

Join GitHub today


GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Branch: master curriculum / 05-advanced-javascript / 01.advanced-arrays.md

Find file Copy path

 riboher Fix typos and add reduce exercises

ddb1c32 5 days ago

1 contributor

366 lines (273 sloc) 11.3 KB

Raw Blame History   

Arrays in depth

So far we've covered how to **loop** over iterable elements in javascript, such as arrays (**Disclaimer:** objects are [also iterable](#) but let's leave that for another time).

We should know by now how to use a `for` loop and calling the `.forEach` function with an array without much struggle.

These forms of iterating over arrays and manipulate them are mostly ok in the majority of cases, and you could easily do pretty much everything you need with those two types of loops, but there's another way which with you can save a lot of code while keeping your codebase clean and [avoid mutating your arrays](#) (more on this later).

Javascript let us do three basic operations with three different methods that we're going to teach you right now. These three functions have all one thing in common, and that is that they are [Higher order functions](#). Don't worry if you don't understand that concept now, just remember that all three work exactly the same: They are **functions** that accept another function as their parameter (or [callback function](#)). Let's see them in action!

Map

According to [MDN](#) documentation, the map function

"creates a new array with the results of calling a provided function on every element in the calling array".

This is pretty much what we were doing with our arrays, right? The main difference is that `map` already returns a new array (which you must **store** somewhere, of course) with the result of **executing our callback function** once for each one of the elements of our array. So let's see it in action:

```
const arr = [1, 2, 3, 4, 5, 6, 7]
const multiplyBy2 = (n) => n * 2

const arrTimes2 = arr.map(num => multiplyBy2(num))

console.log(arrTimes2) // [2, 4, 6, 8, 10, 12, 14]
```

Ok, so let's dive first into that code from above:

1. First, we've **declared** an array with some numbers, from 1 to 7.
- 2.