New Features in ES6

* [The let keyword](https://www.w3schools.com/js/js_es6.asp#mark_let)
* [The const keyword](https://www.w3schools.com/js/js_es6.asp#mark_const)
* [Arrow Functions](https://www.w3schools.com/js/js_es6.asp#mark_arrow)
* [The ... Operator](https://www.w3schools.com/js/js_es6.asp#mark_spread)
* [For/of](https://www.w3schools.com/js/js_es6.asp#mark_forof)
* [Map Objects](https://www.w3schools.com/js/js_es6.asp#mark_map)
* [Set Objects](https://www.w3schools.com/js/js_es6.asp#mark_set)
* [Classes](https://www.w3schools.com/js/js_es6.asp#mark_class)
* [Promises](https://www.w3schools.com/js/js_es6.asp#mark_promise)
* [Symbol](https://www.w3schools.com/js/js_es6.asp#mark_symbol)
* [Default Parameters](https://www.w3schools.com/js/js_es6.asp#mark_param)
* [Function Rest Parameter](https://www.w3schools.com/js/js_es6.asp#mark_rest)
* [String.includes()](https://www.w3schools.com/js/js_es6.asp#mark_includes)
* [String.startsWith()](https://www.w3schools.com/js/js_es6.asp#mark_startswith)
* [String.endsWith()](https://www.w3schools.com/js/js_es6.asp#mark_endswith)
* [Array.from()](https://www.w3schools.com/js/js_es6.asp#mark_array_from)
* [Array keys()](https://www.w3schools.com/js/js_es6.asp#mark_array_keys)
* [Array find()](https://www.w3schools.com/js/js_es6.asp#mark_array_find)
* [Array findIndex()](https://www.w3schools.com/js/js_es6.asp#mark_array_findIndex)
* [New Math Methods](https://www.w3schools.com/js/js_es6.asp#mark_math_methods)
* [New Number Properties](https://www.w3schools.com/js/js_es6.asp#mark_number_properties)
* [New Number Methods](https://www.w3schools.com/js/js_es6.asp#mark_number_methods)
* [New Global Methods](https://www.w3schools.com/js/js_es6.asp#mark_global_methods)
* [Object entries](https://www.w3schools.com/js/js_es6.asp#mark_entries)
* [JavaScript Modules](https://www.w3schools.com/js/js_es6.asp#mark_modules)

## JavaScript let

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

<!DOCTYPE html>

<html>

<body>

<h2>Redeclaring a Variable Using let</h2>

<p id="demo"></p>

<script>

let x = 10;

// Here x is 10

{

let x = 2;

// Here x is 2

}

// Here x is 10

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

</script>

</body>

</html>

## JavaScript const

The const keyword allows you to declare a constant (a JavaScript variable with a constant value).

Constants are similar to let variables, except that the value cannot be changed.

<!DOCTYPE html>

<html>

<body>

<h2>Declaring a Variable Using const</h2>

<p id="demo"></p>

<script>

var  x = 10;

// Here x is 10

{

  const x = 2;

  // Here x is 2

}

// Here x is 10

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

</script>

</body>

</html>

## Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don't need the function keyword, the return keyword, and the **curly brackets**.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrow Functions</h2>

<p>With arrow functions, you don't have to type the function keyword, the return keyword, and the curly brackets.</p>

<p id="demo"></p>

<script>

const x = (x, y) => x \* y;

document.getElementById("demo").innerHTML = x(5, 5);

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrow Functions</h2>

<p>With arrow functions, you don't have to type the function keyword, the return keyword, and the curly brackets.</p>

<p id="demo"></p>

<script>

const x = (x, y) => {

  return x \* y };

document.getElementById("demo").innerHTML = x(5, 5);

</script>

</body>

</html>

## The Spread (...) Operator

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

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<h2>The ... Operator</h2>

<p>The "spread" operator spreads elements of iterable objects:</p>

<p id="demo"></p>

<script>

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];

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

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Operators</h1>

<h2>The ... Operator</h2>

<p>The "Spread" operator can be used to expand an iterable into more arguments for function calls:</p>

<p id="demo"></p>

<script>

const numbers = [23,55,21,87,56];

let maxValue = Math.max(...numbers);

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

</script>

</body>

</html>

## 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.

The for/of loop has the following syntax:

for (*variable* of *iterable*) {  
  // *code block to be executed*  
}

variable - For every iteration the value of the next property is assigned to the variable. Variable can be declared with const, let, or var.

iterable - An object that has iterable properties.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript For Of Loop</h2>

<p>The for of statement loops through the values of any iterable object:</p>

<p id="demo"></p>

<script>

const cars = ["BMW", "Volvo", "Mini"];

let res="";

for (const iterator of cars) {

  res=res+iterator+"<BR>";

}

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

</script>

</body>

</html>

### **Looping over a String**

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript For Of Loop</h2>

<p>The for of statement loops through the values of an iterable object.</p>

<p id="demo"></p>

<script>

let language = "JavaScript";

let text = "";

for (let x of language) {

  text += x + "<br>";

}

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

</script>

</body>

</html>

## The For In Loop

The JavaScript for in statement loops through the properties of an Object:

### **Syntax**

for (key in object) {  
  // *code block to be executed*  
}

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript For In Loop</h2>

<p>The for in statement loops through the properties of an object:</p>

<p id="demo"></p>

<script>

const person = {fname:"John", lname:"Doe", age:25};

let txt = "";

for (let x in person) {

  txt += person[x] + " ";

}

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

</script>

</body>

</html>

* The **for in** loop iterates over a **person** object
* Each iteration returns a **key** (x)
* The key is used to access the **value** of the key
* The value of the key is **person[x]**

## For In Over Arrays

The JavaScript for in statement can also loop over the properties of an Array:

### **Syntax**

for (variable in array) {  
  code  
}

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Arrays</h1>

<h2>For In Loops</h2>

<p>The for in statement can loops over array values:</p>

<p id="demo"></p>

<script>

const numbers = [45, 4, 9, 16, 25];

let txt = "";

for (let x in numbers) {

  txt += numbers[x] + "<br>";

}

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

</script>

</body>

</html>

## JavaScript Maps

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Map Objects</h2>

<p>Creating a Map from an Array:</p>

<p id="demo"></p>

<script>

// Create a Map

const fruits = new Map([

  ["apples", 500],

  ["bananas", 300],

  ["oranges", 200]

]);

document.getElementById("demo").innerHTML = fruits.get("apples");

</script>

</body>

</html>

## JavaScript Sets

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Sets</h2>

<p>Add values to a Set:</p>

<p id="demo"></p>

<script>

// Create a Set

const letters = new Set();

// Add Values to the Set

letters.add("a");

letters.add("b");

letters.add("c");

// Display set.size

document.getElementById("demo").innerHTML = letters.size;

</script>

</body>

</html>

## JavaScript Classes

JavaScript Classes are templates for JavaScript Objects.

Use the keyword class to create a class.

Always add a method named constructor():

### **Syntax**

class ClassName {  
  constructor() { ... }  
}

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Classes</h1>

<p>Creating two car objects from a car class:</p>

<p id="demo"></p>

<script>

class Car {

  constructor(name, year) {

    this.name = name;

    this.year = year;

  }

}

const myCar1 = new Car("Ford", 2014);

const myCar2 = new Car("Audi", 2019);

document.getElementById("demo").innerHTML =

myCar1.name + " " + myCar2.name;

</script>

</body>

</html>

## 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.

### **Promise Syntax**

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 \*/ }  
);

Example:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Promise</h2>

<p>Wait 3 seconds (3000 milliseconds) for this page to change.</p>

<h1 id="demo"></h1>

<script>

const myPromise = new Promise(function(myResolve, myReject) {

setTimeout(function(){ myResolve("HI !!"); }, 3000);

});

myPromise.then(function(value) {

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

});

</script>

</body>

</html>