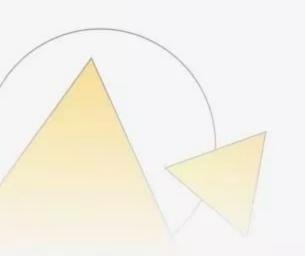


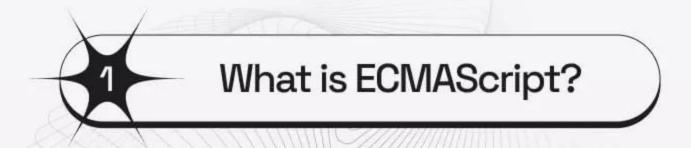


Understanding MODERN JAVASCRIPT

The Essential History of JavaScript From ES6 & Beyond!



Swipe 🗘



ECMAScript is a scripting language specification and it determines how JavaScript works and what new features it should have. Features are usually released incrementally each year to ensure smooth updates to the JavaScript programming language.

The **most notable** release of the **ES** standard was done in the year **2015**, with the release of **ES6**, which brought a lot of the amazing JavaScript features we use today in **modern** code.

This post will help you **explore** and **understand** the shifts in the **JavaScript** programming language throughout the years, **showcasing** key features that have been part of **past releases**.

Let's get started!





The **biggest year** for **JavaScript**, this release includes some really **important** updates that any **JS developer** should know!

Key Features

The let & const keywords for variable declaration

Template Literals

Destructuring

Spread Syntax

New Array Iteration

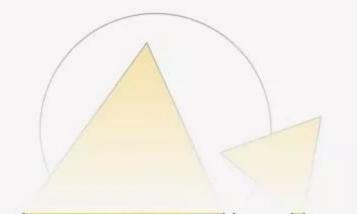
Promises

Classes & Modules

Arrow Functions

Default Parameters

This update changed **JavaScript** by a lot, so let's see how it **compares** to the old version **(ES5 - Released in 2011)**





1. Template Literals

```
let first = 'david';
let second = 'webdev';

let fullES5Name = first + '.' + second; // ES5
let fullES6Name = `${first}.${second}`; // ES6
```

Embed expressions in strings with a cleaner syntax!

2. Destructuring

```
const list = [4, 5, 6];

// ES5
const first = list[0];
const second = list[1];

// ES6
const [one, two, ...rest] = list;
console.log(one, two, rest); // 4 5 [6];
```

Unpack values from arrays or properties from objects into distinct variables! Heavily used in React

3. New Array Iteration

```
const list = [4, 5, 6];

// ES5
for (let i = 0; i <= list.length; i++) {
    console.log(list[i])
}

// ES6
for (i of list) {
    console.log(i);
}</pre>
```

A simpler and more declarative way to iterate over arrays.





Array.prototype.includes()

```
const list = ['bread', 'cheese', 'ham'];
list.includes('bread') // returns true
list.includes('tomatoes') // returns false
```

A new, easy way to check if an array includes a certain value

The Exponentiation Operator

```
// ES6
Math.pow(2, 2) // 4

// ES7
2 ** 2 // 4
2 ** (2 ** 2) // 16
```

A new math operator which returns the result of the first operand raised to the power of the second operand





Object.values() & Object.entries()

```
const person = { name: 'Matt', age: 23 };
console.log(Object.entries(person)); // [['name', 'Matt'], ['age', 23]]
console.log(Object.values(person)); // ['Matt', 23]
```

Trailing Commas

```
const person = {
  name: 'Matt',
  age: 23,
};
```

Trailing commas allow us to add new lines to an object without having to modify the previous line since it already has a comma

String .padStart() & padEnd() methods

Asynchronous Functions (Async/Await)





Asynchronous Generators & Iterators

Object Rest & Spread Operators

```
const obj1 = { one: 10, two: 20 }
const obj2 = { three: 30 }

// Clone obj1
const clone = { ...obj1 }

// Combine obj1 & obj2
const obj3 = { ...obj1, ...obj2 }
// { one: 10, two: 20, three: 30 }
```

An easier way to clone, combine and work with objects in general!

Promise.prototype.finally()

RegExp Features (Named Capture Groups, dotAll)



Array.flat() & Array.flatMap()

Object.fromEntries()

```
const data = [
   ['filename', 'ecmascript.txt'],
   ['date', new Date()]
]

const file = Object.fromEntries(data);

console.log(file);
// { filename: 'ecmascript.txt', date: 2022-09-29T11:51:19.931Z }
```

Create objects from an array of entries!

String .trimStart() & trimEnd() methods

Optional Catch Binding (Catch Block Error Param)



Private Class Variables

```
class Person {
    #birthyear = 1998
    age() { console.log(2022 - this.#birthyear) }
}

const david = new Person()
david.age() // 24
console.log(david.#birthyear) // Error
```

Private class variables are declared using a hash in front of the variable or function name. Accessing them outside the class will throw an error.

Promise.allSettled() method

String.prototype.matchAll()

Optional Chaining Operator

```
let car = { color: 'red' };
let interiorColor = car?.interiorColor;
console.log(interiorColor); //undefined
```

Prevent errors in your code by using optional chaining!

