Javascript ES6

‹ Previous Next ›

ECMAScript 2015 was the second major revision to JavaScript.

ECMAScript 2015 is also known as ES6 and ECMAScript 6.

This chapter describes the most important features of ES6.

New Features in ES6

- The let keyword
- The const keyword
- Arrow Functions
- For/of
- Map Objects
- Set Objects
- Classes
- Promises
- Symbol
- Default Parameters
- Function Rest Parameter
- String.includes()
- String.startsWith()
- String.endsWith()

- Array.from()
- Array.keys()
- Array.find()
- Array.findIndex()
- New Math Methods
- New Number Properties
- New Number Methods
- New Global Methods
- Iterables Object.entries
- JavaScript Modules

Browser Support for ES6 (2015)

Safari 10 and Edge 14 were the first browsers to fully support ES6:

Chrome 58	Edge 14	Firefox 54	Safari 10	Opera 55
Jan 2017	Aug 2016	Mar 2017	Jul 2016	Aug 2018

JavaScript let

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

Example

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

Try it Yourself »

Read more about `let` in the chapter: JavaScript Let.

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.

Example

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

Try it Yourself »

Read more about `const` in the chapter: JavaScript Const.

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**.

Example

```
// ES5
var x = function(x, y) {
   return x * y;
}
```

```
// ES6
const x = (x, y) => x * y;
```

Try it Yourself »

Arrow functions do not have their own `this`. They are not well suited for defining **object methods**.

Arrow functions are not hoisted. They must be defined **before** they are used.

Using `const` is safer than using `var`, because a function expression is always a constant value.

You can only omit the `return` keyword and the curly brackets if the function is a single statement. Because of this, it might be a good habit to always keep them:

Example

```
const x = (x, y) => { return x * y };
```

Try it Yourself »

Learn more about Arrow Functions in the chapter: JavaScript Arrow Function.

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.

Looping over an Array

Example

```
const cars = ["BMW", "Volvo", "Mini"];
let text = "";

for (let x of cars) {
  text += x + " ";
}
```

Try it Yourself »

Looping over a String

Example

```
let language = "JavaScript";
let text = "";

for (let x of language) {
    text += x + " ";
}
```

Try it Yourself »

Learn more in the chapter: JavaScript Loop For/In/Of.

JavaScript Map Objects

Being able to use an Object as a key is an important Map feature.

Example

```
// Create Objects
const apples = {name: 'Apples'};
const bananas = {name: 'Bananas'};
const oranges = {name: 'Oranges'};

// Create a new Map
const fruits = new Map();

// Add new Elements to the Map
fruits.set(apples, 500);
fruits.set(bananas, 300);
fruits.set(oranges, 200);
```

Try it Yourself »

Learn more about Map objects in the the chapter: JavaScript Map().

JavaScript Set Objects

Example

```
// Create a Set
const letters = new Set();

// Add some values to the Set
letters.add("a");
letters.add("b");
letters.add("c");
```

Try it Yourself »

Learn more about Set objects in the the chapter: JavaScript Set().

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() { ... }
}
```

Example

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

The example above creates a class named "Car".

The class has two initial properties: "name" and "year".

A JavaScript class is **not** an object.

It is a **template** for JavaScript objects.

Using a Class

When you have a class, you can use the class to create objects:

Example

```
const myCar1 = new Car("Ford", 2014);
const myCar2 = new Car("Audi", 2019);
```

Try it Yourself »

Learn more about classes in the the chapter: JavaScript Classes.

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 Using a Promise

```
const myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function() { myResolve("I love You !!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = value;
});
```

Try it Yourself »

Learn more about Promises in the the chapter: JavaScript Promises.

The Symbol Type

A JavaScript Symbol is a primitive datatype just like Number, String, or Boolean.

It represents a unique "hidden" identifier that no other code can accidentally access.

For instance, if different coders want to add a person.id property to a person object belonging to a third-party code, they could mix each others values.

Using Symbol() to create a unique identifiers, solves this problem:

Example

```
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};

let id = Symbol('id');
person[id] = 140353;
// Now Person[id] = 140353
```

```
// but person.id is still undefined
```

Try it Yourself »

Symbols are always unique.

If you create two symbols with the same description they will have different values.

```
Symbol("id") == Symbol("id") // false
```

Default Parameter Values

ES6 allows function parameters to have default values.

Example

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

Try it Yourself »

Function Rest Parameter

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

Example

```
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);
```

Try it Yourself »

String.includes()

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

Example

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

Try it Yourself »

String.startsWith()

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

Example

```
let text = "Hello world, welcome to the universe.";

text.startsWith("Hello")   // Returns true
```

Try it Yourself »

String.endsWith()

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

Example

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

Try it Yourself »

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:

Example

```
const numbers = [4, 9, 16, 25, 29];
let first = numbers.find(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

Try it Yourself »

Note that the function takes 3 arguments:

- The item value
- The item index

- The array itself

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:

Example

```
const numbers = [4, 9, 16, 25, 29];
let first = numbers.findIndex(myFunction);

function myFunction(value, index, array) {
  return value > 18;
}
```

Try it Yourself »

Note that the function takes 3 arguments:

- The item value
- The item index
- The array itself

New Math Methods

ES6 added the following methods to the Math object:

- `Math.trunc()`
- `Math.sign()`
- `Math.cbrt()`
- `Math.log2()`
- `Math.log10()`

The Math.trunc() Method

`Math.trunc(x)` returns the integer part of x:

Example

```
Math.trunc(4.9);    // returns 4
Math.trunc(4.7);    // returns 4
Math.trunc(4.4);    // returns 4
Math.trunc(4.2);    // returns 4
Math.trunc(-4.2);    // returns -4
```

Try it Yourself »

The Math.sign() Method

`Math.sign(x)` returns if x is negative, null or positive:

Example

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

Try it Yourself »

The Math.cbrt() Method

`Math.cbrt(x)` returns the cube root of x:

Example

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

```
Math.cbrt(125);     // returns 5
```

Try it Yourself »

The Math.log2() Method

`Math.log2(x)` returns the base 2 logarithm of x:

Example

```
Math.log2(2);     // returns 1
```

Try it Yourself »

The Math.log10() Method

`Math.log10(x)` returns the base 10 logarithm of x:

Example

```
Math.log10(10);     // returns 1
```

Try it Yourself »

New Number Properties

ES6 added the following properties to the Number object:

- EPSILON
- MIN_SAFE_INTEGER
- MAX_SAFE_INTEGER

Example

```
let x = Number.EPSILON;
```

Try it Yourself »

Example

```
let x = Number.MIN_SAFE_INTEGER;
```

Try it Yourself »

Example

```
let x = Number.MAX_SAFE_INTEGER;
```

Try it Yourself »

New Number Methods

ES6 added 2 new methods to the Number object:

- `Number.isInteger()`
- `Number.isSafeInteger()`

The Number.isInteger() Method

The `Number.isInteger()` method returns `true` if the argument is an integer.

Example

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

Try it Yourself »

The Number.isSafeInteger() Method

A safe integer is an integer that can be exactly represented as a double precision number.

The `Number.isSafeInteger()` method returns `true` if the argument is a safe integer.

Example

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

Try it Yourself »

Safe integers are all integers from $-(2^{53} - 1)$ to $+(2^{53} - 1)$.
This is safe: 9007199254740991. This is not safe: 9007199254740992.

New Global Methods

ES6 added 2 new global number methods:

- `isFinite()`
- `isNaN()`

The isFinite() Method

The global `isFinite()` method returns `false` if the argument is `Infinity` or `NaN`.

Otherwise it returns `true`:

Example

```
isFinite(10/0);       // returns false
isFinite(10/1);       // returns true
```

Try it Yourself »

The isNaN() Method

The global `isNaN()` method returns `true` if the argument is `NaN`. Otherwise it returns `false`:

Example

```
isNaN("Hello");       // returns true
```

Try it Yourself »

❮ Previous Next ❯

LIKE US

Get started

CODE GAME

Play Game

Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
Angular Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Web Courses

HTML Course
CSS Course
JavaScript Course
Front End Course
SQL Course
Python Course
PHP Course
jQuery Course
Java Course
C++ Course
C# Course
XML Course

Get Certified »