**JAVASCRIPT CONCEPT**

**JavaScript**: JavaScript is scripting programming language of HTML and the web. It is object based,

Light-weight (consume minimal memory for an application) and cross platform (a product or system that can work across multiple types of platforms or operating environments)

* JavaScript is case sensitive language.
* JavaScript accepts both double and single quotes

**Where javaScript used**: javaScript used to create interactive website, mainly used for

* Client-side validations
* Dynamic dropdown menu
* Displaying date and time
* Displaying popup and dialog boxes
* Displaying Clock

**JavaScript Data Type**

|  |  |
| --- | --- |
| **Primitive Data Type** | **Non-Primitive Type** |
| String  Number  Boolean  Undefined  Null | Object  Array  RegExp |

**BOM**: BOM Stands for browser object model, is used to interact with the browser.

The default object of browser is windows means you can call the functions of windows by

specifying windows or directly:

The window object supported all browsers. It represents browsers window.

ex. windows.alert(“Hello world”)

windows.open() // Open new window

windows.close(); // Close current window

windows.moveTo(); // Move the current window

windows.resizeTo(); // resize current window

**DOM**: DOM Stands for Document Object Model

The DOM represents whole HTML Document. When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page. Ex below

1. getElementById (): this Method used for access DOM element

2.

function printValue(){

     var name=document.form1.name.value;

     alert("Welcome" + name);

}

**Output**: JavaScript can display data in different ways

* Using innerHTML
* Using document.write ()
* Using windows.alert ()
* Using console.log ()
* Using console.table ()

|  |  |
| --- | --- |
| View state | Session state |
| Maintained at page level only. | Maintained at session level. |
| View state can only be visible from a single page and not multiple pages. | Session state value availability is across all pages available in a user session. |
| It will retain values in the event of a postback operation occurring. | In session state, user data remains in the server. Data is available to user until the browser is closed or there is session expiration. |
| Information is stored on the client’s end only. | Information is stored on the server. |
| used to allow the persistence of page-instance-specific data. | used for the persistence of user-specific data on the server’s end. |
| ViewState values are lost/cleared when new page is loaded. | SessionState can be cleared by programmer or user or in case of timeouts. |

**NEED DISCUSS**

Usage:

• SessionState: It can be used to store information that you wish to access on different web pages.

• ViewState It can be used to store information that you wish to access from same web page.

**JavaScript Events:** HTML events are “things” that happen to HTML elements. When JavaScript is used

In HTML page, javaScript can “react” on these events.

Common HTML events

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

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

Functions:

var curriedMultiply = currying(multiply);

multiply(4, 3); // Returns 12

curriedMultiply(4)(3); // Also returns 12

**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 is an object-based language. Everything is an object in JavaScript.
* JavaScript is template based not class based. Here, we don't create class to get the object. But we direct create objects.

Creating Objects in JavaScript there are 3 ways to create objects.

1. By object literal

2. By creating instance of Object directly (using new keyword)

3. By using an object constructor (using new keyword)

1.By object literal: The name: values pairs in JavaScript objects are called properties:

const person= {

    firstName:"John",

    lastName:"Doe",

    age:50,

    eyeColor:"blue",

    findAdult : function() {

        if( this.age >= 18){

            console.log ( this.firstName + " Can vote" )

        }

    }

};

2. By creating instance of Object directly (using new keyword)

var emp = new Object();

emp.id = 1;

emp.name = "Komal";

emp.sal = 100000;

3. By using an object constructor (using new keyword)

function emp (id,name,sal){

        this.id=id;

        this.name=name;

        this.sal=sal;

    }

e = new emp (10,"Yogesh",20);

Need Discuss : When we can use dot and bracket notation? When you want to access object property with a variable, use Bracket *objectName["propertyName"]*

**Object Methods**

* Objects can also have methods.
* Methods are actions that can be performed on objects.
* Methods are stored in properties as function definitions.

Ex.

const person= {

firstName:"John",

lastName:"Doe",

age:50,

22:"222",

eyeColor:"blue",

findAdult : function() {

if( this.age >= 18){

console.log ( this.firstName + " Can vote" )

}

}

};

1. **Object.assign**() : The Object.assign() method copies all enumerable own properties from one or more source objects to a target object.

It returns the modified target object.

let komalSchool = {

schoolName: "RMG",highestQalification : "10Th" , Year:1922

}

let komalClg = {

clgName: "Kalmadi",highestQalification: "MCA", cPin:200

}

let companyDate = Object.assign(komalSchool,komalClg);

console.log(komalSchool.pin)

console.table(companyDate);

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**: The Array object is used to store multiple values in a single variable:

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**

The findIndex() method executes a function for each array element.

The findIndex() method returns the index (position) of the first element that passes a test.

The findIndex() method returns -1 if no match is found.

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

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

Ex. const ages = [3, 10, 18, 20];

console.log(ages.findIndex(checkAge));

function checkAge(age){

return age >= 18;

}

* **IndexOf**

The indexOf() method returns the first index (position) of a specified value.

The indexOf() method returns -1 if the value is not found.

The indexOf() method starts at a specified index and searches from left to right.

By default the search starts at the first element and ends at the last.

Negative start values counts from the last element (but still searches from left to right).

Ex.

const myArr = ["Ranjit", "Sankt", "komal","yogesh","Nivant","komal", "Shrikant" , "Pooja", "komal","Ranjit"];

let index = myArr.indexOf("Ranjit",2)

console.log(index);

* **Find**

The find() method returns the value of the first element that passes a test.

The find() method executes a function for each array element.

The find() method returns undefined if no elements are found.

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

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

const ages = [3, 10, 18, 20];

console.log(ages.find(checkAge));

function checkAge(age){

return age >= 18;

}

* **Reduce**

The reduce() method executes a reducer function for array element.

The reduce() method returns a single value: the function's accumulated result.

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

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

const numbers = [50,300,175, 50, 25];

console.log(numbers.reduce(myFunc))

function myFunc(total, num) {

console.log(total,num)

return total + num;

}

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;

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**JavaScript Questions/Answer**

1. What is javaScript?

JavaScript is scripting programming language of HTML and the web. It is object based,

Light-weight (consume minimal memory for an application) and cross platform (a product or system that can work across multiple types of platforms or operating environments)

* JavaScript is case sensitive language.
* JavaScript accepts both double and single quotes

2.What is variables.

Ans->Variables are containers for storing data (storing data values).

Javascript variables is simply name of storage location.

Two types of variable-global and local

3. JavaScript Data Types.

Ans-> Two type- primitive and non primitive

Primitive-number,string,Boolean,undefine,null

Non primitive-object

4. Functions in javascript.

Ans->A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

Types-predifine(parseInt, parseFloat, Eval, call(),bind(),apply()) and user define

5. What is Objects and explain in details.

Ans->an object is **a standalone entity, with properties and type**.

Object is non primitive datatype in js.

Object class represent one of the javascript datatype ,it is used to store the various keyed collections and more complex entity.

5.1: What is “**this**”?

“this” keyword always refers to the owner object to the function. Which object depends on how this is being invoked (used or called).The this keyword refers to different objects depending on how it is used

6. How to access object.

Ans->const car = {type:"Fiat", model:"500", color:"white"};

console.log(car);

access object by 3 ways:

1.object.property

2.object[“property”]

3.const{property}=object

Q. How to create object in multiple ways?

Ans-> Four ways (methods):

1.Object Literal:

e.g. - //creating js objects with object literal

let car = {

    name : 'GT',

    maker : 'BMW',

    engine : '1998cc'

};

//property accessor

console.log(car.name); //dot notation

console.log(car['maker']); //bracket notation

2. Creating By Constructor or Operator:

e.g. - //simple function

**function** vehicle(name,maker,engine){

**this**.name = name;

**this**.maker = maker;

**this**.engine = engine;

}

//new keyword to create an object

let car  = **new** vehicle('GT','BMW','1998cc');

//property accessors

console.log(car.name);

console.log(car.maker);

console.log(car['engine']);

3.Object.create() method:

e.g. - const coder = {

    isStudying : **false**,

    printIntroduction : **function**(){

        console.log(`My name is ${**this**.name}. Am I studying?: ${**this**.isStudying}`);

    }

};

const me = Object.create(coder);

me.name = 'Mukul';

me.isStudying = **true**;

me.printIntroduction();

4.classes:

//using es6 classes

class Vehicle {

  constructor(name, maker, engine) {

**this**.name = name;

**this**.maker =  maker;

**this**.engine = engine;

  }

}

let car1 = **new** Vehicle('GT', 'BMW', '1998cc');

console.log(car1.name);  //GT

7. How to add property in object.

Ans-> access object by 3 ways:

1.object.property

2.object[“property”]

3.const{property}=object

8. How to display object property.

Ans->

1.Displaying the Object Properties by name

e.g. - const person = {

name: "John",

age: 30,

city: "New York"

};

document.getElementById("demo").innerHTML =

person.name + "," + person.age + "," + person.city;

2.Displaying the Object Properties in a Loop

const person = {

name: "John",

age: 30,

city: "New York"

};

let txt = "";

for (let x in person) {

txt += person[x] + " ";

};

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

3.Displaying the Object using Object.values()

Any JavaScript object can be converted to an array using Object.values().

const person = {

name: "John",

age: 30,

city: "New York"

};

const myArray = Object.values(person);

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

4.Displaying the Object using JSON.stringify()