AI - @thenaubit Follow

Feb 1 · 3 min read · ✦ · Listen

Save

🚀 5 Advanced ES6 Features Every JavaScript Developer Should Master

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

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:

```
const person = {
  name: 'John Doe',
  age: 32,
  location: 'San Francisco'
};

const { name, age, location } = person;

console.log(name, age, location); // John Doe 32 San Francisco
```

2023020101.js hosted with ❤️ by GitHub view raw

And here's an example with an array:

```
const colors = ['red', 'green', 'blue'];

const [first, second, third] = colors;

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:

```javascript
if (true) {
  let message = 'Hello, world!';
  console.log(message); // Hello, world!
}

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:

```javascript
const first = [1, 2, 3];
const second = [4, 5, 6];

const combined = [...first, ...second];

console.log(combined); // [1, 2, 3, 4, 5, 6]
```

2023020104.js hosted with ❤️ by GitHub view raw

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

```javascript
const arrayLike = {
  0: 'one',
  1: 'two',
  2: 'three'
};

const realArray = [...arrayLike];

console.log(realArray); // ['one', 'two', 'three']
```

2023020105.js hosted with ❤️ by GitHub view raw

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:

```javascript
const name = 'John Doe';
const age = 32;

const message = `Hello, my name is ${name} and I am ${age} years old.`;

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:

```
const numbers = [1, 2, 3, 4, 5];

const square = number => number * number;

const squares = numbers.map(square);

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-depth articles)

Javascript Development