

JavaScript Shorthands Guide



javascript all shorthands with proper documentaion

Here's a comprehensive list of JavaScript shorthand techniques along with proper documentation and examples.

1. Variable Declaration Shorthand

Instead of:

```
js  
  
let a = 1;  
let b = 2;  
let c = 3;
```

Use:

```
js  
  
let a = 1, b = 2, c = 3;
```

2. Object Property Shorthand

Instead of:

```
js  
  
let name = "John";  
let age = 25;  
  
let person = {  
  name: name,  
  age: age  
};
```

Use:

```
js  
  
let person = { name, age };
```

3. Template Literals (String Concatenation)

Instead of:

```
js

let name = "Alice";
let greeting = "Hello, " + name + "!";
```

Use:

```
js

let greeting = `Hello, ${name}!`;
```

4. Arrow Functions (Function Expression Shorthand)

Instead of:

```
js

function add(a, b) {
  return a + b;
}
```

Use:

```
js

const add = (a, b) => a + b;
```

For single-parameter functions:

```
js

const square = x => x * x;
```

For single-line implicit return:

```
js

const greet = name => `Hello, ${name}`;
```

5. Ternary Operator (Conditional Shorthand)

Instead of:

```
js

let age = 20;
let status;

if (age >= 18) {
  status = "Adult";
}
```

```

} else {
  status = "Minor";
}

```

Use:

```

js

let status = age >= 18 ? "Adult" : "Minor";

```

6. Logical AND (`&&`) for Conditional Execution

Instead of:

```

js

if (isLoggedIn) {
  showDashboard();
}

```

Use:

```

js

isLoggedIn && showDashboard();

```

7. Logical OR (`||`) for Default Values

Instead of:

```

js

let name;
let userName = name ? name : "Guest";

```

Use:

```

js

let userName = name || "Guest";

```

8. Nullish Coalescing (`??`) for Safer Default Values

Instead of:

```

js

let count = someValue !== null && someValue !== undefined ? someValue : 10;

```

Use:

```
js

let count = someValue ?? 10;
```

Note: `||` would return `10` even if `someValue = 0`, but `??` only checks `null` and `undefined`.

9. Destructuring Assignment

Array Destructuring

Instead of:

```
js

let colors = ["red", "green", "blue"];
let first = colors[0];
let second = colors[1];
```

Use:

```
js

let [first, second] = colors;
```

Object Destructuring

Instead of:

```
js

let person = { name: "Alice", age: 30 };
let name = person.name;
let age = person.age;
```

Use:

```
js

let { name, age } = person;
```

10. Spread Operator (`...`)

Merging Arrays

Instead of:

```
js

let arr1 = [1, 2, 3];
let arr2 = [4, 5, 6];
```

```
let combined = arr1.concat(arr2);
```

Use:

```
js

let combined = [...arr1, ...arr2];
```

Copying Objects

Instead of:

```
js

let obj1 = { a: 1, b: 2 };
let obj2 = Object.assign({}, obj1, { c: 3 });
```

Use:

```
js

let obj2 = { ...obj1, c: 3 };
```

11. Default Function Parameters

Instead of:

```
js

function greet(name) {
  name = name || "Guest";
  console.log(`Hello, ${name}`);
}
```

Use:

```
js

function greet(name = "Guest") {
  console.log(`Hello, ${name}`);
}
```

12. Short-Circuiting with `||=` and `&&=`

Assign Default Value (`||=`)

Instead of:

```
js

if (!user.name) {
  user.name = "Anonymous";
}
```

```
}

```

Use:

```
js

user.name ||= "Anonymous";

```

Assign Only If Already Set (`&&=`)

Instead of:

```
js

if (user.isActive) {
  user.lastLogin = new Date();
}

```

Use:

```
js

user.isActive &&= new Date();

```

13. Optional Chaining (`?.`)

Instead of:

```
js

let street;
if (user && user.address) {
  street = user.address.street;
}

```

Use:

```
js

let street = user?.address?.street;

```

This prevents errors if `user` or `address` is `undefined` or `null`.

14. Immediately Invoked Function Expression (IIFE)

Instead of:

```
js

function init() {
  console.log("Initialized");
}

```

```
}
init();
```

Use:

```
js

(() => {
  console.log("Initialized");
})();
```

15. forEach, map, filter, reduce (Loop Shorthand)

`forEach` for Iteration

Instead of:

```
js

for (let i = 0; i < arr.length; i++) {
  console.log(arr[i]);
}
```

Use:

```
js

arr.forEach(item => console.log(item));
```

`map` for Transformation

Instead of:

```
js

let doubled = [];
for (let i = 0; i < numbers.length; i++) {
  doubled.push(numbers[i] * 2);
}
```

Use:

```
js

let doubled = numbers.map(num => num * 2);
```

`filter` for Filtering

Instead of:

```
js

let evens = [];
for (let i = 0; i < numbers.length; i++) {
  if (numbers[i] % 2 === 0) {
    evens.push(numbers[i]);
  }
}
```

Use:

```
js

let evens = numbers.filter(num => num % 2 === 0);
```

`reduce` for Aggregation

Instead of:

```
js

let sum = 0;
for (let i = 0; i < numbers.length; i++) {
  sum += numbers[i];
}
```

Use:

```
js

let sum = numbers.reduce((acc, num) => acc + num, 0);
```

16. Set Timeout Shorthand

Instead of:

```
js

setTimeout(function () {
  console.log("Hello");
}, 1000);
```

Use:

```
js

setTimeout(() => console.log("Hello"), 1000);
```

17. Convert String to Number Quickly

Instead of:


```
js  
  
let num = parseInt("42");
```

Use:

```
js  
  
let num = +"42";
```

These JavaScript shorthand techniques improve code readability, efficiency, and maintainability. Let me know if you want more details! 🚀

ChatGPT can make mistakes. Check important info.