1. Variables
2. Functions
3. Arrays
4. JSON
5. Promises
6. DOM Manipulation
7. CallBacks
8. Events Handling
9. Event Bubbling and Event Capturing
10. Debouncing
11. Throttling
12. AJAX Calls
13. Closuers

JavaScript

Netscape company released the JS

<script> is used to include js in html

ES23 latest ECMA Script

-JavaScript is scripting lang

-JavaScript used to build the following applications

Angular React NodeJs Dynamic web pages

-We will save the js files with .js extension

-JS execution is synchronous execution(line by line)

-JavaScript is the object based scripting language.

Ex: Window,document,date,console..….

console.log(100+ +"100"); gives 200 inbuilt conversion with only one extra +

console.log(10+"10"+10);  //101010

console.log(10+ +"10"+10);  //30

console.log(10-"2") //8

console.log(10+"2"-"2") //102-2=100

console.log(10>9>8);    //10>9 ->true =1 ; 1>8 false o/p=false

console.log(10\*"10");   //100

console.log(10/"10");   //1

console.log(100/"0");   //infinity

console.log(100+ +"100"-"100"\*"100");   //-9800 BODMAS

== checks value

=== value and datatype

(1 == “1”) //true

(1 === “1”)//false not same date type

console.log(1 == "1"); //true same value

console.log(1 === "1"); //false not same date type

console.log(1=="one")//false not same value

console.log(1==="one")//false not same value and type

console.log(1+true);    //2 true=1

console.log(1-true);    //0 true=1

Variables

Variables are used to store the data

String,number,boolean,object

We can declare the variables by using “var”,”let”,”const”

Let and const keywords are introduced in es6

Variables should contain a-z, A-Z 0-9,$ and \_

Variables should not start with digits

Syntax

Var/let/const varibalename=value;

Numbers

1.decimal 2.float or double 3. Hexadecimal 4.ocatal 5.binay

Hexadecimal numbers will prefix with “0x”

Octal number will prefix with “0o” “0b” for binary

var decimalNum=100;

var doubleNum=100.113;

var hexadecimalNum=0x1321ABC;

var octalNum=0o31231;

var binaryNum=0b01010;

true=1;

false=0;

console.log(1+true);

console.log(true-false); //1-0=1

console.log("1"+true);//1tr

// ue

console.log(+"1"+true); //2

var x=true;

x==true?console.log("Hello"):console.log("Bye");

Strings

Collection of characters called as string

` backtick operator called as template literal in es6 and used to define paragraphs

Var sub=”JavaScript”

var sub\_one="ReactJs";

var sub\_two="NodeJs";

var stack=`${sub\_one}=>${sub\_two}`

console.log(stack);

var tb\_name="employees";

var sal=4000;

var sql=`select \* from ${tb\_name} where esal>${sal}`;

console.log(sql);

//undefined ->value will be there but it is of no use that is memory is occupied like bench people

//null->no value no memory like no job

Var x; //undefined

x1=null;

console.log(x1);

//bigint at the end keep "n" so that we can print big number also

var bigint=123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456123333333333333333333333333333333333333333332387654688765567887655678765567887654678763456n;

console.log(bigint);

console.log(typeof "hello");    //string

console.log(typeof 100);    //number

console.log(typeof true);   //boolean

console.log(typeof undefined);

console.log(typeof null); //object

console.log(typeof []); //object

//for loop

for(var i=0;i<5;i++){

}

console.log(i);//var:5 //let:error i is not defined

//var keyword breaks the scope rule

//let keyword obeys the scope rule

var sub="JavaScript";

var sub="Nikhil Thula";

console.log(sub);   //var:Nikhil Thula  //let gives error it wont allow duplicates

var and let and const

//global

var data=100;

//console.log(data);

//block

{

    //local

    var data=200;   //actually 200 should be accessible inside that block only

}

console.log(data);  //200

here var is not obeying its rule like we think var data =200 will be inside block but when we print data 200 is printing outside which is not wrong 100 should print because it is global variable. So use let which is scope specific. Let data=200; and for 100 also same.

If any block of code effecting global member called as global polluting issue

Global polluting issue raised bcz of var keyword

We can overcome global polluting issue by using keyword

//var msg;    //variable declaration

//var msg"Hello"  //variable intilization

var msg="Hello";

console.log(msg);

console.log(xyz);   //undefined with var but expected error

var xyz=600;        //so use let

//getting undefined before declaration and initialization called as variable hoisting

//we can overcome variable hoisting with the help of let keyword.

|  |  |
| --- | --- |
| var | let |
| ES1 | ES6 |
| Scope rule break | Scope rule follows |
| Duplicate members allows | Duplicate not allow |
| Global polluting issue | We can overcome |
| Variable hoisting raised | We can overcome |

//we cant modify complete array but we can modify inside its values by using const keyword

const arr=[1,2,3,4,5];

arr[0]=100;

arr[2]=300;

arr[5]=600; //extra added

console.log(arr);

arr.pop();  //remove last

console.log(arr);

arr.shift();

console.log(arr);

const obj={

    "one":"React",

    "two":"Angular",

    "three":"java"

};

//error due to const we can modifiy inside only

 /\*    obj={

} \*/

 obj.one="harish";

 obj.three="kiran";

 console.log(obj);

Functions

It is also called as business logic

Functions are used to reuse the business logic

We will define the functions with function keyword

We have following type of functions

1)Named functions

2)Anonymous Functions/ArrowFunctions/Callback functions/fat arrow functions/nameless

3)Rest Parameters in functions

4)Default parameters

5)Optional parameter

6)Generators

7)IIFE (Immediate Invokable functional expression)

8)constructor functions

//Named Functions

The function with the name is called named function

Syntax

1)Function Definition

2)Function Calling

1)Function Definition

Function functionname(arg1,arg2,arg3….argn){

Business logic

}

2)Function Calling

Functionname(parm1,parm2,parm3…paramn);

<script>

    function fun\_one(arg1,arg2){

        console.log(arg1,arg2);

    }

    fun\_one("vamshi","shiva");

    fun\_one("vamshi","shiva");

    fun\_one("vamshi","shiva");

    fun\_one();  //undefined

    fun\_one(null,null);

    fun\_one(undefined,"harish");

</script>

Check this imp

<script>

    function fun\_one(){

        return fun\_two;     //check no bracket thatwhy ()()

    }

    function fun\_two(){

        return "Hello";

    }

    console.log(fun\_one()()); //hello;

</script>

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

<script>

    let arr=[];

    function fun\_one(){

        return "Hello";

    }

    for(let i=0;i<5;i++){

        arr.push(fun\_one());

    }

    for(let j=0;j<arr.length;j++){

       console.log(arr[j]);

    }

</script>

<script>

    let arr=[];

    function fun\_one(){

        return "Hello";

    }

    for(let i=0;i<5;i++){

        arr.push(fun\_one);

    }

    for(let j=0;j<arr.length;j++){

        console.log(arr[j]());  //hello

    }

</script>

<script>

    function fun\_one(){

        return "Hello";

    }

    //it is executed only once that is setTimeout

    setTimeout(function(){

        console.log(fun\_one());

    },3000);

</script>

<!-- If we want to execute the function for every interval of time we need to use setInterval -->

<script>

    function fun\_one(){

        return "Hello";

    }

    setInterval(function(){

        console.log(fun\_one());

    },3000);

</script>

=================================

<script>

    for(let i=0;i<5;i++){

        setTimeout(function(){

            console.log(i);

        },3000);

    }

//var : 5 5 5 5 5

//let : 0 1 2 3 4

</script>

========================================================================

Rest parameters in Functions

It is used to store more than one value in single argument

… called spread operator

It is introduced in ES6.

1)We can take only one spread operator per function (…arg1) same as varargs in java

2)position of sread operator always last in occurances (arg1,...arg2)

<script>

    function fun\_one(...arg1){      //this ...is called "spread operator" Rest Parameter same as var agrs there we keep at the end

        console.log(arg1);           //we can take only one spread operator ...arg1...arg2 error

    };

    fun\_one("Hello1","Hello2");

    fun\_one("Hello1","Hello2","Hello3");

    fun\_one();

</script>

<script>

    function fun\_one(arg1,...arg2){

        console.log(arg1,arg2);

    };

    fun\_one();      //undefined[]

    fun\_one(undefined); //undefined[]

    fun\_one("Hello");   //Hello[]

    fun\_one("Hello1","Hello2","Hello3");//Hello ['Hello2','Hello3']

</script>

<script>

    function fun\_one(...arg1,arg2){     //error ...arg1 should be last parameter

    }

</script>

Default

<!-- Default parameters in function -->

<!-- While defining the functions, we will assign some default values to parameters. -->

 <script>

    function fun\_one(arg1="Hello1",arg2="Hello2"){

        console.log(arg1,arg2);

    }

    fun\_one();  //Hello1,Hello2

    fun\_one("ReactJS"); //ReactJS,Hello2

    fun\_one(undefined,undefined); //Hello1,Hello2

    fun\_one(null,null); //null,null

    fun\_one(undefined,"Welcome"); //Hello1,Welcome

 </script>

<script>

    function fun\_one(arg1,arg2="Hello2"){

        console.log(arg1,arg2);

    }

    fun\_one();  //undefined 'Hello2'

    fun\_one(undefined); //undefined'Hello2'

    fun\_one('Hello1'); //hello1,hello2

    fun\_one(null);//null,hello2

    fun\_one(undefined,undefined);   //undefined,hello2

    fun\_one(null,null); //null null

 </script>

<script>

    function fun\_one(arg1,arg2="Hello",...arg3){

        console.log(arg1,arg2,arg3);

    }

    fun\_one();      //undefined 'Hello' []

    fun\_one(undefined,undefined,undefined); //undefined,Hello,[undefined]

    fun\_one(null,null,null);    //null null [null]

    fun\_one("Hello1",undefined,"Hello3");

</script>

Arrow Functions

Arrow Functions:

The function without name called as Anonymous function.

Anonymous functions also called as arrow Functions

arrow functions behaves like callback function

we will represent arrow fucntion with =>

arrow function introduced in es6

syntax

1.function definition

2.function calling

var/let/const variablename=(arg1,arg2,arg3,....argn)=>{

    //bussiness logic

}

variable(para1,para2,.....paran);

 <script>

    let fun\_one=()=>{

        return "Hello";

    }

    console.log(fun\_one);   //fun def

    console.log(fun\_one()); //Hello

</script>

<script>

    let fun\_one=()=>"Hello";

    console.log(fun\_one());

</script>

<script>

    let fun\_one=(arg1,arg2,arg3)=>{

        console.log(arg1,arg2,arg3)

    }

    fun\_one("Hello1","Hello2","Hello3");//Hello1 Hello2 Hello3

    fun\_one();//undefined undefined undefined

    fun\_one(null,null,null);    //null null null

</script>

<script>

    let fun\_one=(arg1,arg2,arg3)=>{

        console.log(arg1(),arg2(),arg3());  //to get values bcz they are fun def we are calling arg1(),...

    }

    fun\_one(()=>{

        return "Hello1"

    },()=>{

        return "Hello2"

    },()=>{

        return "Hello3"         //Hello1 Hello2 Hello3

    });

</script>

<script>

    let intilize=(user,getElementsByEmail,getElementsById)=>{

        console.log(user,getElementsByEmail(),getElementsById());   //getElementsByEmail if write only this then it will only definition if we want o/p write getElementsByEmail()

    };

    let getElementsByEmail=()=>{

        return "hr@nikhl.com";

    }

    let getElementsById=()=>{

        return "Nikhil thula";

    }

    intilize("Hello1",getElementsByEmail,getElementsById);

</script>

<script>

    let arr=[];

    for(let i=0;i<5;i++){

        arr.push(()=>{

            return "Hello";

        })

    }

    for(let j=0;j<arr.length;j++){

        console.log(arr[j]());

    }

    //console.log(arr[0]());

</script>

IIFE

Immediate Invokable Functional Expression

es9

these functions called as self invokable Functions

syntax

((arg1,arg2,arg3,.....,argn)=>{

})(para1,para2,para3,.....paran);

<script>

    (()=>{      //Direct function calling

        console.log("Welcome to IIFE");

    })();

</script>

<script>

    let msg=(()=>{

        return "welcome to iife";

    })();

    console.log(msg);

</script>

<script>

    function fun\_one(arg1,arg2,arg3){

        console.log(arg1,arg2,arg3);

    }

    fun\_one((()=>{

        return "Hellooo"

    })());

</script>

<script>

    ((arg1,arg2)=>{

        console.log(arg1,arg2);

    })("vijay","ajay");

</script>

<script>

    for(let i=0;i<5;i++){

    setTimeout(()=>{

        console.log(i)

    },2000);

    }

</script>