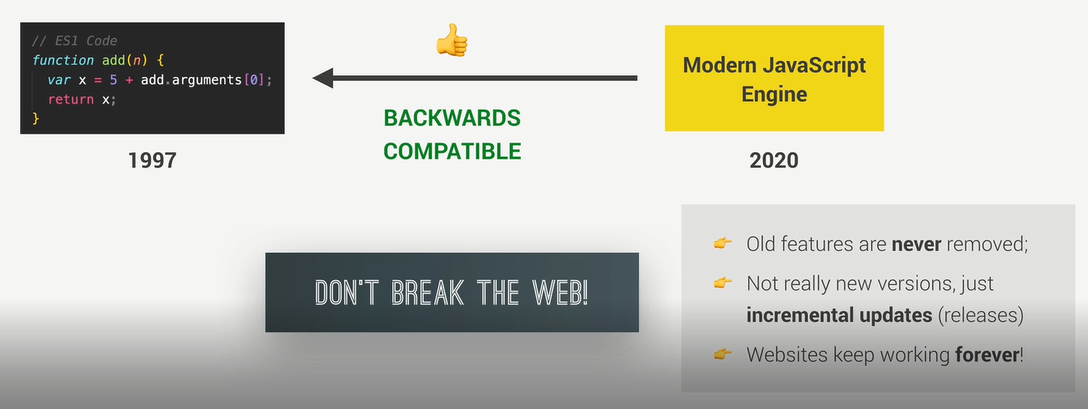
History of Javascript:

1995 Mocha>1996 Livescript>Javascript to attract java developers+ IE copy javascript from Netscape and named Jscript>ECMA relesease ECMA script 1(ES1) 1997>2009 ES5>2015 ES6



let bill=275;

        let tip=(bill>=50&& bill<=300)?bill\*.15:bill\*.2;

        console.log(`The customer have to pay ${bill+tip} `);

JavaScript has 3 types of scope:

* Block scope: let, const
* Function scope: function a(){--function scope---}
* Global scope
* {var a=3} can be accessed from outside but when function a(){var b=3} b can’b be accessed from outside

Strict mode: **“use strict”**

* It helps you to write cleaner code, like preventing you from using undeclared variables.
* in normal JavaScript, mistyping a variable name creates a new global variable. In strict mode, this will throw an error, making it impossible to accidentally create a global variable
* Normaly this keyword represents objects

"use strict";

function myFunction() {

alert(this);

}//in this program if we don’t use strict this will represent object window.But now I will undefined

// Assignment to a non-writable global NaN

var undefined = 5; // throws a TypeError

var Infinity = 5; // throws a TypeError

<script>

"use strict";

eval ("var x = 2");

alert (x); // This will cause an error

</script>

**String+intiger: both string**

**Intiger+intiger+string=integer+string**

 document.write(hello+10+2+"<br>");//output hello102

        document.write(10+2+hello+"<br>");//output 12hello

        document.write('<br>'+(10%2));//output 5

ternary:a?b write a: write b

//++n increase value at the same line

n++ increase vale in the next line

**for in for of:**

var car={

           name: "BMW",//key:value

           Model: "X3",

           Model\_Year: "2021"

       };

       var numbers=new Array();

        numbers.push(10);

        numbers.push(20);

        numbers.push(30);

for(x of numbers){

            document.write(x+'<br>');//x=value not position

            //index associative array

            //array location:index

        }

for(x of car){

            //won't work in object

             document.write(x+'<br>');//nothing will appear

        }

for(x in car){

            document.write(car[x]+'<br>');//output property value

           document.write(x+'<br>');//x= propertyname like name,model

            //document.write(x+'<br>');

        }

 for(x in numbers){

            document.write(x+'<br>');//position value

            document.write(numbers[x]+'<br>');

        }

**Hoisting:**  Hoisting is the default behavior of moving all the declarations at the top of the scope before code execution;

  x = 5; // Assign 5 to x

        document.write(x);                   // Display x in the element

        var x; // Declare x

**Type conversion:** Number()

Number("3.14")    // returns 3.14  
Number(" ")       // returns 0  
Number("")        // returns 0  
Number("99 88")   // returns NaN

Number("99,88")   // returns NaN

String(), x.toString(),(12).toString()

let x = 9.656;//return string value  
x.toFixed(0);//10  
x.toFixed(2);//9.66

let x = 9.656;//return string  
x.toPrecision();//9.656  
x.toPrecision(2);//9.7

parseFloat()//covert string and returns number

parseInt() parses a string and returns a whole number

Date Methods:

|  |  |
| --- | --- |
| **Method** | **Description** |
| getFullYear() | Get the **year** as a four digit number (yyyy) |
| getMonth() | Get the **month** as a number (0-11) |
| getDate() | Get the **day** as a number (1-31) |
| getHours() | Get the **hour** (0-23) |
| getMinutes() | Get the **minute** (0-59) |
| getSeconds() | Get the **second** (0-59) |
| getMilliseconds() | Get the **millisecond** (0-999) |
| getTime() | Get the time (milliseconds since January 1, 1970) |
| getDay() | Get the weekday as a number (0-6) |
| Date.now() | Get the time. ECMAScript 5. |

const d = new Date();//value of d=Fri Dec 17 2021 23:11:43 GMT+0600 (Bangladesh Standard Time)

d.getFullYear();//2021

d.getMonth();//now month December means output 11

d.getDay();//5

const months = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"];  
  
const d = new Date();  
let month = months[d.getMonth()];//december

This keyword: points to the owner. In dom points to the element where event handler added

In object it points to the object

 var method={

            name:"akash",

            displayName:function(){

                document.write(*this*.name);//this means method

            }

        }

**array:**

arrayName.shift()—remove an element from the beginnig array and return the item

arrayName.pop()—add elements at the end of an array and return array length

arrayName.push()—add elements at the end of an array and return array length

arryName.unshift()—add items to the beginning of the array and overwrite it

arrayName.slice(position to add or remove, no of items to be remove, items list to add

**This keyword:**

**Jar syntax e ekta call kora ji take nirdesh kore**

 let jonas={

            firstName:"jonas",

            calcAge:function(){

                console.log(*this*);//Jonas

            },

            calc:()=>{

                console.log(*this*);//ans window bcz arrow function has no block // arrow function er modthe lekha r Jonas object er modhe lekha aki kotha

            },

            calce:console.log(*this*)//ans window

//object has don’t create block also

        }

        jonas.calcAge();

        jonas.calc();

        jonas.calce;

 function john (){

                console.log(*this*);//functions has no block also but it represts the context whom it runs with

//with strict undefined without strict window

            };

        john();

**Dom: document object model Structural representation of html elments api te thake js dara jake access kora ji.api onek ta buit-in library er moto kaj kore**

**Document.queryselector(‘.classname’).textContent,.value=**