18 JavaScript Tips: You Should Know for Clean and Efficient Code

devShefali · Follow
Published in Stackademic · 4 min read · Jan 22, 2024

419 6

In this post, I'll share 18 JavaScript tips, with examples that you should know for writing clean and efficient code.

Let's get started!🚀

Arrow Function

You can use arrow functions to simplify function declarations.

For example:

```
function add(a, b) {
  return a + b;
}
// Arrow function
const add = (a, b) => a + b;
```

Array.from()

The `Array.from()` method can be used to convert any iterable objects into arrays.

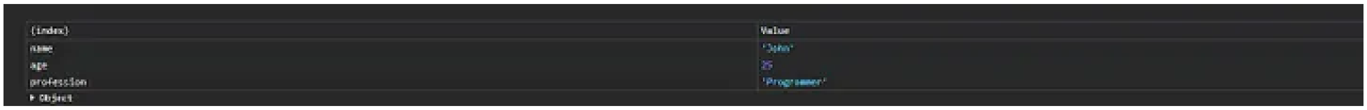
```
const str = "Hello!";
const arr = Array.from(str);
console.log(arr); //Output: ['H', 'e', 'l', 'l', 'o', '!']
```

Display Data with console.table()

If you want your data organized or in tabular format in the console, then you can use `console.table()`.

```
const person = {
    name: 'John',
    age: 25,
    profession: 'Programmer'
}
console.table(person);
```

Output:



(index)	Value
name	'John'
age	25
profession	'Programmer'

▶ Object

Use const and let effieciently

Use `const` for variables that won't be reassigned and `let` for those that will, for better code organization.

```
const PI = 3.14;
let timer = 0;
```

Extract Object Properties with Destructuring

By using destructuring to extract properties from objects, you can enhance code readability.

```
const person = {
    name: 'John',
    age: 25,
    profession: 'Programmer'
}

//Instead of this 👇
console.log(person.name);
console.log(person.age);
```

```
//Use this👇
const {name, age} = person;
console.log(name);
console.log(age);
```

Set Default Values with Logical OR Operator

Set default values easily using the `||` operator.

```
function greet(name) {
  name = name || 'Person';
  console.log(`Hello, ${name}!`);
}
greet(); //Output: Hello, Person!
greet("John"); //Output: Hello, John!
```

Open in app ↗

Search

Write

For example:

```
let numbers = [1, 2, 3, 4];
numbers.length = 0;
console.log(numbers); //Output: []
```

JSON.parse()

Use `JSON.parse()` to convert a JSON string into a JavaScript object, this ensures seamless data manipulation.

```
const jsonStr = '{"name": "John", "age": 25}';
const person = JSON.parse(jsonStr);
console.log(person);
//Output: {name: 'John', age: 25}
```

Map() Function

Use the `map()` function to transform elements in a new array without modifying the original array.

For example:

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(num => num * 2);

console.log(numbers); //Output: [1, 2, 3, 4]
console.log(doubled); //Output: [2, 4, 6, 8]
```

Object.seal()

You can use `Object.seal()` method to prevent adding or removing properties in the object.

```
const person = {
    name: 'John',
    age: 25
};
Object.seal(person);
person.profession = "Programmer";
console.log(person); //Output: {name: 'John', age: 25}
```

Object.freeze()

You can use `Object.freeze()` method to prevent any changes to an object, including adding, modifying or deleting properties.

```
const person = {
    name: 'John',
    age: 25
};
Object.freeze(person);
person.name = "Mark";
console.log(person); //Output: {name: 'John', age: 25}
```

Remove Array Duplicates

You can remove duplicate elements from an array using `Set`.

```
const arrWithDuplicates = [1, 12, 2, 13, 4, 4, 13];
const arrWithoutDuplicates = [...new Set(arrWithDuplicates)];
console.log(arrWithoutDuplicates);
//Output: [1, 12, 2, 13, 4]
```

Swap values using Destructuring

You can swap two variables easily using destructuring.

For example:

```
let x = 7, y = 13;
[x, y] = [y, x];
```

```
console.log(x); //13
```

Spread Operator

You can copy or merge arrays efficiently using the spread operator.

For example:

```
const arr1 = [1, 2, 3];
const arr2 = [9, 8, 7];

const arr3 = [...arr2];
const mergedArr = [...arr1, ...arr2];

console.log(arr3); //[9, 8, 7]
console.log(mergedArr); //[1, 2, 3, 9, 8, 7]
```

Template Interpolation

Utilize template literals for string interpolation and enhanced code readability.

For example:

```
const name = 'John';
const message = `Hello, ${name}!`;
```

Ternary Operator

You can simplify conditional statements with the ternary operator.

```
const age = 20;

//Instead of this👇
if(age>=18){
    console.log("You can drive");
}else{
    console.log("You cannot drive");
}

//Use this👇
age >= 18 ? console.log("You can drive") : console.log("You cannot drive");
```

Use === Instead of ==

Prevent type coercion issues by using strict equality (`===`) instead of loose equality (`==`).

```
const num1 = 5;
const num2 = '5';

//Instead of using ==
if (num1 == num2) {
  console.log('True');
} else {
  console.log('False');
}

//Use ===
if (num1 === num2) {
  console.log('True');
} else {
  console.log('False');
}
```

Use Descriptive Variable and Function Names

Use meaningful and descriptive names for variables and functions to enhance code readability and maintainability.

```
// Don't declare variable like this
const a = 18;

// use descriptive names
const numberOfTips = 18;
```

That’s all for today.

I hope it was helpful.

Thanks for reading.

For more content like this, click here.

You can also follow me on X(Twitter) for getting daily tips on web development.

Keep Coding!!

Stackademic

Thank you for reading until the end. Before you go:

- Please consider **clapping** and **following** the writer! 👏
- Follow us **X** | **LinkedIn** | **YouTube** | **Discord**
- Visit our other platforms: **In Plain English** | **CoFeed** | **Venture**

Web Development JavaScript Js Javascript Tips Clean Code

Written by devShefali

Follow

1.1K Followers · Writer for Stackademic

I am a passionate web developer from India. I find pleasure in talking about programming. I love to help everyone who needs any guidance related to programming.

More from devShefali and Stackademic