by the given angles.

7.skewX()-skews an element along the X-axis by the given angle.[degree 0-360]

8.skewY()-skews an element along the Y-axis by the given angle.

9.matrix()- 6 parameters mathematical function

- **matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())**
- **transform: matrix (1, -0.3, 0, 1, 0, 0);**

3D Transforms:-

- rotateX()- rotates an element around its X-axis.rotateY()-rotates an element around its Y-axis.
- rotateZ()-rotates an element around its Z-axis.

Animations

- CSS allows animation of HTML elements without using JavaScript
- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.
- **@keyframes:-**
- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

- EXAMPLE :

-
  ```
  @keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
  }

  /* The element to apply the animation to */
  div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
  }
  ```

animation-name :Name of the animation

animation-duration: This property defines how long an animation should take to complete, If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds)

animation-delay : Property specifies a delay for the start of an animation.

animation-iteration-count : property specifies the number of times an animation should run.

animation-direction : property specifies whether an animation should be played forwards, backwards or in alternate cycles.

normal - The animation is played as normal (forwards). This is default

reverse - The animation is played in reverse direction (backwards)

alternate - The animation is played forwards first, then backwards

alternate-reverse - The animation is played backwards first, then forwards

animation-timing-function: property specifies the speed curve of the animation

Media queries

- Media query is a CSS technique introduced in CSS3.
- It uses the @media rule to include a block of CSS properties only if a certain condition is true.
- In this example, the only keyword is used to target only screens.
- Example
- If the browser window is 600px or smaller, the background color will be lightblue:

-
  ```
  @media only screen and (max-width: 600px) {
    body {
  ```

```
        background-color: lightblue;
      }
    }
```

JAVASCRIPT

INTRODUCTION TO JAVASCRIPT

- JavaScript is the most popular programming language which is used to create interactive webpages
- JavaScript is the client-server side scripting language which is used to create dynamic web pages
- JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.
- ECMA-262 is the official name of the standard. ECMAScript is the official name of the language.
- Current version of JS is ES6 (introduced after 2015)

Characteristic of JavaScript

- **Client-side scripting language**: The code is readable analyzable and executable by browser itself
- **Interpreted Language**: JavaScript is an interpreted language, meaning it is executed line by line at runtime.
- **Dynamically types Language**: JavaScript is dynamically-typed, which means you don't need to specify the data type of a variable when declaring it. The type is determined at runtime.
- **Weakly Typed Language**: Semicolon is not mandatory at all the time
- **Synchronous in nature**: It will execute line by line; we can make it as asynchronous using promise and async await

Advantages of JS

- Easy to learn and implement
- Reduces server load
- Efficient performance
- Regular updates
- Platform independent
- It has more frameworks and libraries

Environmental Setup

- We can run JavaScript code in two environments.
  - Browser
  - Node.js
- Browser understand the js through JS engine and every browser consists of JavaScript engine which helps to run JS code

Browser	**JS engine**
Chrome	V8
Mozilla Firefox	Spider monkey
Microsoft Edge	Chakra
Safari	JavaScript Core Webkit

How to include JS into HTML

We can insert JS code to HTML in two ways

1.Internal JS: In the HTML file we use <script></script> tags and we will write JS code

2.External JS : We will create new file with .js extension and we can link this to html file using <script src='app.js'></script>

Output Methods

- console.log("hello")

It will print the output in console window

- document.write("Hello")

It will print the output in user interface

- document.writeln("Hello")

It will print the output in user interface

Tokens

Building blocks of every programming language

1. Keywords: Predefined words whose task is already predefined

Ex: for, if, while, var, let, const

1. Identifiers: The name given for the variables, arrays, objects etc.
2. Literals: The values written by developer

Variable in JS

- Variable is a named memory allocation to store the data
- In JavaScript We have 3 keywords to declare variables (var, let, const)
- Syntax To declare variable:

var/let/const identifier;

var/let/const identifier=value;

EX: let age = 27

Here let is keyword, age is identifier and 27 is literal

Difference between var, let and const

	var	let	const
Declaration	var a	let b	const c = 40
Initialization	a = 30	b = 45	NP
Re-initialization	a = 700	b = 90	NP
Re-declaration	var a;	NP	NP
Scope	global	script	script

Hoisting

- Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).
- In JavaScript, a variable can be declared after it has been used.

Example:

console.log(myVar); // Outputs: undefined

var myVar = 42;

The let and const Keywords

- Variables defined with let and const are hoisted to the top of the block, but not *initialized*.
- Meaning: The block of code is aware of the variable, but it cannot be used until it has been declared.
- Using a let variable before it is declared will result in a Reference Error.
- The variable is in a "temporal dead zone" from the start of the block until it is declared:
- A **temporal dead zone (TDZ)** is the area of a block where a variable is inaccessible until the moment the computer completely initializes it with a value.

Datatypes in JS

Datatype defines the type of data to be stored in a particular memory allocation

JavaScript has 2 types of Datatypes

1.primitive

- string: We can declare the string in js in 3 ways

Example: const emp = 'suman',

var str =" hello"

let str = `Hello everyone`

- number: Integer and floating-point numbers

Ex: let num = 80

var salary = 90008.67

- boolean: True or false

Ex: var isDeveloper = true

- bigint: large amount of data
  - to create bigInt variable suffix number with 'n'

Ex: var hike = 8798798n

- undefined: It represents the value of an uninitialized values

Ex: let project;

- null: It is empty variable and for future use

Ex: let company = null

- symbol: It will create function
  - Ex: let work = symbol

2.Non primitive

A function

An object

An array

The typeOf() operator

- This is used to check the which type of data is stored in the variable

Ex: typeOf(age)

JavaScript Template Strings

- Also known as String Templates, Template Strings and Template Literals
- **Template Strings** use back-ticks (``) rather than the quotes ("") to define a string

Ex: let text = `Hello World!`;

- **Template Strings** allow both single and double quotes inside a string:
  - let text = `He's often called "Johnny"`;
- **Template Strings** allow multiline strings:
  - let text =
    `The quick
    brown fox
    jumps over
    the lazy dog`;
- Template String provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

The syntax is: ${var_name}

Variable Substitutions

Template Strings allow variables in strings:

let firstName = "John";
let lastName = "Doe";
let text = `Welcome ${firstName}, ${lastName}! `;

let res = ` ${50+67} `;

Operators in JS

- Predefined symbols used to perform particular task
- Types of operators
- Arithmetic Operators: +, -, *, /, %, ++, --
- Assignment Operators: +=, -=, *=, /=, %=, =
- Comparison Operators: >, <, !=, >=,<=,==,===
- Logical Operators: &&, ||, !
- Ternary Operators:

The ternary operator is a simplified conditional operator like if / else.

Syntax: condition? <expression if true>: <expression if false>

Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

Loops in JS

Loops can execute a block of code a number of times

while loop : Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body

Do while loop : It is more like a while statement, except that it tests the condition at the end of the loop body.

For loop : Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable

Functions In JS

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- With functions you can reuse code
- You can write code that can be used many times.
- You can use the same code with different arguments, to produce different results.
- **Syntax :**

```
function myFunction(p1, p2) {
  return p1 * p2;
}
```

- A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas:
  (*parameter1, parameter2, ...*)
- The code to be executed, by the function, is placed inside curly brackets: **{}**

Types of Functions in JS

1. Anonymous function
2. Named function
3. Function with expression
4. Nested function
5. Immediate invoking function
6. Arrow function
7. Higher order function
8. Callback function
9. Generator function

1.Anonymous function:

A function without name is known as Anonymous function

Syntax:

```
function(parameters) {

// function body

}
```

2. Named Function

A function with name is called as named function

Syntax :

```
function functionName(parameters) {

// function body

}
```

3 . Function with expression

It is the way to execute the anonymous function

Passing whole anonymous function as a value to a variable is known as function with expression.

The function which is passed as a value is known as first class function

EX : let x=function(){

//block of code

}

x();

4.Nested function

- A function which is declared inside another function is known as nested function.
- Nested functions are unidirectional i.e., We can access parent function properties in child function but vice-versa is not possible.
- The ability of js engine to search a variable in the outer scope when it is not available in the local scope is known as lexical scoping or scope chaining.
- Whenever the child function needs a property from parent function the closure will be formed and it consists of only required data.
- A closure is a feature of JavaScript that allows inner functions to access the outer scope of a function. Closure helps in binding a function to its outer boundary and is created automatically whenever a function is created. A block is also treated as a scope since ES6. Since JavaScript is event-driven so closures are useful as it helps to maintain the state between events.

function parent(){

let a=10;

function child(){

let b=20;

console.log(a+b);

}

Child();

}

Parent();

JavaScript currying:

Calling a child function along with parent by using one more parenthesis is known as java script currying

Example:

Function parent(){

let a=10;

function child(){

let b=20;

console.log(a+b);

}

return child;

}

parent()(); ==➔JavaScript currying

5.Immediate invoking function(IIF)

- A function which is called immediately as soon as the function declaration is known as IIF
- We can invoke anonymous function by using IIF

Example:

(function(){

console.log("Hello");

})();

6.Arrow function

- It was introduced in ES6 version of JS.
- The main purpose of using arrow function is to reduce the syntax.
- If function has only one statement then block is optional.
- It is mandatory to use block if function consists of multiple lines of code or if we use return keyword in function.
- Example:

```
Let x=(a,b)=>console.log(a+b);

Let y=(a,b)=>{return a + b };
```

7.Higher order function(HOF)

- A function which accepts a function as a parameter is known as HOF
- It is used to perform multiple operations with different values.
- Example:

```
Function hof(a , b , task ){

Let res=task(a,b);

return res;

};

Let add=hof(10,20,function(x , y){

return x + y;

}

Let mul = hof(10,20,function(x , y){

return x*y;

}

Console.log(add());

Console.log(mul());
```

8.Callback function

- A function which is passed as an argument to another function is known as callback function.
- The function is invoked in the outer function to complete an action
- Example

```
function first(){

Console.log("first");

}

function second(){

Console.log("third");

}

function third(callback){

Console.log("second");

Callback()

}

first();

third(second);
```

1. **Generator function**

- A generator-function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return.
- Example:

```
Function* gen(){
```

Yield 1;

Yield 2;

…….

………

}

Let res=gen();

Console.log(res.next().value);

Console.log(res.next().value);

Console.log(res.next().value);

JAVASCRIPT ARRAYS

ARRAYS:- Array is collection of different elements. Array is heterogenous in nature.

- In JavaScript we can create array in 2 ways.

1.By array literal

2.By using an Array constructor (using new keyword)

1) JavaScript array literal: -

- The syntax of creating array using array literal is:

var arrayname = [value1, value2......valueN];

- As you can see, values are contained inside [ ] and separated by , (comma).
- The .length property returns the length of an array.

2) JavaScript array constructor (new keyword):-

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

var emp=new Array("Jai","Vijay","Smith");

Java script Array methods

- push() : It will insert an element of an array at the end

flipkart.push("laptop","tv");

- unshift() : It will insert an element of an array at the first

flipkart.unshift("Shirts","jacket")

- pop() : It will remove elements of array from the end

flipkart.pop();

- shift() : It will remove elements of array from the start

flipkart.shift();

- indexOf() :It will return a index of particular element

console.log(flipkart.indexOf("glossery"));

- Includes() : It will check whether the particular element is present in array or not.

console.log(flipkart.includes("footwares"))

- At() : It will return the element which is present in particular index.

console.log(flipkart.at(3))

- Slice() : It will give slice of an array and it will not affect the original array.

let newArr=flipkart.slice(1,4);

console.log(newArr);

console.log(flipkart);

- Splice() : It is used to add or remove elements from an array

flipkart.splice(2,0,"bluetooth","kurtha")

console.log("after splicing");

console.log(flipkart);

- Join() : It is used to join all elements of an array into a string.

console.log(flipkart.join());

console.log(flipkart.join(""));

- Concat() : It is used to join/concat two or more arrays.

console.log(flipkart.concat(amazon));

- toString() : It converts all the elements in an array into a single string and returns that string.

console.log(flipkart.toString());

- Split() : It is used to split a string into an array.

let str="Hello";

console.log(str.split(""));

- Reverse() : It is used to reverse an array.

console.log(flipkart.reverse());

- Array.from() : It will convert string into an array

filter(), map(), reduce()

filter, map and reduce are all array methods in JavaScript. Each one will iterate over an array and perform a transformation or computation. Each will return a new array based on the result of the function.

1. **filter():**

The filter() method in JavaScript is used to create a new array with all elements that pass a test specified by a provided function. It does not modify the original array.

Example:

let newArray = array. filter(callback(element, index, array){

thisArg;

}

callback: This is the function that is used to test each element in the array. It takes three arguments:

element: The current element being processed in the array.

index (optional): The index of the current element being processed.

array (optional): The array filter was called upon.

thisArg (optional): Value to use as this when executing callback.

2.map():

The map() method in JavaScript is used to create a new array by applying a function to every element in an existing array. It does not modify the original array. Instead, it returns a new array with the modified elements.

EX:

const numbers = [1, 2, 3, 4, 5];

const doubledNumbers = numbers. Map(number => number * 2);

```
console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]
```

3.reduce():

The **reduce()** method in JavaScript is used to apply a function to an accumulator and each element in an array, in order to reduce it to a single value. It iterates over each element in the array and accumulates a value based on the provided function.

Ex :

```
const numbers = [1, 2, 3, 4, 5];

const sum = numbers.reduce(function(accumulator, currentValue) {

return accumulator + currentValue;

}, 0);

console.log(sum); // Output: 15
```

Objects in JavaScript

- Object is nothing but the thing which has existence in the world.
- A JavaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- In java script object is used to store the data in key value pairs.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- Objects are mutable.
- We can access, update, add and delete the properties from an object.

Creating Objects in JavaScript:

There are 3 ways to create objects.

1.By object literal.

2.By creating instance of Object directly (using new keyword).

3.By using constructor function

1.By object literal.

The syntax of creating object using object literal is given below:

```
VariableName object=

{

property1:value1,

property2:value2,

propertyN:valueN

}
```

As you can see, property and value is separated by : (colon).

Example of creating object in JavaScript.

```
let emp={

id:102,

name:"Kumar",

salary:40000

}
```

2. By creating instance of Object directly (using new keyword).

The syntax of creating object directly is given below:

```
var objectname=new Object();
```

Here, **new keyword** is used to create object.

Example of creating object directly.

var emp=new Object();

emp.id=101;

emp.name="Ravi";

emp.salary=50000;

3.By using constructor function

- Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.
- The **this keyword** refers to the current object.

In JavaScript, there are two main ways to access and manipulate object properties: dot notation and bracket notation.

1. Dot notation

Dot notation is the most common way to access object properties. It uses a period (.) to access the value of a property by its key.

Here's an example:

const person = {

name: 'John',

age: 30,

address: {

street: '123 Main St',

city: 'New York'

}

};

console.log(person.name); // John

console.log(person.address.city); // New York

2.Bracket notation:

Use square brackets [] to access the value of a property by its key.

Here's an example:

console.log(person['name']); // John console.log(person['address']['city']); // New York

OBJECT METHODS

- Objects can also have methods.
- Methods are actions that can be performed on objects.

OBJECT METHODS:

1.keys: It will return array of keys

console.log(Object.keys(mobile)); //array of keys

2.Values: It will return array of values

console.log(Object.values(mobile)); //array of values

3.Entries: It will return array of keys and values

console.log(Object.entries(mobile)); //array of keys and values

4.Assign: It is used to merge two objects

Object.assign(mobile,mobile1)

console.log(mobile);

let resObj=Object.assign({},mobile,mobile1);

```
console.log(resObj);

console.log(mobile);

console.log(mobile1);
```

5.Seal: We can only update the properties

```
Object.seal(mobile);

// console.log(mobile.brand); //access

// mobile.brand="redmi"        //updation

// console.log(mobile.brand);

// mobile.battery="5000mah"  //not able to add prop

// console.log(mobile);

// delete mobile.cost;        //not able to delete

// console.log(mobile);
```

6.isSealed: It is used to check whether the particular object is sealed or not

```
// console.log(Object.isSealed(mobile));
```

7.Freeze: We cannot do any modifications in an object

```
// Object.freeze(mobile);

// console.log(mobile.color);  //access

// mobile.color="grey";        //cannot update

// console.log(mobile.color);

// mobile.battery="6000mah";  //cannot add

// console.log(mobile);

// delete mobile.camera;   //cannot delete

// console.log(mobile);
```

8.Isfrozen: It is used to check whether the particular object is frozen or not

```
// console.log(Object.isFrozen(mobile));
```

9.hasOwn: It is used to check whether the particular property is present in an object or not.

```
// console.log(mobile.hasOwn("cost"));   //true

let obj={

    firstName:"Mahesh",

    lastName:"Babu",

    eid:123,

    sal:25000,

    address:{

        city:"bng",

        pin:560021

    },

    fullname:function(){

        console.log(`${this.firstName} ${this.lastName}`);

    }
}

obj.fullname();
```