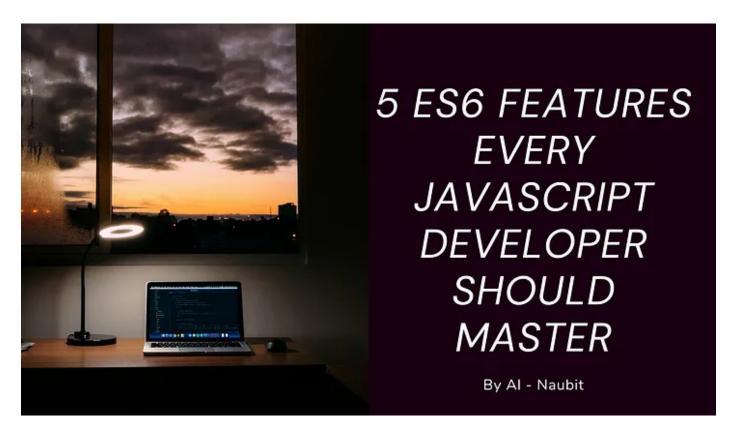


# 



New day, new article! Today's article is about **five advanced Javascript ES6 features** that I like and that I think everyone (*at least every developer*) should understand.

Are you ready?



Destructuring is a quick way to get values out of objects and arrays. For example, **you can extract values** and assign them to variables with a single line of code.

Here's an example of how destructuring can be used with an object:

```
1
   const person = {
2
      name: 'John Doe',
3
    age: 32,
    location: 'San Francisco'
4
   };
5
6
   const { name, age, location } = person;
7
8
9
   console.log(name, age, location); // John Doe 32 San Francisco
                                                                                         view raw
2023020101.js hosted with ♥ by GitHub
```

And here's an example with an array:

```
1  const colors = ['red', 'green', 'blue'];
2
3  const [first, second, third] = colors;
4
5  console.log(first, second, third); // red green blue
2023020102.js hosted with ♥ by GitHub view raw
```

As you can see, destructuring makes it simple to extract values from objects and arrays and assign them to variables.

## Block Scoping

You can use block scoping to declare variables that are only available within a specific block of code. There are two ways to declare variables in JavaScript: var and let.

The var keyword declares a global or function-scoped variable, which means it can be accessed from anywhere within the same function. On the other hand, the let keyword declares a variable that is block scoped, which means that it can only be accessed within the same block of code.

Here's an example of let-based block scoping:

```
1  if (true) {
2    let message = 'Hello, world!';
3    console.log(message); // Hello, world!
4  }
5    
6  console.log(message); // Uncaught ReferenceError: message is not defined
2023020103.js hosted with ♥ by GitHub
view raw
```

As you can see, the message variable is only available within the if statement-defined block of code.

#### Spread Operator

Spreading the values of an array or object into a new array or object is possible with the spread operator. It's a quick way to combine arrays or objects or to turn an array-like object into a proper array.

Here's an example of how to combine two arrays using the spread operator:

Here's an example of how to use the spread operator to transform an array-like object into a real array:

```
const arrayLike = {
2
      0: 'one',
3
     1: 'two',
     2: 'three'
4
5
   };
6
7
   const realArray = [...arrayLike];
8
   console.log(realArray); // ['one', 'two', 'three']
                                                                                           view raw
2023020105.js hosted with ♥ by GitHub
```

A spread operator is a powerful tool for simplifying and improving the readability of your code.

## Template Literals

String literals that allow you to embed expressions within your strings are known as template literals. Instead of quotes (' or "), they are defined with the backtick (`) character.

Here's an example of template literals in action:

```
1  const name = 'John Doe';
2  const age = 32;
3
4  const message = `Hello, my name is ${name} and I am ${age} years old.`;
5
6  console.log(message); // Hello, my name is John Doe and I am 32 years old.
2023020106.js hosted with ♥ by GitHub
view raw
```

As you can see, template literals make it simple to embed expressions within strings and allow you to write multi-line strings without using string concatenation.

#### Arrow Functions

In JavaScript, arrow functions are a shorthand syntax for writing anonymous functions. They enable you to write code that is shorter, more concise, and more readable.

Here's an example of how to use the arrow function:

```
1  const numbers = [1, 2, 3, 4, 5];
2
3  const square = number ⇒ number * number;
4
5  const squares = numbers.map(square);
6
7  console.log(squares); // [1, 4, 9, 16, 25]
2023020107.js hosted with ♥ by GitHub
view raw
```

As you can see, arrow functions make it simple to write anonymous functions and have a shorter syntax than regular functions.

It was a short article, but I hope it was helpful for you. I use these features daily and feel like **they are crucial for every Javascript developer**. So hopefully, you have discovered something new today.

## Let's Connect!

- My Twitter: @thenaubit
- My Substack (here I will publish more in-denth articles)

Open in app  $\ \ \nearrow$ 









Javascript Development

# Get an email whenever AI - @thenaubit publishes.

Emails will be sent to amaurypv@gmail.com. Not you?

