IroncladDev
Posted on 26 Nov 2021

Six (Dirty) Ways to shorten your Javascript

#javascript #tutorial #programming #codequality

Shortening your code might not be the best way to do something, but it can help you out a lot if you are competing or trying to make a small javascript package! Here are ten tips I'll show you that you can use to make all your javascript files miniature.

Backticks instead of Parentheses

```
let splitStr = "this is a test"
let splitStr = str.split` `; // ['this', 'is', 'a', 'test']
```

This technique won't *dramatically* shorten your code but it does help if you are trying to minimize on code. It actually won't return that direct value. It returns it in an array instead. Here's the input/output of that if you use backticks:

```
console.log`Hello`; // ['Hello']
console.log`${1+1}`; // [ '', '' ] 2
console.log`My name is ${"John"}`; // [ 'My name is ', '' ] John
console.log`bleeeeh`; // [ 'bleeeeh' ]
console.log`${false}`; // [ '', '' ] false
```

It's weird but it gets the job done. Don't try passing values into a function that requires more than one argument. This will *not* work.

```
"Heyya Worlld".replace`yy`,`ll`
```

One `let` Declaration

This one isn't a super-secret one but it works well. Simply declare a variable and you can chain off of that.

```
let num = 0, dog = "dog", me = { age: 100, name: "Me" }, f = false, t = true;
```

Replace `true` and `false`

Literally, the booleans `true` and `false` can work as `1` and `0`. Only use this if you really want to shorten your code. It makes your JS a little harder to read.

```
let isThisTrue = 1;
let isThisAlsoTrue = true;

let isThisFalse = 0;
let isThisAlsoFalse = false;

if(isThisTrue) { ... } //executes
if(isThisAlsoTrue) { ... } //executes as well

if(isThisFalse || isThisAlsoFalse) { ... } //won't execute
```

Start using `for...of` and `for...in` loops

Typically, for...of and for...in loops perform slower and less efficient than normal loops, but if you are wanting a small code size, be sure them!!

```
let array = ["apple", "banana", "orange", "severed finger"];

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

//does the same thing as the above loop ^^
for(let i in array) {
  console.log(array[i])
}

//literally, "i" becomes the value
for(let i of array){
  console.log(i)
}
```

Let's go over the differences of each loop.
First, you should know that the first one simply loops through an array.

`for...in` loops will loop through objects, arrays, and even strings and return the variable (e.g. "i") as the index or object key.

```
let obj = {
  "a": 1,
  "b": 2,
  "c": 3
};
let arr = ["a", "b", "c", "d"];
let str = "abcde";

for(var i in obj){
  console.log(i);
} // logs "a", "b", and "c"

for(var j in arr){
  console.log(j);
} // logs 0, 1, 2, and 3

for(var k in str){
  console.log(k);
} //logs 0, 1, 2, 3, and 4
```

For...of loops behave slightly different. In these types of loops, "i" *becomes* the value in the array, or string. Objects are not iterable in for...of loops.

```
let arr = ["a", "b", "c", "d"];
let str = "abcde";

for(var j of arr){
  console.log(j);
} // logs "a", "b", "c", and "d"

for(var k of str){
  console.log(k);
} //logs "a", "b", "c", "d", and "e"
```

Use more javascript array functions

Array functions took me a while to grasp when I first started coding, but now I use them constantly. Functions like `.map`, `.forEach`, and `.reduce` can seriously shorten your code. You can replace loops with these sometimes.

Log items in an array:

```
let arr = ["a", "b", "c"];
arr.forEach(console.log);
```

Double String length in an array:

```
let arr = ["a", "b", "c"];
arr.map(value => value.repeat(2)); // ["aa", "bb", "cc"]
```

There are many more things you can do with array functions, but I won't fill this article with too much of them.

No declarations

This is probably the dirtiest javascript shortening trick ever. Don't try this on the client side of javascript unless you want to be bombarded with errors and also, server-side javascript sometimes doesn't allow it. It depends on where you're coding.

```
arr=["a","b","c","d"]
for(i in arr)console.log(i);
```

Declaring a variable without `const`, `let`, or `var` typically creates a non-constant variable like `let` or `var` that can be changed.

Tips

I guess this is the end of my article, but then I have a number of tricks up my sleeve for shortening your javascript.

1. Split string by each index

```
let str = "abc"
console.log([...str]) // prints ["a", "b", "c"]
```

2. Use `Number()` instead of `parseInt()`

It's shorter and performs better!!

```
let num = "10"
console.log(Number(parseInt))
```

3. Unique set of array items

```
let arr = ["a", "b", "b", "a", "c", "d", "c", "e"]
console.log([...new Set(arr)]); // [ 'a', 'b', 'c', 'd', 'e' ]
```

4. Crash a browser

```
while(true)window.open("https://dev.to")
```

5. Split an array on different characters

```
let str = "aaabccdddddefffgggh"
console.log(srt.match(/(.)\1*/g)); // logs ["aaa", "b", "cc", "ddddd", "e", "fff",
"ggg", "h"]
```

I hope you enjoyed my article.
Have fun, don't try the browser crasher, and make sure to still use declarations when necessary.
Be sure to follow and subscribe.

Discussion (5)

DarkWiiPlayer • Nov 26 '21

```
arr.forEach(console.log);
```

If you *really* want to golf your code, you could also replace `forEach` with `map`, since this will have the same effect except creating a new array that never gets used:

```
arr.map(console.log);
```

- Misleading
- Poor performance
- Saves 4 letters

Worth 👍

kqwq • Dec 20 '21

Declaring a variable without (var, let, const) will put it in the global scope. Wouldn't recommend it for anything besides code golf lol

```
x = 5;
```

Is equivalent to

```
window.x = 5; // Web JavaScript
global.x = 5; // Node.js
```

Rasmus Schultz • Nov 28 '21

If you want to help, don't teach developers to write unreadable source code - teach them to use a minifier instead.

I realize a minifier won't be able to use all of these patterns, but that's really no excuse either - if you know how to improve it, consider contributing new transforms to a minifier instead. (Probably a few of these would need to be marked as "unsafe" though...)

Vitaliy Markitanov • Nov 27 '21

BS. Why? Nonsense..
Even if you save few bytes in source code - there isn't any reason for it, especially nowadays when you have no limits in storage and data transfer speeds.

Rasmus Schultz • Nov 28 '21

There are reasons to minify, if you ship your script to a million users every minute.

But I agree, there is definitely no good reason to mangle your source code by hand - you might save a few more bytes, but you will likely pay back with bugs. 🙄

IroncladDev

Sixteen-year-old fullstack developer who is addicted to Next.js

LOCATION
Dallas TX USA

EDUCATION
Homeschooled

JOINED
15 Dec 2020

More from IroncladDev

Integrating socket.io with your Next.js application

#nextjs #node #javascript #tutorial

Sending Custom emails with NodeJS from scratch and for no cost whatsoever🗡️

#node #javascript #tutorial #programming

Adding a Next.js Administration Dashboard to your site in seconds

#showdev #javascript #nextjs #react