**DOM (Document Object Model)**

By using HTML DOM, JavaScript can access and change all the elements of an HTML document.

The W3C Document Object Model(DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.

The HTML DOM is a standard object model and programming interface for HTML. It defines:

* The HTML elements as objects
* The properties of all HTML elements
* The methods to access all HTML elements
* The events for all HTML elements

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

* JavaScript can change all the HTML elements in the page
* JavaScript can change all the HTML attributes in the page
* JavaScript can change all the CSS styles in the page
* JavaScript can remove existing HTML elements and attributes
* JavaScript can add new HTML elements and attributes
* JavaScript can react to all existing HTML events in the page
* JavaScript can create new HTML events in the page

# Methods:-

HTML DOM methods are **actions** you can perform on HTML Elements.

HTML DOM properties are **values** of HTML Elements that you can set or change.

The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as objects.

The programming interface is the properties and methods of each object.

A property is a value that you can get or set (like changing the content of an HTML element).

A method is an action you can do (like add or deleting an HTML element).

e.g. <html>  
<body>  
  
<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML = "Hello World!";  
</script>  
  
</body>  
</html>

**The getElementById Method:-**

The most common way to access an HTML element is to use the id of the element.

In the example above the getElementById method used id="demo" to find the element.

**The innerHTML Property:-**

The easiest way to get the content of an element is by using the innerHTML property.

The innerHTML property is useful for getting or replacing the content of HTML elements.

The innerHTML property is used to get or change any HTML element including <html> and <body>.

## DOM Document Object:-

The document object represents your web page.

If you want to access any element in an HTML page, you always start with accessing the document object.

## Finding HTML Elements:-

document.getElementById(id):- Finds an element by id.

document.getElementsByTagName(name):- Find elements by tag name.

document.getElementsByClassName(name):- Find elements by class name.

## Finding HTML Elements by CSS Selectors:-

If you want to find all HTML elements that match a specified CSS selector use the querySelectorAll() method.

## Changing HTML Elements:-

## Properties:-

## element.innerHTML=new html content:- Change inner HTML of an element.

## Syntax:- document.getElementById(id).innerHTML = new HTML

## element.attribute=new value:- Change the attribute value of an html element.

## Syntax:- document.getElementById(id).attribute = new value

## element.style.property=new style:- Change the style of an html element.

Syntax:- document.getElementById(id).stye.cssproperty = value

Creating dynamic html content:-

e.g. document.getElementById("demo").innerHTML = "Date : " + Date();

**Validation:-**

It is the process of checking whether the entered data is correct or not.

All fields can be validated to check that they should not be kept empty. Numeric and text fields can be validated to check length or to check whether text or numbers are entered.

By default some validations are provided by using input type like email or date.

**BOM(Browser Object Model)**

**Window Object:-**

Window object represents browser’s window. Window object is supported by all browsers. All JavaScript objects, functions, and variables are members of the window object. variables are properties of the window object & functions are methods of the window object.

**Properties of window object:-**

* window.innerHeight - the inner height of the browser window (in pixels)
* window.innerWidth - the inner width of the browser window (in pixels)

**Window Methods:-**

* window.open() - open a new window
* window.close() - close the current window
* window.moveTo() - move the current window
* window.resizeTo() - resize the current window
* window.name-displays name of window
* window.status:-displays status bar information
* window.location:- displays location of the window.

**Asynchronous Javascript:-**

**Callbacks**:- JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined. You can control the sequence by using callbacks. A callback is a function passed as an argument to another function.

e.g. function myDisplayer(something) {

document.getElementById("demo").innerHTML = something;

}

function myCalculator(num1, num2, myCallback) {

let sum = num1 + num2;

myCallback(sum);

}

myCalculator(5, 5, myDisplayer);

**Asynchronous Functions:-**

**SetTimeout():-**you can specify a callback function to be executed on after some time. E.g. setTimeout(a,3000)

Time is set in form of milliseconds.

**SetInterval():-**you can specify a callback function to be executed for each interval. E.g setInterval(myFunction, 1000);

**ClearInterval():-**clears a timer set with the setInterval() method.

E.g. myInterval = setInterval(function, milliseconds);

clearInterval(myInterval);

# Arrow Function:-

Arrow functions allow us to write shorter function syntax

e.g. let myFunction = (a, b) => a \* b;

or

hello = () => {  
  return "Hello World!";  
}

### Arrow Functions Return Value by Default

### Arrow Function With Parameters:

hello = (val) => "Hello " + val;

### Arrow Function Without Parentheses:

hello = val => "Hello " + val;

# this Keyword:-

In JavaScript, the this keyword refers to an **object**.

The this keyword refers to different objects depending on how it is used:

In an object method, this refers to the **object**.

Alone, this refers to the **global object**.

In a function, this refers to the **global object**.

e.g. this in a object

const **person** = {  
  firstName: "Sarika",  
  lastName : "Kore",  
  id       : 5566,  
  fullName : function() {  
    return **this**.firstName + " " + **this**.lastName;  
  }  
};