***JAVASCRIPT***

JavaScript was invented by **Brendan Eich** in 1995. JavaScript or JS is a **lightweight, interpreted, object-oriented language, client-side** scripting language and widely used in dynamic websites. The script can be embedded within the HTML or stored in an external file.

Server-side Scripting : Python, PHP, Ruby, etc

Client-side Scripting : HTML, CSS, JavaScript, VBScript, etc

**ClientSide vs ServerSide**

1. ServerSide scripting is done at the backend where the source code is hidden on the client side(Browser) and the ClientSide scripting is done at the front end which users can access or manipulate them on Browser.
2. ServerSide scripting is mostly done for personalizing web pages(eg. Gmail, Facebook) for implementing dynamic webpages whereas ClientSide scripting is done to minimize the server load (eg. Gmail SIGN UP Form).
3. ClientSide scripting works faster than ServerSide Scripting.
4. ClientSide scripting is less secure than ServerSide scripting because users has the access of client side script.
5. ServerSide script is run on remote local server and ClientSide script is run on local computer.

**Variables in JS**

* Variable is an container to store or hold the values.
* In JS, there are 3 types of variables:

1. By using **var** keyword. (From 1995)
2. By using **let** keyword. (From 2015, introduced in ES6)
3. By using **const** keyword. (From 2015, introduced in ES6)

* **var** keyword:

var keyword is used to tell the JS that variable is being declared.

It has Global and Functional scope.

It is hoisted.

var a = 10;

* **let** keyword:

let keyword allows us to declare variables that are limited to the scope of a block statement.

let b = 20;

* **const** keyword:

const keyword is used to declare constants ie. The const variables can’t be reassigned. const declared variables also have the block scope.

const pi = 3.14;

**Functions in JS**

A function is a block of code which is design to perform a particular task.

#Types of Functions:

1. Non – Parameterized Functions.
2. Parameterized Functions.
3. Function Expression(Anonymous Function, Arrow Function).
4. **Non – Parameterized Function:**

The functions with no parameters ie. the paranthesis after the function name is empty.

Syntax:

function <function\_name> ()

{

#Function Body

}

1. **Parameterized Function:**

The function with parameters defined at the time of function declaration.

Syntax:

function <function\_name> (parameters)

{

#Function Body

}

1. **Function Expression:**

A function expression is very similar to and has almost the same syntax as a function declaration.

The main difference between a function expression and a function declaration is the function name, which can be omitted in function expressions to create anonymous functions.

**Arrow Function:**

Arrow function is one of the features introduced in the ES6 version of JavaScript.

It allows you to create functions in a cleaner way compared to regular functions.

The main advantage of using arrow function is it’s **shorter syntax** and **require less code.**

Syntax:

1. **Arrow function with No Parameter**

let x = () => console.log(“Hello World”);

1. **Arrow function with One Parameter**

let x = name =>console.log(“Hello”, name);

1. **Multi – Line Arrow Function**

let add = (a, b) => {

res = a + b;

console.log(“Addition is:”, res);

}

**Debugger**

The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.

Using debugger, we can set breakpoints in the JavaScript code.

**DOM**

It refers to Document Object Model.

Document 🡪 The HTML Page(document).

Object 🡪 Elements and Attributes in HTML.

Model 🡪 Tree Structure of HTML elements.

DOM is an programming interface for web documents.

It represents the page so that programs can change and manipulate the document structure, style and content.

The DOM represents HTML document as node objects.

**DOM Methods:**

1. **getElementById –** It is used to fetch HTML nodes by id attribute.
2. **getElementsByClassName** **–** It is used to fetch HTML nodes by class attribute.
3. **getElementsByName –** It is used to fetch HTML nodes by name attribute.
4. **getElementsByTagName –** It is used to fetch HTML nodes by Tag Names.
5. **querySelector –** It returns the first match of an element that is found within the

HTML document.

1. **querySelectorAll –** It returns a node list of the objects specified selector in the

method.

**BOM**

It refers to Browser Object Model.

BOM is used to interact with the browser with the help of window object.

Window object represents a window in browser.

The window object is automatically created by browser.

All the Global javascript objects, functions and variables with var keyword automatically become members of the window object.

Global variables are properties and Global functions are methods of window object.

Eg. alert, prompt, confirm

**Template Literals**

Template literals provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.

**Syntax: `**${expression}**`**

**Array**

An array is a collection of multiple data elements.

Array is represented by [ ] (square brackets).

Array in JS supports positive indexing.

**Syntax:**

1. datatype <array\_name> = [ele1, ele2, ele3, ….];
2. datatype <array\_name> = new Array(ele1, ele2 , ele3, ….);

**Array methods in JS**

**INSERTION Methods (Pushing)**

1. **push():**

The push method is used to add a single element at the end of Array.

Array.push(element)

1. **shift():**

The shift method removes the first element from the array and returns that value.

Array.shift(element)

1. **unshift():**

The unshift method adds an new element at the beginning of the Array and returns the array length.

Array.unshift(element)

1. **length:**

The length property returns the length of the Array.

Array.length

**Merging (Concatenating) Arrays**

1. **concat():**

The concat method returns a new array by merging the existing arrays.

datatype Array = Arr1.concat(Arr2, Arr3)

1. **sort():**

The sort methods sort the array alphabetically.

By default, the sort method sort values as strings.

Array.sort()

**For Numeric Sorting:**

Array.sort(function(a, b) { return a - b});

1. **reverse():**

The reverse method reverse the elements in an Array.

Array.reverse();

1. **forEach():**

The forEach() method calls a function once for each array element.

Array.forEach(function\_name);

Note: The function can have 3 parameters:

1. value
2. index
3. array
4. **map():**

The map methods calls a function once for each element in an array.map().

It does not execute the function for empty elements.

Map() methods does not change the Array.

Array.map(function\_name);

1. **filter():**

The filter methods create an new array with array elements that pass a test.

const Array\_name = Array.filter(function\_name);

**Object**

JavaScript object allows us to store multiple collections of data.

An object is an entity having state and behaviour (properties and methods).

Object values are written as name : value pairs (name and value is separated by colon).

Ways of Creating objects in JS:

1. By Object Literal.
2. By creating instance of Object class.
3. By using Object Constructor.
4. **By Object Literal:**

It is the easiest way to create an object.

Object literal is an list of name: value pairs.

**Syntax:**

const obj = {prop1: value1, prop2: value2, prop3: value3, …, propN: valueN};

1. **By Creating an instance of Object Class:**

By using the **new** keyword we can create object.

**Syntax:**

const obj = new Object();

obj.prop1 = value1;

obj.prop2 = value2;

1. **By using Object Constructor:**

**this** keyword refers to an object(like self in Python).

**Syntax:**

function m1(v1, v2, v3)

{

this.prop1 = v1;

this.prop2 = v2;

this.prop3 = v3;

}

Obj = new m1(val1, val2, val3);

**Loops**

Loops are use to run the same code over and over again, each time with a different value.

**Syntax:**

for( initialization; termination; steps)

{

//Loop Body

}

1. **For In Loop**

for in statement loop over the properties of an Array or object.

for..in iterates over all enumerable property keys of an object.

**Syntax:**

for(let variable in Array or Object)

{

//Loop Body

}

1. **For of Loop**

for..of iterates over the values of an iterable object. Examples of iterable objects are arrays, strings, and NodeLists.

**Syntax:**

for(let variable of Array or Object.values(obj))

{

//Loop Body

}

**Spread Operator:**

Spread operator (...) allows us to quickly copy all or part of an existing array or object into another array or object.

**Syntax:**

const arr1 = [10, 20, 30, 40, 50];

const arr2 = [1, 2, 3, 4, 5];

const arr3 = […arr1, …arr2];

**Destructuring of Array and Object:**

Destructuring in JavaScript allows us to unpack values from Objects, Arrays and store them into variables.

## **Array Destructuring**

const arr = [100, 200, 300];

const [v1, v2, v3] = arr;

## **Object Destructuring**

const obj = {“name”: “Joe”, “sal”: 5678};

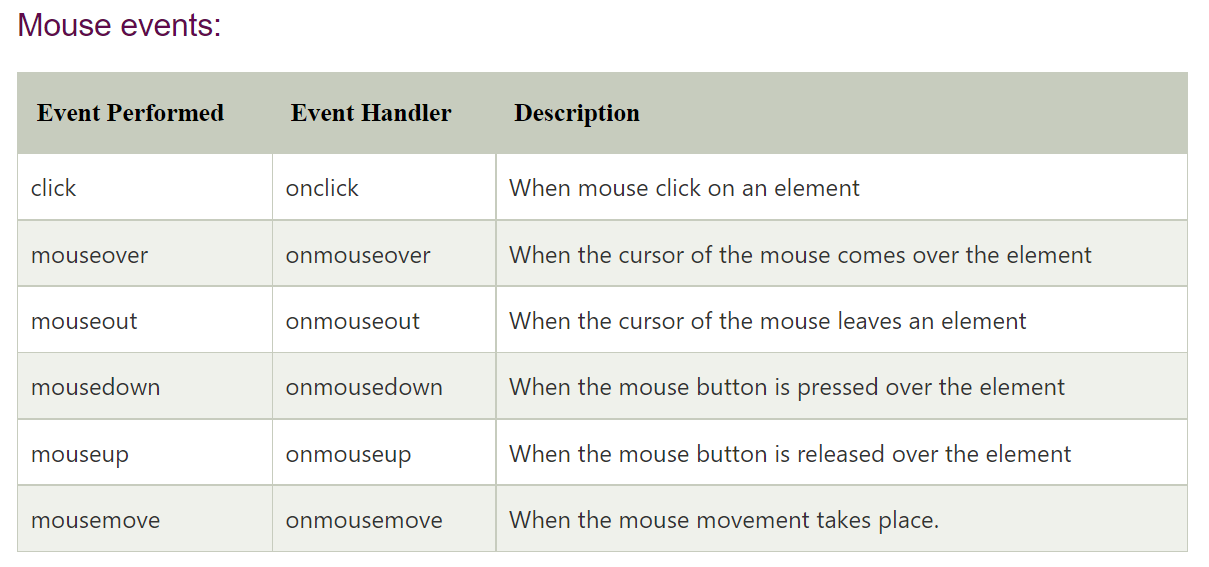
const {name, sal};

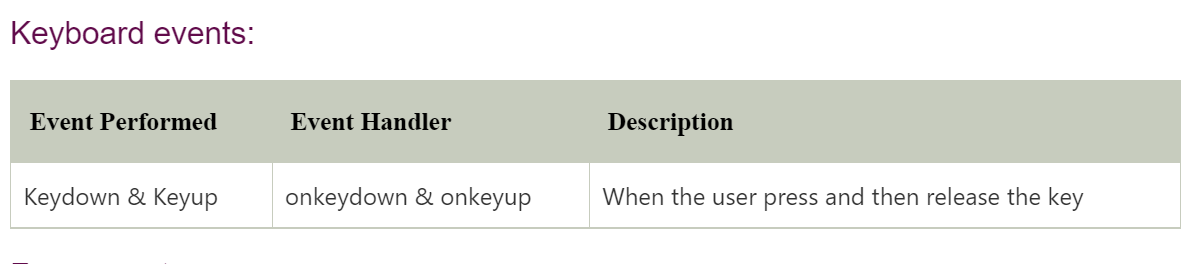
**Note:** In object destructuring the name of variables should be same as property names of objects.

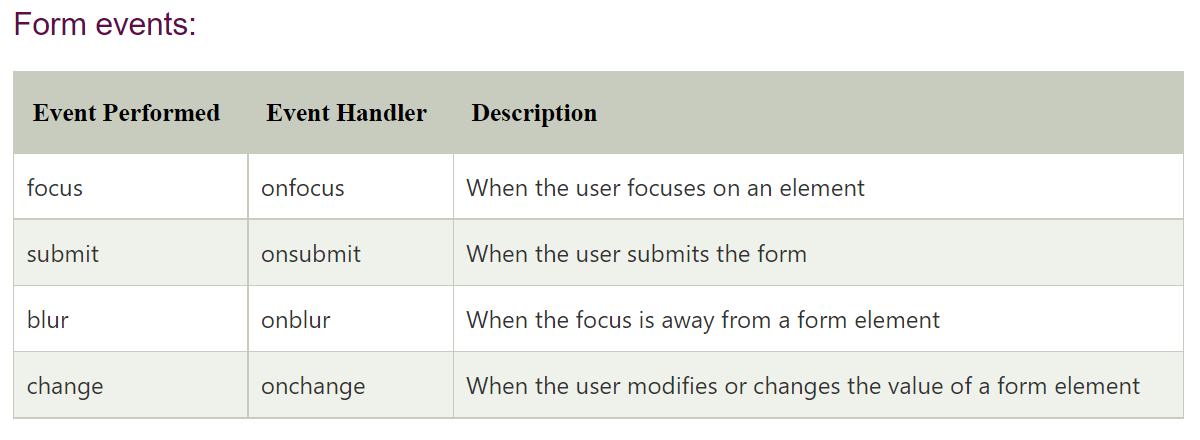
**Events:**

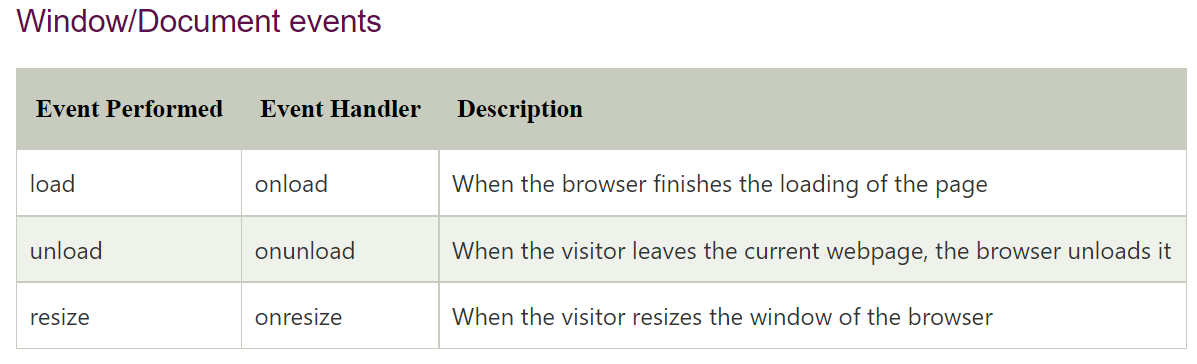
Events are actions that happen when a user interacts with the page - like clicking an element, typing in a field, or loading a page.

The browser notifies the system that something has happened, and that it needs to be handled. It gets handled by registering a function, called an event handler, that listens for a particular type of event.









**EventListener:**

An event listener in JavaScript is a way that you can wait for user interaction like a click or keypress and then run some code whenever that action happens.

**Syntax:**

element.addEventListener(event, function);