JavaScript

Day-1

\*\*\* Basics of JavaScript \*\*\*

Why I call it the basics of JavaScript means in every programming language the concepts are the same but the way we are using is different.

What is JavaScript?

JavaScript is a programming language.it was called scripting language.

Ex: java, c#, scale.js

What is the purpose of JavaScript?

It is used to create dynamic UI. (UI=Static [HTML, CSS] + dynamic [Js])

We can create entire web-applications. (frontend & backend & DB)

We can do client-side and server-side validation ()

We can create a mobile application.

we can create standalone application. (which runs on desktop--paint-file,)

We can use ML-algorithms also.

What is the difference between p-lang and scripting-lang?

How to work with JS?

We required a few tools.

1)vs-code (IDE: integrated development env)

2)install Nodejs

note:

vs-code: implement JS code.

NodeJS: run JS.

Day-2

What are variables in JavaScript?

Variables are the name given to the memory location where we store the data.

Syntax: [it has set of rules and regulation] or [grammar of any P-lang]

How to create a variable in JS?

There are 4 ways to create a variable in Js.

1. <variable>;

Ex: id; city = “hyd”; var productName = “Iphone” let price = 1000

const pin = 3231

1. <scope-statement> <var-name> = <data>

Ex: var let const

Var city =” Hyd” let city = “Pune” const city = “Delhi”

There are some rules which need to be followed to give names for the variables.

1)We cannot start variable name with number.

2)We cannot use variable names with special characters, expect$, underscore\_

3)We can start variable name with some letter than we can use some number.

4)We can create JS variable without scope statement as well.

Day-3

What are datatypes in JavaScript?

Datatypes is nothing but the type of information or data.

Ex: number, string, Boolean

Number: any data we can represent in numeric or decimal form. We can do all the arithmetic operations.

String: Any data we can represent the “”, ‘’, comes under string type.

Boolean: any data which we can go to represent in true or false.

How many datatypes in js?

There are total of 8 datatypes in JS.

1)number 2) string 3) Boolean 4) null 5) undefined 6) Bigint 7) symbol 8) object

In JS there are two special values.

1)undefined – undefined only

2)null –null only

Bigint:

Any data which is re-presented in numeric form and whose value is greater than 2 power 53 -1 such type of data comes under bigint.

Symbol:

Any data which is created using symbol constructor, comes under symbol datatype.

Ex: var data = symbol(“clksdnclkdsn”)

Object:

Any data which is represented in object or array format, then such data comes under object type.

Ex: var data = {name:” Vinay”, id:101, phn:9876541}

Var data = [100,222,333,44,555]

Typeof:

It is a special operator used to find the data type of a variable.

Example:

var info = Symbol("sagar");

var id = 100001;

var username = "sagar";

//Special Operator called typeof

//Typeof is a Special operator is used to find the data type of a variable

console.log(typeof info);

console.log(typeof id);

console.log(typeof username);

Day-4:

What are functions in JavaScript?

Functions are the block which consists of some set of statements or instructions to perform some task or some actions.

Example: In Js file write the code to perform addition operation.

100+100

How to re-use the code multiple times?

Using function concept we can re-use the code.

How to create a function?

Function <fnName>(){

//set of statements

}

Example:

Function firstFun () {

//statements

}

Note: Function is the keyword

{function block}

firstFun is the function name.

How to execute the logic of the fuction?

For this we have to call the function

We can call function any number of times.

Example:

function firstFun() {

    var num1 = 100;

    var num2 = 200;

    var result = num1 + num2;

    console.log(result);

  }

  firstFun(); //100 + 200 --🡪function call

  firstFun(); // 300 + 400

  firstFun(); // 500+500

How to give input data to the functions?

Using Parameters and arguments we can give input data to the functions.

Parameters:

1)parameters are variables without any scope statements

2)parameters are always defined in the function definition.

3)parameters are defined inside parenthesis of function definition.

Example:

Function f1(name, city) {

}

Note: name and city are the parameters.

4)Parameter are used to hold (or)store the input data of the functions.

5)We can define any number of parameters inside the function definition.

6)Each parameters should be differentiated (or) separated with comma(,).

Example:

Function fnAdd(x,y) {

}

Arguments:

1)Arguments are the actual data (or) information that we can pass to the function.

2)Arguments are always passed while calling the function.

3)We can pass any number of arguments to the function.

4)we can pass any type of data as a argument.

5)arguments are diffrenetiated (or) separated using comma(,).

Example:

fnAdd(100,200)

fnAdd(“vinay”,”varma”)

fnAdd(100,”vinay”)

Explain the difference between parameters and arguments?

Explain the difference between variable and parameter?

Day-5:

Returned function:

How to return some data from the function?

1)using return statement we can return the data from the function.

2)return statement should be the last statement inside the function.

3)return statement will terminate the execution function.

Example:

Function fnGet() {

Return “Hello js”;

}

Var info = fnGet();

Console.log(info);

Anonymous function:

Any function which does not have any name is called anonymous function.

(Or)

Functions without name are called anonymous function.

Example:

Function(){

}

Example:

Var getCity = function(){

}getCity();

Arrow function:

A function without function keyword and function name is called arrow function.

Example:

Var postinfo = (x,y)=>{

}

Callback function:

Any function which is passed as a arguments is called callback function.

Generally, we use anonymous (or) arrow function called callback function.

Example:

function f1(x) {

    x();

  }

  function f2() {

    console.log("f2 is called");

  }

  f1(f2);

  f1(function () {

    console.log("Anonymous function is called");

  });

  f1(() => {

    console.log("Arrow function is called");

  });

Higher order function:

Any function which takes another function as a argument is called higher order function.

Note:

Here f1 is the higher order function because we passed at least one argument as a function.

In the below example we passed two arguments as a function i.e; anonymous function and arrow function.

Example:

function f1(x) {

    x();

  }

  function f2() {

    console.log("f2 is called");

  }

  f1(f2);

  f1(function () {

    console.log("Anonymous function is called");

  });

  f1(() => {

    console.log("Arrow function is called");

  });

Day-6:

What are objects in JavaScript?

1)Object is a **Data Structure** and collection of **properties** where each property will be **key value pair**.

2)Object is also **technique** or some **mechanism** to store and manage the data.

How Object stores the data?

Object stores the data in the form of property.

City: “Delhi”

Name: “Vinay”

Note: Where city is key and Delhi is the value.

How to create an object?

a) using literal notation b) using constructor

Using literal notation:

Example: <scope statement> <objname> = {prop}

Var obj = {

Id:123,

Name:” Vinay”};

Note:

In the object, no two properties should have the same key and values can be same.

Keys should be unique in the object.

We can add any number of key names with different values.but it should be taken only one value that also latest value.

Data structure:

It is a technique or mechanism to store and manage data.

Property:

Property will be in key value pair.

To any property we can assign any type of data as a value.

We can perform All the CRUD operation on the object?

**1)Read:** we can access the value of specific property.

2)**Insert**: We can add new property dynamically.

3)**Update**: We can update the value of any existing property.

4)**Delete**: We can remove any property dynamically .

How to access any prop value from the object:

Syntax:

Var value = <object name>. <key name>

Var value = <object name> [“key name”]

Day-7

How to add a new property dynamically inside the object ?

Syntax:

<object-name>. <new-prop key-name> = <new-value>

Example:

var obj = {

    name:"Vinay",

    id:101,

};

console.log(obj);

//new property

obj.gender = "male";

obj.email = [vinay@gmail.com](mailto:vinay@gmail.com);

console.log(obj);

In object, properties order does not matter.

New property will always be added at the end.

How to update the property value in the object?

Syntax:

<obj-name>.<key-name> = <updated value>

Example:

var user ={

    name:"Vinay sai",

    city:"Delhi",

    id:434,

};

console.log(user);

//update

user.name = "Batta Vinay sai";

user.id = "101";

console.log(user);

How to remove the specific prop from the object?

delete is a special operator! Which we can delete the values.

Syntax:

Delete <objname>.<keyname>

Example:

var emp = {

    id: 101,

    productName: "XYZ",

    price: 999,

};

console.log(emp);

//delete

delete emp.id;

delete emp.price;

console.log(emp);

How to access the keys of an object?

Var keys = Object.keys(<object-name>)

Example:

var obj = {

    id:101,

    city:"hyd",

    name:"XYZ",

    Price:999,

}

var arr =Object.keys(obj);

console.log(arr);

//if we access particular value

console.log(arr[0]);

Note:

Console: is a predefined object

Log: is a function

In the object we can have function

Example:

var obj = {

    name:"Vinay",

    id:101,

    gender:"male",

    fn:function(Newname){

        console.log("fn is called",Newname);

    },

};

obj.fn(sai);

Day:8

What are Array and Array method?

1)Array is a data structure and collection of elements.

2)Array is used to store multiple values of the same type and different type also.

3)whatever data we stored in the array all those data are called elements.

How to create an array?

1)Literal method

Syntax:

let <var-name> = []

Example:

var data = [] //empty array

var info = [10,20,30,40,50]; //same type of array

var names = ["Vinay","sai","rahul","dev"]; //diff type of array

var users = [101,"vinay",true,{city:"hyd"},[true,false]];

console.log(users);

console.log(names);

4)Every element in the array will have a unique number which is called index.

5)using index number we can identified position of the element in the array.

Ex: var info = [10,20,30,40,50]; //index number starts from zero.

How to access the array element?

Using the index number we can access the array element.

Example:

var arr = [10,11,22,33,44,55,66];

 var element = arr[4];

 console.log(element);

length:

Using this length property, we can find the number of elements in the array.

Example:

var arr = [10,11,22,33,44,55,66];

 var element = arr[4]; // access the array with index number

 console.log(element);

 var numberOfelements = arr.length; //find the length of the array

 console.log(numberOfelements);

To perform operations on array, we have some pre-defined functions and those are called array methods.

There are nearly 30+ array methods(functions)

Array methods:

1)PUSH ():

Using this function (or) method we can add new elements in the last position of the array.

We add multiple values in the array.

Syntax:

<array-name>.push(newElement)

Example:

var arr = [10,11,22,33,44,55,66];

console.log(arr);

console.log("------------------------------------");

arr.push("vinay");

arr.push(true);

arr.push(80);

arr.push(50);

console.log(arr);

2)UNSHIFT ():

Using this method, we can add element in the first position.

We can add multiple elements in the array.

Syntax:

<array-name>.unshift(newElement)

Example:

var arr = [10,11,22,33,44,55,66];

console.log(arr);

console.log("------------------------------------");

arr.unshift("Vinnu");

arr.unshift(true);

console.log(arr);

3)POP ():

It will remove the last element from the array.

We can remove only one value at a time.

Syntax:

<array-name>.pop()

Example:

var arr = [10,20,30,40,50,60];

console.log(arr);

console.log("-------------------------------");

var removedElement = arr.pop();

console.log(arr);

console.log(removedElement);

4)SHIFT ():

It will remove the element from the first position

It will return the removed element.

Syntax:

Var removedElement = <arrname>.shift()

Example:

var arr = [10,20,30,40,50,60];

console.log(arr);

console.log("-------------------------------");

var removedElement = arr.shift();

console.log(arr);

console.log(removedElement);

5)Splice ():

Using splice, we can add (or) remove (or) update any element in the array.

Removing any specific element in the array:

Syntax:

<array-name>.splice(indexNumberElementToBeRemoved,Count)

Example:

var arr = [10,20,30,40,50,60,70];

console.log(arr);

console.log("------------------------");

arr.splice(3,1);

console.log(arr);

arr.splice(3,2);

console.log(arr);

Adding new element in any position of the array?

Syntax:

<array-name>.splice()

Example:

var arr = [10,20,30,40,50,60];

arr.splice(6,0,"Vinay");

console.log(arr);

Update new element in any position of the array?

Example:

var arr = [10,20,30,40,50];

arr.splice(3,1,true);

console.log(arr);

Day-9:

6)IndexOf():

It is used to find the index number of element.

Syntax:

Arr.indexOf(element)

Example:

var arr =[10,20,30,40,50];

var indexnumber = arr.indexOf(100);

console.log(indexnumber);

7)Include:

It is used to check element Is present in array or not.

Example:

var arr = [10,20,30,40,50,60];

var result = arr.includes(50);

console.log(result);

access:

How to access the each and every element in the array.

Example:

var arr = [10,20,30,40,50,60];

console.log(arr[0]);

console.log(arr[1]);

console.log(arr[3]);

console.log(arr[4]);

console.log(arr[5]);

console.log(arr[6]);

\*\*\*

8)forEach():

Using forEach we can access each and every element in the array.

Syntax:

arr.forEach(function(element,index){

})

Example:

var arr = [10,20,30,40,50,"vinay"];

arr.forEach(function(element,index){

    console.log(element,index);

});

\*\*\*

9)Map ():

It is used to perform the same operation on each and every element in the array and store the result of the operation in the new array.

Note:

Map having call back function and having return statement.

Syntax:

Var newArray = arr.map(function(element, index){

Return(value)

})

Example:

var arr = [10,20,30,40,50,60]; // \* 100 = []

 var newArray = arr.map(function(element,index){

    var product = element \* 100;

    return product;

});

console.log(arr);

console.log(newArray);

\*\*\*

10)Filter():

It is used to filter the array based on the condition.

Syntax:

Arr.filter(function(element,index){

return <Boolean value>

})

Example:

var arr = [10,20,30,40,50,60];

var newFilteredArray = arr.filter(function(element,index){

     return element < 20;

});

console.log(newFilteredArray);

Day-10:

SELECTION STATEMENTS (or) control statements

In JS we have 4-control statements.

1)if 2) else 3) else if 4) switch

Using these statements will help us to execute a certain set of statements based on the condition.

It will help to control the flow of the program.

We can add any number of “if blocks” in the program.

1)If:

Syntax:

If(condition) {

Statement (or) instructions};

Note: start of the curly brace to end of the curly brace we can call it as “if block”.

Condition will return true or false.

Example:

var x = 0;

if(x > 100){ // 0 greater than 100  = false

    console.log("x is greater than 100");

}

console.log("if block is terminated");

Example:

Write a number to find if the given number is positive or negative?

function checkPositiveNegative(value){

   if(value >= 0){

      console.log(value, "is a positive number");

   }

   if(value < 0){

    console.log(value,"is a negative number");

   }

}

checkPositiveNegative(100);

Note:

In JS we have something called relational operator (> < <= >= == === !=).

The relational operator always returns Boolean value.

2)else:

Else statement is always used with the combination of if statement.

Syntax:

If(condition) {

}

else {

}

Note:

If the condition is true whatever we return in the if block will return.

If the condition becomes false, whatever we return in the else will return.

Example:

var x = 10;

if(x > 15) {

    console.log("x is greater then 15");

}

else{

    console.log("x is smaller then 15");

}

Note:

Find the difference between == and === operator?

Example:

var productData = {

    pNmae: "Iphone 15",

    price: 150000,

    inStock:false,

};

//WAF to print the given product is available or it is out of stock

function productAvailbility(){

    if(productData.inStock){

        console.log("Available")

    }

    else{

        console.log("Out of stock")

    }

    }

Example:

var productData = {

    pNmae: "Iphone 15",

    price: 150000,

    inStock:”outofstock”,

};

//WAF to print the given product is available or it is out of stock

function productAvailbility(){

    if(productData.inStock == “instock”){

        console.log("Available")

    }

    else{

        console.log("Out of stock")

    }

    }

Else-if:

It is one of the selection statements we can add multiple conditions in the program.

Else-if should be used with if statement only.

Syntax: if(){

}

Else if(){

}else if(){

}else{

}

Note: We can add any number of else if in between if statement.

Example:

var x = 10;

var y = 20;

var z = 30;

if(x > y){ // 10 > 20 false

    console.log("x is greater than y");

}

else if(x > z){ 10 > 30 false

     console.log("x is greater than z");

}

else{

   console.log("x is smaller than y and z");

}

Task: WAF to find the employee designation ?

function finddesignation(experience){

   if(experience >= 0 && experience <=2){

    console.log("Jr engineer");

   }

   else if(experience > 2 && experience <= 5){

    console.log("SR eng");

   }

   else if(experience > 5 && experience <= 7){

    console.log("Team lead");

   }

   else if(experience > 7 && experience <= 10){

    console.log("Project manager");

   }

   else{

    console.log("SW architect");

   }

}

finddesignation(1.8);

Switch:

Switch is one of the selection statements, using switch we can create multiple blocks.

Based on some values we can execute certain blocks.

Switch statement takes expression or value (---)

Switch block contain series of cases where every case Is one block.

Syntax:

Switch (expression or value){

Case “<case-value1>”:statement:

--------------------------------------------

Break;

Case “<case-value2>”:statement:

--------------------------------------------

Break;

Case “<case-value3>”:statement:

--------------------------------------------

Break;

Default:

}

Example:

var color = "red";

switch(color){

    case "red":

        console.log("red color is selected");

        break;

        case "blue":

        console.log("red color is selected");

        break;

        case "black":

        console.log("red color is selected");

        break;

        case "yellow":

        console.log("red color is selected");

        break;

        default:

            console.log("color out of the range");

}

Task: WAF to perform addition subtraction multiple division like calculator?

Day-11:

What are operators in JavaScript?

There are 6 categories of operators.

1)Arithmetic operator

Example: +, - , \* , / , %(modulus, will return remainder), ++(post and pre) , --(post and pre) ,

2)comparison (or) relational operator

Example: >,<,==,===,<=,>=,!=

3)logical operator

Example: &&(AND), ||(OR),!(NOT)

4)Bitwise operator

Example: &(Bitwise And), |(Bitwise or),~(Bitwise not),<<(bitwise leftshift),>>(bitwise rightshift),>>>.

5)Assignment operator

Example: =, +=, -=,\*=,/=,%=

6)Special operator

Example: type of, delete,new,of,in,dot,comma,terinary-opeartor,void,yield,instance of.

What are comments in JS?

Comments are those lines (or) statements which are not participated in the JavaScript Execution.

Comments are used for the documentation.

// -🡪 comment in js

In JS we have single line and multiline comments.

If we want to comment multiline just press cntl+?

If we want to uncomment multiline just press cntl+?

1)Arithmetic operator:

Example: Modulus

var x = 100;

var y = 50;

var result = x % y; //will return the remainder

console.log(result);

Note:

X + Y Here + is the operator and X and Y are operands.

Example: Division

var x = 100;

var y = 3;

var result = x / y; //will return the coffiecient

console.log(result);

Example: Pre-Inc

Pre-increment will increase the value by one.

++(operand)

++x

var x = 1;

console.log(x);

++x;

console.log(x);

Example: Post-inc

Post-increment will also increase value by one.

X++

var x =3;

console.log(x);

x++;

console.log(x);

What is the difference between post and pre-increment?

Pre-increment: first it is performing the operation and printing the value.

Post-increment: it is printing the value then it performs increment value.

var num1 = 100;

console.log(++num1); //pre-inc

var num2 = 100;

console.log(num2++); //post-inc

var x = 100;

var y = 200;

var result1 = x + ++y; // 100 + 201 = 301 pre-inc

var result2 = x + y++; // 100 + 200 =300 post-inc

console.log(result1,result2);

Example: pre-post inc and dec

var  x = 100;

var y = 50;

var result1 = --x + --y; //pre-dec 99 + 49 = 148

console.log(result1); // x=99, y=49

var result = x-- + y--; //post-dec  99 + 49 = 148

console.log(result); // x = 98,y = 48

console.log(x,y); //x =98, y =48

2)Relational (or) comparison operator:

Using these operators, we can compare values of two operands.

Example: [>]

var num1 = 100;

var num2 = 200;

console.log(num1 > num2); // 100 > 200 false

Example: [<]

var num1 = 100;

var num2 = 200;

console.log(num1 < num2); // 100 < 200 true

Example: [==, ===]

The difference between == and ===?

== it will check the value and data type it will do some kind of operations.

=== it will check the value and data type

var x = 100; //num

var y = "100"; //string

console.log (x == y); //true

console.log (x === y); //false

var a = 100; //num

var b = 100; //num

console.log (a == b); //true

console.log (a === b); //ture

Example: !=

var a = 10;

var b =20;

console.log (a != b); // 10 != 20  true

3)Logical operator:

Logical operators are used to check two statements or two Boolean values.

It can be used for decision making.

Note: AND

T && T =True

F && T =False

T && F = False

F && F = False

Example:

Var x = 10 var y = 20 var z = 30

x>y && x<z 🡪f && t =false

Note: OR

T || T = true

T || F = true

F || T = true

F || F = false

Example:

var details = [

     {

       brand: "Apple",

       price:100000,

       mode:"iphone15"

     },

     {

        brand: "One plus",

        price:45000,

        mode:"Oneplus nord"

      },

      {

        brand: "samsung",

        price:90000,

        mode:"samsung ultra"

      },

      {

        brand: "Vivo",

        price:30000,

        mode:"Vivo 5"

      },

      {

        brand: "Nokia",

        price:30000,

        mode:"Nokia C2"

      },

      {

        brand: "Redmi",

        price:50000,

        mode:"folded"

      },

    ];

    var Filterdata = details.filter(function(element){

        return element.brand == "Apple" || element.brand == "samsung";

    });

    console.log(Filterdata);

Example:

print all the mobile details whose brand should either apple or Samsung and price should be greater than 80000?

var details = [

    {

      brand: "Apple",

      price:100000,

      mode:"iphone15"

    },

    {

       brand: "One plus",

       price:45000,

       mode:"Oneplus nord"

     },

     {

       brand: "samsung",

       price:90000,

       mode:"samsung ultra"

     },

     {

       brand: "Vivo",

       price:30000,

       mode:"Vivo 5"

     },

     {

       brand: "Nokia",

       price:30000,

       mode:"Nokia C2"

     },

     {

       brand: "Redmi",

       price:50000,

       mode:"folded"

     },

   ];

   var Filterdata = details.filter(function(element){

       return (

       (element.brand == "Apple" || element.brand == "samsung") && element.price > 80000

       );

   });

   console.log(Filterdata);

DAY-12:

LOOPS

Loops are used to execute same statements multiple times until certain conditions become false.

There are total 5-loops:

1)while 2) do-while 3) for 4) for of 5) for in

While loop:

1)It is a loop statement

2)using while we can execute certain statements or instructions multiple times until conditions become false.

Syntax:

While(condition){

//statements

}

Example:

function printValue(){

    var A = 1;

    while(A <=10){

        console.log(A);

         A = A+1;

    }

}

Do-while:

1)It is one of the loop statements.

2)it is also executing certain set of statements multiple times, until conditions become false.

Syntax:

Do {

} while (condition){}

Example:

function printValue() {

    y = 0;

    do {

      console.log(y);

      y = y + 5;

    }

    while(y <= 50);

}

printValue();

for-loop:

1)It is also a loop statement

2)I t is used to execute certain statements multiple times based on the condition.

Syntax:

For (initialize var; condition; inc/dec){

//statements

{

Example:

function printValue(){

     for(var i = 1; i <= 10; i++){

        console.log(i);

     }

}

printValue();

Example:

function printValue(){

    for(var i = 52; i <= 90; i = i+2){

        console.log(i);

      }

    console.log("for loop terminated");

}

printValue();

\*\*\*\* Advance concepts of JavaScript \*\*\*

As of know we are not integrated with HTML we can integrate with html using DOM and DOM manipulation concepts.

DOM and DOM Manipulation concept is very important to learn advance technologies like reactjs.