**Javascript**

- JS is a client-side scripting language used to create dynamic web-applications and websites

- It runs in the browser(client-side) and also in our local environment(server-side) with NodeJs

- It adds the functionalities/actions to web-pages and makes them interactive Or it reacts to user-events like click event , request-response from server event , mouse-movement event etc .

- Since it was specificly developed for browser , it doesn’t have low-level access of memory or cpu and hence Javascript is safer though.

- JS gets executed through Browser

**difference b/w Dynamic-website(web-app) and Static-website**

|  |  |
| --- | --- |
| Static site | Dynamic site |
| No request response scenario | It generates data /pages in real-time , as per the request to server . |
| Pages/Data will remain same until someone changes its manually | Content of pages are different for different visitors |
| Less compelexity | High complexity |
| Information changes rarely | Information changes frequently |
| less time for loading | more time for loading |
| Data-base is not used | database is used |

**AJAX(Asynchronous Javascipt and XML)**

- Reads data from a web-server after a web page has loaded

- Update a web-page dynamically without reloading the page

- Send data to a web-server in the background

- It can send and receive informations in various formats like JSON , XML , HTML and text files

**DOM(Document Object Model)**

- When a web-page is loaded, the browser creates DOM of the page . Hence all the HTML elements like , div , nav , body etc comes under DOM

- With HTML DOM , Javascript can access and change all the elements of an HTML document

- The HTML DOM model is constructed as a tree of Objects

HTML DOM Tree of Objects



**Cookies**

- It is a piece of data consisting user’s information(that is stored by the browser on user’s system(l**ocal storage**)) by which the server will recognizes the user when the user sends a request to the server next-time .

- Cookies are saved in name-value pairs . Ex: username = John Koay

**Local-storage**

- It is a property that allows JS sites and apps to save key-value pairs in a web browser with no expiration date , it means that the data stored in the browser will persist even after the browser window is closed .

**Same-origin-policy** - It is a critical security mechanism that restricts how a document or script loaded by one origin can interact with a resource from another origin . like , one website owner can’t make an ajax-request to get it’s user’s facebook details

**What can’t IN-BROWSER JS do?**

- Read/write to and from the system’s hardisk

- Same-origin-policy applied

**What makes JS unique ?**

- HTML/CSS support

- Simple APIs for simple things

- Major modern-browsers support(enabled by-default)

**ECMAScript(European Computer Manufacturers Association Script) - It is a standard for scripting languages like , Javascript ,Jscript , ECMAscript(Google it)**

Important keywords/Functions:-

**document** – It is the keyword allows to get access to HTML page .

**getElementById()** – It allows us to grab any element of HTML by ID .

**console.log(token)** – print items

array.**push(element) -** this method is used to insert any array element into the array

array.**indexof(element) –** it gives the index of the given element in an array

array.**splice(index , number of elements)** – it takes off the given index-element from an array

var x = **window.prompt(“Enter = ”) –** it will receive an user-input and will assign it to the variable x

array.**length** – returns the array length .

***OBJECTS***

**objects –** it is the collection of key-value pair

**var student = {firstName: ”Rahul” , lastName:”Kumar” , age : 12};**

**var student = {} ; <- empty object**

**var x = new Object();** - we can create an empty object by this only . Afterwards , we can add key-

After creating object , we can add key-value to it by following way and same way will be applicable for an empty object.

***x.keyOne = value1;***

***x.keyTwo = value2;***

value pair that empty object

*- We can create javascript objects manually by functions(constructors) also .*

***function Student(first,last,age)***

***{***

***//down below , we are creating keys/properties explicitly and we are adding their values by function-parameters***

***this.firstName = first;***

***this.lastName = last ;***

***this.Age = age ;***

***this.greeting = function(){***

***return "Hello everyone , this is " + this.firstName +" " + this.lastName + " and I am " + this.Age + " years old . " ;***

***}***

***}***

***var studentOne = new Student(“Rahul”,”Kumar”,85); <- This is the object of Student()***

**-** ***In javascript , we create the objects of a function too like in classes***

variable**.value –** value function modifies or set the value of the variable(inputs)

variable**.innerText** – it works as same as **.value** but for variables holding non-input elements .

- target.**addEventListener(event-type , listener)** – this method allows you to set up a function to be called when a specified event happens , such as when user clicks a button .

- **parseFloat()** method turn a string into float

- **console.log()** and **alert()** methods turns anything into strings automatically

- Default behavior of form is that it **refreshes** itself on every submission and wipe all your data off . To prevent this we off this default feature , using **preventDefault()** method .

event.**preventDefault()** – it cancels the event if it is canceleable , or the default action that belongs to the event will not occur .