



By *[David Walsh](#)* on April 16, 2019

Just like every other programming language, JavaScript has dozens of tricks to accomplish both easy and difficult tasks. Some tricks are widely known while others are enough to blow your mind. Let's have a look at seven JavaScript tricks you can start using today!

Get Unique Values of an Array

Getting an [array of unique values](#) is probably easier than you think:

```
var j = [...new Set([1, 2, 3, 3])]
>> [1, 2, 3]
```

I love the mixture of rest expression and Set !

Array and Boolean

Ever need to [filter falsy values](#) (0 , undefined , null , false , etc.) out of an array? You may not have known this trick:

```
myArray
  .map(item => {
    // ...
  })
```

```
// Get rid of bad values  
.filter(Boolean);
```



Just pass `Boolean` and all those falsy value go away!

Create Empty Objects

Sure you can create an object that seems empty with `{}`, but that object still has a `__proto__` and the usual `hasOwnProperty` and other object methods. There is a way, however, to [create a pure "dictionary" object](#):

```
let dict = Object.create(null);  
  
// dict.__proto__ === "undefined"  
// No object properties exist until you add them
```

There are absolutely no keys or methods on that object that you don't put there!

Merge Objects

The need to [merge multiple objects](#) in JavaScript has been around forever, especially as we started creating classes and widgets with options:

```
const person = { name: 'David Walsh', gender: 'Male' };  
const tools = { computer: 'Mac', editor: 'Atom' };  
const attributes = { handsomeness: 'Extreme', hair: 'Brown', eyes: 'Blue' };  
  
const summary = {...person, ...tools, ...attributes};
```

```
/*  
Object {  
  "computer": "Mac",  
  "editor": "Atom",  
  "eyes": "Blue",  
  "gender": "Male",  
  "hair": "Brown",  
  "handsomeness": "Extreme",  
  "name": "David Walsh",  
}  
*/
```



Those three dots made the task so much easier!

Require Function Parameters

Being able to set default values for function arguments was an awesome addition to JavaScript, but check out this trick for [requiring values](#) be passed for a given argument:

```
const isRequired = () => { throw new Error('param is required'); };  
  
const hello = (name = isRequired()) => { console.log(`hello ${name}`) };  
  
// This will throw an error because no name is provided  
hello();  
  
// This will also throw an error  
hello(undefined);  
  
// These are good!
```

```
hello(null);  
hello('David');
```



That's some next level validation and JavaScript usage!

Destructuring Aliases

[Destructuring](#) is a very welcomed addition to JavaScript but sometimes we'd prefer to refer to those properties by another name, so we can take advantage of aliases:

```
const obj = { x: 1 };  
  
// Grabs obj.x as { x }  
const { x } = obj;  
  
// Grabs obj.x as { otherName }  
const { x: otherName } = obj;
```

Useful for avoiding naming conflicts with existing variables!

Get Query String Parameters

For years we wrote gross regular expressions to get query string values but those days are gone -- enter the amazing [URLSearchParams](#) API:

```
// Assuming "?post=1234&action=edit"
```

```
var urlParams = new URLSearchParams(window.location.search);  
  
console.log(urlParams.has('post')); // true  
console.log(urlParams.get('action')); // "edit"  
console.log(urlParams.getAll('action')); // ["edit"]  
console.log(urlParams.toString()); // "?post=1234&action=edit"  
console.log(urlParams.append('active', '1')); // "?post=1234&action=edit&active=1"
```

Much easier than we used to fight with!

JavaScript has changed so much over the years but my favorite part of JavaScript these days is the velocity in language improvements we're seeing. Despite the changing dynamic of JavaScript, we still need to employ a few decent tricks; keep these tricks in your toolbox for when you need them!

What are your favorite JavaScript tricks?



Students and Teachers, save up to 60% on Adobe Creative Cloud.

ADS VIA CARBON

[Learn More](#)

LightFace: Facebook Lightbox for MooTools

One of the web components I've always loved has been Facebook's modal dialog. This "lightbox" isn't like others: no dark overlay, no obnoxious animating to size, and it doesn't try to do "too much." With Facebook's dialog in mind, I've created LightFace: a Facebook lightbox...

Animating CSS3 Transforms with MooTools Fx

I recently posted an awesome (if I may say so myself) CSS3 / MooTools tutorials called [Create a Photo Stack Effect with Pure CSS Animations or MooTools](#). The post presented two ways, a pure CSS method or MooTools-powered class, to duplicate Google+'s elegant photo stack...

Incredible Demos



Using Opacity to Show Focus with MooTools

I'm a huge fan of using subtle effects like link nudging ([jQuery](#), [MooTools](#)) to enhance the user experience and increase the perceived dynamism of my websites. Trust me -- a lot of little things are what take websites to the next level.

Create Spinning Rays with CSS3 Animations & JavaScript

Thomas Fuchs, creator of [script2](#) (scriptaculous' second iteration) and Zepto.js (mobile JavaScript framework), creates outstanding animated elements with JavaScript. He's a legend in his own right, and for good reason: his work has helped to inspire developers everywhere to drop Flash and opt...

Discussion

Marie Thomas



Thanks for sharing these JavaScript tricks. All tricks are awesome but Merge Objects is my favorite one as it helps to make your task much easier and speed up my work. Keep up the great work.

Yan Bloch



Regarding “filter falsy values”, you could use reduce function for single iteration.

Brian De Sousa



Great list! I must be living under a rock because I hadn’t heard of URLSearchParams yet. That’s awesome!

David Hibshman



Convert arrays element types:

```
// number to strings
var strArr = [1,2,3,4,5].map(String);
// ["1","2","3","4","5"]
```

Joel Brighton



Nice! I also particularly like the URLSearchParams tip.

My only (friendly) word of caution in using ‘tricks’ (not necessarily those above) is that obscure coding practices can sometimes cause more confusion than they’re worth. (Of course, what’s obscure can be a matter of perspective). :)

Maximilian Berkmann



And the good ol:

`exp | 0` instead of `Math.floor(exp)` and the `exp | 1 / Math.ceil(exp)` equivalent.

And the ones using bit shifts.

Michał Zalecki



These aren't true equivalents

```
0 | 1 // 1  
Math.ceil(0) // 0
```

Anh Tran



URLSearchParams is amazing. It's been a while struggling with query string and all the regex stuff!

James



Should `isRequired` throw a `ReferenceError` instead?

Jakub Słowik



Is it just me or the `Math.floor()` and `Math.ceil()` is better in comparison to `exp | 0` and `exp | 1`. It's just more readable.

Avraham Hamu



what is the impact if we create an empty object with `{}` ? why does `__proto__` is an issue? is it a performance issue?

Tobi Reif



Shortcuts for updating a variable based on its current value:

```
let speed = 10;  
speed *= 2;  
// → 20
```



```
speed += 10;  
// → 30
```



Jordan Ball



In your tip on filtering falsey values, I don't understand why you included the `map()` method in your illustration. From a pedagogical perspective, it does nothing except cause a reader's attention to focus on a non-issue.

Rich



Favorite ES trick- Quick value “flip-flopping” between two variables without an intermediary variable: (change a to b, change b to a)

Could be done before but was super funky:

```
let a = 'thing';  
let b = 'otherThing';  
  
/* Works because of destructuring */  
[a,b]=[b,a]; // a is now 'otherthing', b is now 'thing'  
  
/* (Old Super-Funky Way) */  
b = [a, a=b][0]; // Works, but barely understandable
```

K L Estes



Not a javascript trick per se, but still interesting:

```
x = x ^ y  
y = x ^ y  
x = x ^ y
```

...and swapped !



Wrap your code in `<pre class="{Language}"></pre>` tags, link to a [GitHub gist](#), [JSFiddle fiddle](#), or [CodePen pen](#) to embed!

☐ Continue this conversation via email

Post Comment!

Use Code Editor