**JAVASCRIPT**

**1. What JavaScript Really Is**

* Definition: JavaScript (JS) is a high-level, interpreted, single-threaded, prototype-based, multi-paradigm scripting language used mainly for web development.
* Main purpose: It makes web pages dynamic and interactive.
* Runs where: In browsers (via the JavaScript engine — e.g., Chrome’s V8, Firefox’s SpiderMonkey) and outside browsers (via Node.js).
* Multi-paradigm: You can write procedural, object-oriented, or functional code in JS.

**2. Scripting Language vs Programming Language**

* Programming Language: Usually compiled before execution (e.g., C, Java). Produces a standalone executable file.
* Scripting Language: Interpreted at runtime (line-by-line), usually depends on another program/environment to run.  
  Example: JavaScript needs a browser or Node.js to run.
* Why JS is called a scripting language: It was originally built to “script” browsers for small tasks. Over time, it became powerful enough to be considered a full programming language.

**3. “Only JS runs on browsers” — What that means**

* Browsers natively understand only JavaScript (and WebAssembly).
* Languages like Python, Java, PHP, etc., can’t run directly in the browser — they need a server or must be transpiled into JavaScript (e.g., TypeScript → JavaScript).
* This is why JS is the only language for client-side logic on the web without extra tools.

**4. Core JavaScript Facts to Remember**

These often pop up in interviews:

a) JS is Single-Threaded but Asynchronous

* JS runs one piece of code at a time (single-threaded).
* It can handle async tasks (like API calls, timers) without blocking using the event loop.

b) Loosely Typed

* You don’t declare variable types (let x = 10; could later hold a string).
* This makes it flexible but can cause bugs.

**c) Dynamically Typed**

* Type checking happens at runtime, not before execution.

**d) Interpreted & JIT Compiled**

* JS is parsed and executed line-by-line, but modern engines just-in-time compile for speed.

**e) Runs Everywhere**

* Browser: Client-side code (DOM, events, UI logic).
* Server: With Node.js (APIs, backends, databases).

**5. Essential Interview Keywords**

If you’re starting JS, get comfortable with these terms:

* ECMAScript (ES): The standard JS follows (e.g., ES6 = modern features like let, const, arrow functions).
* DOM (Document Object Model): JS’s interface to HTML/CSS.
* Hoisting: Variable and function declarations are moved to the top during execution.
* Closures: Functions remember variables from their outer scope.
* Prototypes: JS’s inheritance system.
* Call Stack & Event Loop: How JS handles synchronous vs async code.

**Important Points**

Code by js is called **Vanilla code**

document.write( “<p>” + var name + “</p>”) : to print on ui

Space is a value is js

Js consider decimal (9.9) and int (9) both datatype as number

If a number is added with any data type except string or number then output is NAN

# is used in anchor tag to stop reload

Regex : used for js validation

Extra space is counted in js

Js is object based programming language

Premitive Datatypes

NonPremitive Datatypes

Modular js is used when importing external files

Class # 05 17/sep/2023

**PROMPT**

Built-in method of js to take input; {work like alert / popup}

Saves data in string datatype

Write +prompt here + will convert datatype to number

Var num1 = prompt (“Enter number1”)

Var userName = prompt (“Enter your name”)

**To enter placeholder**

Var userName = prompt (“Enter your name”, **sarimbaig**)

**CONDITIONS**

If = keyword

() = condition

{} = block of statement

Class # 06 19/sep/2023

Three assignment operators

= => assignment

== => comparison of values

=== => comparison of values & data types

Class # 07 21/sep/2023

**ARRAY**

Array = collection of data

Multiple datatypes can be saved in an array

[ ] = array bracket

Undefined data = it can have many meanings eg.(lead)

Defined data = it have single meaning eg.(city)

Class # 07 23/sep/2023

**ARRAY METHODS**

* Type of: (to check datatype)

**Methods for demicals**

* Math.round (var name) : (to round off a number)
* Math.floor (var name) : (only picks number neglect number after point)
* Math.ceil(var name): (add 1 number, number after point doesn’t matter)
* Pop: (used to remove value from end in an array) {Single value is removed at a time}
* Push: (Used to insert value in end of array) { Multiple values can be added at a time }
* Shift: (used to remove value from start in an array) {Single value is removed at a time}
* Unshift: (Used to insert value in start of array) { Multiple values can be added at a time }
* Slice: get copy of values in and array and takes 2 parameters (index, length/ starting, ending point)
* Splice: Can perform pop push shift unshift slice all together.
  + It takes 3 values (starting point, delete-count, values) (index, quantity, values)

**Case convertions;**

VariableName.toLowercase

VariableName.toUppercase

variableName.split(“ Enter thing from where you want to break”): {used to spit any word for any condition from string to array}

variableName.join(“enter thing you want to insert between”): {used to join two words from array to string}

variableName.trim(remove extra continuous space)

Class # 08 26/sep/2023

**FOR LOOPS**

Don’t use else in loop due to inaccurate answer

Break: (used to break the loop for a certain condition)

Continue: (to skip an item )

Class # 11 10/oct/2023

**METHODS**

Var rn= math.random() : 0.99999-16 decimals

Var rn = math.random() \* 2 : 0.9999 – 1.9999

Var rn = math.floor(math.random() \* 2) : 0-1

Var rn= math.floor(math.random() \* 2 +1) : 1-2

Range, no zero

Number(var name) : to convert any data type into number

parseInt(var name) : to convert in to int

String(var name): to convert any data type into string

Varname.toString():to convert any data type into string

a.toFixed(range of number after point) : to control decimal lengths {it show round off output and convert num in to string}

Class # 12 12/oct/2023

**METHODS**

Var now = new Date() : (method to get recent date and time)

Class # 13 14/oct/2023

**FUNCTIONS**

() = parenthesis

2 types of functions{

Predefined: build in functions of js / or any language

Userdefined: functions made by user or developer}

Class # 14 17/oct/2023

**FUNCTIONS**

Function func(parameters){}

Func(arguments)

DEFAULT PARAMETER:

Function add(num1,num2=0){}

Class # 17 24/oct/2023

**JS EVENTS**

Eval(var name) : {Performs operation of a single string }

Class # 18 26/oct/2023

**DOM**

Document .getElementbyid

Document = html

.value is used only for input values,text area,radio brtn, checkbox,select option, password (text areas)

.innerHTML is used for rest of elements

Onmouseover = “ mouseover(this) : this take the element itself to js

Class # 21 4/nov/2023

**STYLING WITH JS**

Class is default name is js

Heading.className += “ black” : to add a new class in existing one

Heading.classlist.add(“black”,”blue”)

Heading.classList.remove(“black”) : classlist bring all classes applied on an element and remove desired class

**3 WAYS TO APPLY CONDITIONS IN JS**

If/else

Switch case

Ternary operator

SWITCH CASE:

Multiple cases and ranges can’t be define here

It is used because is runs all the cases after any case becomes true

Switch (age){

Case 18:

Console.log(“Allow”)

Break;

Default:

Console.log(“not allow”);

}

Case = if, default = else

Class # 22 8/nov/2023

**Target Elements**

Document.getelementsbytagname(“p”) : it will get all the p tags and save it in an array

Returntype is array

Class # 23 11/nov/2023

**Target Elements**

Var parent = document.getElementsById(“parent”)

Var para = parent.getElementsByTagName(“p”)

{ to get paragraphs from an specific container}

**DOM**

**Document object model**

D => HTML

O => properties, methods

M => logic

childNode = child elements of a container

Node = element

Extra space is counted in js { if lines breaks in html js consider it as an text element called junk artifact}

Var parent = document.getElementById(“parent”)

Var lastChild = parent.previousSibling

Children : will only bring html elements

Class # 25 16/nov/2023

**Getting Target Name**

Var parent = document.getElementById(“parent”)

Console.log(parent.nodeName)

Console.log(parent.nodeType)

Console.log(parent.nodevalue)

Parent.hasAttribute(“placeholder”) : returns Boolean true/false

Parent.setAttribute(attributeName , attributeValue) : (“placeholder” , “enter your name”)

Parent.placeholder = “ enter your name”

**Creating elements using js**

Var element = document.createElement(“h1”)

Var elementValue = document.createTextNode(“Hello World”)

Element.appendChild(elementValue)

Class # 26 18/nov/2023

**Js timing methods**

Two methods related to time:

**setTimeout()** : take input => (function(){}, time in milisecond)

can write function in argument or can just write name of a prebuild function but will not call it otherwise it will run without waiting

**Code Example:**

Console.log(“hello 1”)

Function foo(){

Console.log(hello 2)

}

setTimeout(foo,200)

Console.log(hello 3)

**Output:**

Hello1, hello3, hello2

**JS IS SYNCRONOUS AND WILL NOT WAIT FOR ANY FUNCTION**

**JS have a background where every line of code is send which takes time until then other code is executed. The time bg code time is completed it is executed**

**setInterval() :** take input => (function(){}, time in milisecond)

it run after every given time till infinity and can also be stopped on conditions

Class # 27 21/nov/2023

**Js timing methods**

Intervals are made in a variable and to stop it we use clearInterval( variable name ) method

Class # 28 23/nov/2023

**OBJECTS**

**Ptoperty** = name : value

Var obj = {

Name : “sarim”

Age : “22”

Marks: {

Html : 80,

Css: 90,

Js: 75} }

Console.log(obj.name) = dot notation

Console.log(obj[“name”]) = array notation

Class # 29 25/nov/2023

**OBJECTS**

stdObj.id=1240 { to add new property}

stdObj.name = “sarim baig” {to reassign}

Delete stdObj.gender {to delete a property}

Var isCheck = “name” in obj {to check weather the property name exit in the given object or not, it returns true and false}

Var obj = {

Fname : “sarim”

getNAme : function () {

console.log(this.fname)

}

}

Function made in object is called method

Class # 30 28/nov/2023

**OBJECTS**

Difference bw function and method

Function can return a value or not

Met hod will always return a value

Function can also be made in array

Var arr = [function foo(){

Console.log(“function is made”)}, 0]

Console.log(arr[0]())

An object has 3 entities its name, its values, and its action

**Objects Creation**

Object literal: var obj = {}

**Constructor**

Constructor name should be start with capital letter for best practice

Var std1 = new Std(“sarim”, 22, 1208)

Console.log(std1)

Prototype = predefined methods

Class # 31 30/nov/2023

**Global Objects**

Window is a **global object:** global objects can be called directly e.g a method in a normal function wants to be called so we have to call the object then its method but in global object its method name can directly be written

Window.location.href : to get url of current page

Page switching between pages is done through js when doing validations

**To set href for page switching**

Window.location.href = ./aboutus.html : property

Window.location.assign (“./abount.html”): method

Window.location.replace (“./about.html”): when switching page through replace then it don’t allow to come back it just send forward. USED IN LOGIN PAGE

Window.location.host : brings hosting name

Window.location.pathname : tells on which page user is standing/ path of file

History.forward()

History.back()

History.go(any number)

Window.open( page url) : opens new tab

Window.open(page url, page name, height,width): opens new window

Class # 32 5/Dec/2023

**Array of Objects**

Array of objects: multiple objects in an array

**Storage types of browser**

1. Local storage

It always save values in string

localStorage.setItem(“key”,”value”)

localStorage.getItem(“key”)

localStorage.removeItem(“key”)

localStorage.clear()

2. Session Storage

Class # 33 7/Dec/2023

**Array of Objects**

JSON = javascript object notation

When key of object is in string then its called json format

To convert object into string{

var stringobj = JSON.stringify(obj)

here converting object into json then in string}

To save object in local storage the first we have to convert it in json then save it in storage, now to get it we need to convert it in json.parse

Find(function(value,index){

If value.email === userobj.email {return true}} method to find any value from an array

For(var index of arr){} : for of loop runs on arrray