**JavaScript Concept**

getElementById() : this Method used for access DOM element

window.print(); : Method use for print current page. This is predefined method of browser

eg. 1. document.getElementById("demo").style.fontSize = "35px";

**Variables**: In a programming language, **variables** are used to **store** data values.

JavaScript uses the keywords var, let and const to **declare** variables.

Primitive data types: String, Number, Boolean, undefined, Null

Non-primitive Data types: Object , Arrays

var firstName = "Sanket"

function test12(){

    document.write("First Name" + firstName);

var firstName = "Komal";

}

Error: output undefined

var firstName = undefined;

var firstName = "Sanket"

function test12(){

    var firstName = undefined;

    console.log("First Name" + firstName);

    var firstName = "Komal";

}

test12()

Self-invoking function

Recurring function

1. What is difference between **Undefined** and **Null**?

In JavaScript **Undefined** means variable has been declared but value not assigned

Null is an assignment value.

It can be assigned to a variable as a representation of no value.

2. What are different datatypes in JS?

String, Number, Big Int, Boolean, undefined, Null, Symbol

3.Hoisting in Javascript.

Array :

1 Self changing

* **Sort**

max, min, random, alpha-sort, num-sort, ascending, descending

The splice() method adds and/or removes array elements.

The splice() method overwrites the original array.

array.splice(index, howmany, item1, ....., itemX)

* **Push**

The push() method adds new items to the end of an array.

The push() method changes the length of the array.

The push() method returns the new length.

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.push("Kiwi", "Lemon");

* **Pop**

The pop() method removes (pops) the last element of an array.

The pop() method changes the original array.

The pop() method returns the removed element.

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.pop();

* **Shift**

The shift() method removes the first item of an array.

The shift() method changes the original array.

The shift() method returns the shifted element.

const fruits = ["Banana", "Orange", "Apple", "Mango"];

document.getElementById("demo").innerHTML = fruits.shift();

* **Unshift**

The unshift() method adds new elements to the beginning of an array.

The unshift() method overwrites the original array.

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.unshift("Lemon", "Pineapple");

2 Return new array

* **Map**

If you want to iterate each value of the array and want to transform based of the logic,

we can use map function.

If you want to iterate each value of the array and want to transform based of the logic,

we can use map function.

map() creates a new array from calling a function for every array element.

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

map() does not execute the function for empty elements.

map() does not change the original array.

const numbers = [65, 44, 12, 4];

const newArr = numbers.map(myFunction);

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

function myFunction(num) {

return num \* 10;

}

* **Filter**

If you have the array but only you want some specific element then we can use filter.

The filter() method creates a new array filled with elements that pass a test provided by a function.

The filter() method does not execute the function for empty elements.

The filter() method does not change the original array.

const ages = [10,22,41,21,18,9,11];

console.table(ages.filter(checkAdult));

function checkAdult(age){

return age > 18;

}

* **Slice Need discuss**

The slice() method returns selected elements in an array, as a new array.

The slice() method selects from a given start, up to a (not inclusive) given end.

The slice() method does not change the original array.

const fruits = ["0Banana", "1Orange", "2Lemon", "3Apple", "4Mango"];

const citrus = fruits.slice( 2, 4);

console.log(citrus);

const fruits = ["0Banana", "1Orange", "2Lemon", "3Apple", "4Mango"];

const myBest = fruits.slice(-3, -1);

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

3 Type return

* Find index
* IndexOf
* Find
* Reduce

Interview:

**Object:**A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

**JavaScript Map :**

If you want to iterate each value of the array and want to transform based of the logic,

we can use map function.

new Map() Creates a new Map

const fruits = new Map([

["apples", 500],

["bananas", 300],

["oranges", 200]

]);

document.getElementById("demo").innerHTML = fruits.get("apples");

set() Sets the value for a key in a Map

The set() method can also be used to change existing Map values:

fruits.set("apples", 200);

get() Gets the value for a key in a Map:

The get() method gets the value of a key in a Map:

fruits.get("apples"); // Returns 500

Size() : he size property returns the number of elements in a Map

document.getElementById("demo").innerHTML = fruits.size; // Returns array size

delete() Removes a Map element specified by the key

The delete() method removes a Map element

// Delete an Element , Note : you can't delete multiple elements for that splice

fruits.delete("apples");

document.getElementById("demo").innerHTML = fruits.size;

has() Returns true if a key exists in a Map

The has() method returns true if a key exists in a Map:

fruits.has("apples"); // Returns “True”

forEach() Calls a function for each key/value pair in a Map

let text = "";

fruits.forEach (function(value, key) {

text += key + ' = ' + value + "<br>"

})

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

// Returns apples = 500

bananas = 300

oranges = 200

entries() Returns an iterator with the [key, value] pairs in a Map

The entries() method returns an iterator object with the [key, values] in a Map:

let text = "";

for (const x of fruits.entries()) {

text += x + "<br>";

}

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