**Java Script – Interview Topics Task**

1. **var, let, constant**

* Variables are used to declare a data with any types and assign memory allocation to each variable.
* Variables are followed by keyword (var, let, const), variable name and the value.

**Syntax:**

**Variablekeyword variablename = value**

**var name = “abc”**

**var num = 23**

**var isNull = true**

**var username; (undefined)**

**var obj = {userID: 1, username: abc, isAdmin: false, loc: }**

* **var –** It is like global keyword where you can re-declare an variable again and reassign the values of any variables with new value.

**Eg –** var name = "abc" O/P = abc

name = "xyz" //-> reassigning of variable O/P = xyz

var name = "vvv" // -> redeclaring the variable with same name O/P = vvv

* **let –** The let type keyword variables can be reassigned but can’t be re-declared. It will throw an error like “variable already declared” if you re-declare it.

**Eg –** let num = 30 O/P = 30

num = 35 //-> reassigning of let variable O/P = 35

let num = 45 // -> re-declaring of let variable O/P = "Identifier 'num' has already been declared"

* **const –** The const type keyword variables is like a static where it cannot be re-declared and reassigned.

**Eg -** const myName = "ABC" // O/P= ABC

myName = "XYZ" // reassigning of const variable O/P = Assignment to constant variable.

const myName = "ANM" // -> redelaring of const variable O/P = "Identifier 'myName' has already been declared"

1. **String methods and Array menthods**

**String Methods:**

* String methods in JS is used to perform various operations with the string values declared.
* **length** – used to find the length of string

**Eg - v**ar name1 = "Naren"

console.log(name1.length) //O/P = 5

* **split(“”) -** Splits the every letter of variable into seperate index of array

**Eg** - var splitExample = "The Jungle Book"

console.log(splitExample.split("")) // O/P [

// 'T', 'h', 'e', ' ',

// 'J', 'u', 'n', 'g',

// 'l', 'e', ' ', 'B',

// 'o', 'o', 'k'

// ]

* **split(“ “) - Sp**lits the every words of variable into seperate index of array

**Eg -**  var splitExample = "The Jungle Book"

console.log(splitExample.split(" ")) // O/P [ 'The', 'Jungle', 'Book' ]

* **slice()** – used to slice the part of content from variable

**Eg** - var splitExample = "The Jungle Book"

**cons**ole.log(splitExample.slice(4,11)) //O/P = Jungle

* **includes()** - return true or false(will check also the Caps)

**Eg** - var name1 = "Naren"

console.log(name1.includes('A')) // O/P = true

**Array Methods:**

* Array methods in JS is used to perform various operations with the array values declared.
* **Map() –** returns true if the given value is present in array and returns false for remaining index values.

**Eg -** var cars = ["BMW", "Audi", "BENZ", "TESLA"]

console.log(cars.map((d) => d === 'TESLA')) O/P = [ false, false, false, True ]

* **Slice()** - used to slice the part of content from array

**Eg -** var cars = ["BMW", "Audi", "BENZ", "TESLA"]

var slicedArray = cars.slice(2,4)

console.log(slicedArray)

O/P = [ 'BENZ', 'TESLA' ]

* **reverse()** - to reverse the values of array

**Eg** - var cars = ["BMW", "Audi", "BENZ", "TESLA"]

console.log(cars.reverse())

O/P = [ 'TESLA', 'BENZ', 'Audi', 'BMW' ]

* **contact() -** To join the existing array with extra given values

**Eg -** var cars = ["BMW", "Audi", "BENZ", "TESLA"]

var newCars = cars.concat(["KIA", "VW"]);

console.log(newCars)

O/P = [ 'BMW', 'Audi', 'BENZ', 'TESLA', 'KIA', 'VW' ]

* **toString()** - Convert an array to a string

**Eg** - var cars = ["BMW", "Audi", "BENZ", "TESLA"]

console.log(cars.toString())

O/P = BMW,Audi,BENZ,TESLA

1. **Callback function**

* Callback function is declared inside one function and it will be executed according to the sequence order it called.

Eg – function firstName(){

secondName()

thirdName()

return "Hello"

}

function secondName(){

console.log("Naveen")

}

function thirdName(){

console.log("Hi..")

}

console.log(firstName())

O/P = Naveen

Hi..

Hello

1. **Async and await**

* **Await –** The execution will be completed one after another. If the first execution contains timeout then it will wait until and start executing the next process.
* **Async –** The execution will be completed immediately if the previous execution is waiting for timeout.

**Eg -** async function f() {

let promise = new Promise((resolve, reject) => {

setTimeout(() => resolve("done!"), 5000)

});

let result = await promise; // wait until the promise resolves (\*)

console.log(result); // "done!"

}

f();

O/P = done

1. **Promise**

* Promises will return the value if it success by using resolve() and will return the error values using reject().
* **Syntax –** var a = new Promise(function(resolve, reject)){

resolve()

reject()

}

a.then{

function.value(){

}

function.error(){

}

}

* Eg - let a = new Promise((resolve, reject) => {

resolve("Hai")

})

let b = new Promise((resolve, reject) => {

resolve("Bye")

})

Promise.all([a,b]).then((val) => {

console.log(val)

})

O/P = [ 'Hai', 'Bye' ]

* **all() –** Will display output if all function are resolved
* **allSettled() -** Will display output with status and value

**Eg – Output for above declared promise** [

{ status: 'fulfilled', value: 'Hai' },

{ status: 'fulfilled', value: 'Bye' }

]

* **any() –** will return the first success value either resolve or reject

**Eg - Output for above declared promise ->** Hai

* **race() –** it will not check success or fail and return first executed promise function.

1. **Hoisting**

* Using of variable or function before it declared is hosisting.

**Eg –**console.log(a) // undefined

var a = 50; // 50 will be stored in a

console.log(a) //50

1. **this keyword**

* this keyword refers to different objects.

**Eg -** var person = {

firstName: "Naren",

lastName: "Naveen",

fullName : function() {

return this.firstName + " " + this.lastName;

}

};

console.log(person.fullName())

O/P =Naren Naveen

1. **Ternary Operator**

* **? –** Symbol for ternary operator in JS.
* It is used to declare conditions like if, is else
* **Syntax –** condition ? true : false

**Eg -** function name1(){

return "Hello"

}

function name2(){

return "Welcome"

}

let auth = true;

console.log(auth ? name1() : name2());

O/P = Hello

1. **Class in JS**

* Class is followed by keyword ‘class’.
* All the class names should start with Caps. Eg -> class Cars{}.
* Java script class with have method name constructor that can be used to set properties and can be called internally/externally.
* **Syntax -** class Cars {

Constructor(){

}

}

* Eg - class Cars {

constructor(){

this.name = "TESLA"

this.model = "2022"

this.price = 100

}

carData(){

return "My Car is "+ this.name + " model "+ this.model

}

}

1. **Async and Sync**

* Synchronous will be executing in sequence order(one-by-one). If anytime out is declared it will not skip that step. It will wait for that timeout and it will continue to execute the steps after that.

Eg - sync function myfunc(){

var a = await setTimeout(()=> { console.log("Hi") }, 5000)

return a

}

console.log(myfun())

O/P = Hi – will be displayed after 5 secs

* Asynchronous will be executing in parallel. If any timeout is declared it will skip and continue to execute the next step and also executes the timeout steps parallel once the timeout reached.

Eg - async function myfunc(){

var a = await setTimeout(()=> { console.log("Hi") }, 5000)

return a

}

console.log(myfun())

O/P = Hi – will be displayed after 5 secs and other steps after this will be executed simultaneously.