**ES6 features**

1. let

The let keyword allows you to declare a variable with block scope.

var x = 10;  
// Here x is 10  
{  
  let x = 2;  
  // Here x is 2  
}  
// Here x is 10

2. const

The let keyword allows you to declare a variable with block scope.

var x = 10;  
// Here x is 10  
{  
  const x = 2;  
  // Here x is 2  
}  
// Here x is 10

3. Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don’t need to the **function** Keyword, the **return** keyword, and the **curly brackets**.

// ES5  
var x = function(x, y) {  
   return x \* y;  
}  
  
// ES6  
const x = (x, y) => x \* y;

4. The Spread (…) Operator

The … operator expands an iterable (like an array) into more elements:

const q1 = ["Jan", "Feb", "Mar"];  
const q2 = ["Apr", "May", "Jun"];  
const q3 = ["Jul", "Aug", "Sep"];  
const q4 = ["Oct", "Nov", "May"];  
  
const year = [...q1, ...q2, ...q3, ...q4];

5. The For/Of Loop

The JavaScript for/of statement loops through the values of an iterable objects.

for/of lets you loop over data structures that are iterable such as Arrays, Strings, Maps, NodeLists, and more.

const cars = ["BMW", "Volvo", "Mini"];  
let text = "";  
  
for (let x of cars) {  
  text += x + " ";  
}

6. JavaScript Maps

const fruits = new Map([  
["apples", 500],  
["bananas", 300],  
["oranges", 200]  
]);

7. JavaScript Sets

// Create a Set  
const letters = new Set();  
  
// Add some values to the Set  
letters.add("a");  
letters.add("b");  
letters.add("c");

8. JavaScript Classes

JavaScript Classes are templates for JavaScript Objects.

Use the keyword class to create a class.

Always add a method named constructor():

class Car {  
  constructor(name, year) {  
    this.name = name;  
    this.year = year;  
  }  
}

9. JavaScript Promises

A Promise is a JavaScript object that links "Producing Code" and "Consuming Code".

"Producing Code" can take some time and "Consuming Code" must wait for the result.

const myPromise = new Promise(function(myResolve, myReject) {  
// "Producing Code" (May take some time)  
  
  myResolve(); // when successful  
  myReject();  // when error  
});  
  
// "Consuming Code" (Must wait for a fulfilled Promise).  
myPromise.then(  
  function(value) { /\* code if successful \*/ },  
  function(error) { /\* code if some error \*/ }  
);

10. Default Parameter Values

ES6 allows function parameters to have default values.

function myFunction(x, y = 10) {  
  // y is 10 if not passed or undefined  
  return x + y;  
}  
myFunction(5); // will return 15

11. Function Rest Parameter

The rest parameter (...) allows a function to treat an indefinite number of arguments as an array:

function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}  
  
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);

12. String.includes()

The includes() method returns true if a string contains a specified value, otherwise false:

let text = "Hello world, welcome to the universe.";  
text.includes("world")    // Returns true

13. String.startWith()

The startsWith() method returns true if a string begins with a specified value, otherwise false:

let text = "Hello world, welcome to the universe.";  
  
text.startsWith("Hello")   // Returns true

14. String.endsWith()

The endsWith() method returns true if a string ends with a specified value, otherwise false:

var text = "John Doe";  
text.endsWith("Doe")    // Returns true

15. Array.from()

The Array.from() method returns an Array object from any object with a length property or any iterable object.

Array.from("ABCDEFG")   // Returns [A,B,C,D,E,F,G]

16. Array keys()

The keys() method returns an Array Iterator object with the keys of an array.

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
const keys = fruits.keys();  
  
let text = "";  
for (let x of keys) {  
  text += x + "<br>";  
}

17. Array find()

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

This example finds (returns the value of ) the first element that is larger than 18:

const numbers = [4, 9, 16, 25, 29];  
let first = numbers.find(myFunction);  
  
function myFunction(value, index, array) {  
  return value > 18;  
}

18. Array findIndex()

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

This example finds the index of the first element that is larger than 18:

const numbers = [4, 9, 16, 25, 29];  
let first = numbers.findIndex(myFunction);  
  
function myFunction(value, index, array) {  
  return value > 18;  
}

19. New Math Methods

ES6 added the following methods to the Math object:

* Math.trunc()

Math.trunc(4.9);    // returns 4  
Math.trunc(4.7);    // returns 4  
Math.trunc(4.4);    // returns 4

* Math.sign()

Math.sign(-4);    // returns -1  
Math.sign(0);    // returns 0  
Math.sign(4);    // returns 1

* Math.cbrt() cube root

Math.cbrt(8);    // returns 2  
Math.cbrt(64);    // returns 4  
Math.cbrt(125);    // returns 5

* Math.log2()

Math.log2(2);    // returns 1 returns the base logarithm of x

* Math.log10()

Math.log10(10);   // returns 1 returns the base 10 logarithm of x

20. New Number Properties

ES6 added the following properties to the Number object:

* EPSILON
* MIN\_SAFE\_INTEGER
* MAX\_SAFE\_INTEGER

21. New Number Methods

ES6 added 2 new methods to the Number object:

* Number.isInteger()

Number.isInteger(10);        // returns true  
Number.isInteger(10.5);      // returns false

* Number.isSafeInteger()

Number.isSafeInteger(10);    // returns true  
Number.isSafeInteger(12345678901234567890);  // returns false

22. New Global Methods

ES6 added 2 new global number methods:

* isFinite()
* isNaN()

23. Object entries()

The entries() method returns an Array Iterator object with key/value pairs:

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
const f = fruits.entries();  
  
for (let x of f) {  
  document.getElementById("demo").innerHTML += x;  
}

**Outout:**

[0, "Banana"]  
[1, "Orange"]  
[2, "Apple"]  
[3, "Mango"]

24. Modules

Modules are imported in two differen ways:

Import from named exports

Import named exports from the file person.js:

import { name, age } from "./person.js";

### Import from default exports

Import a default export from the file message.js:

import message from "./message.js";