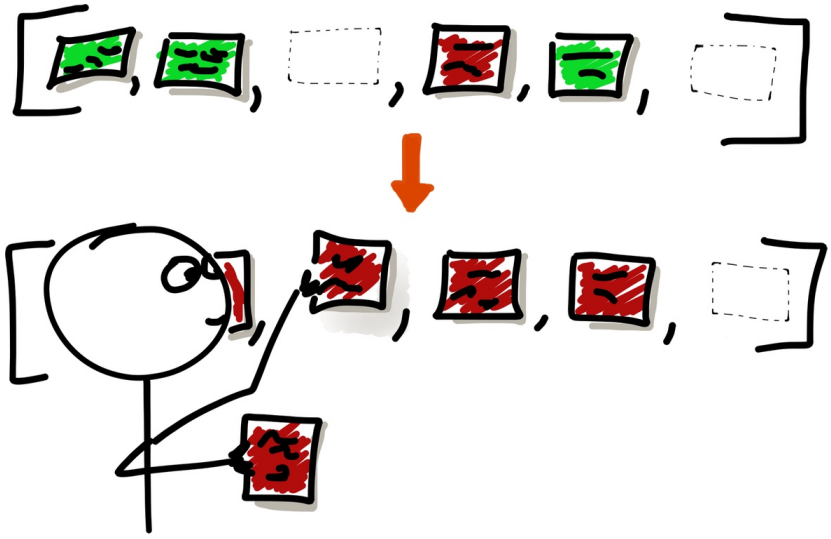
How To Get The Difference Between Two Arrays, As A Distinct List, With JavaScript ES6

September 23, 2015 By derickbailey

I recently found myself in a situation where I had a non-distinct list of things that were invalid (it contained duplicates), which came from a larger list of things. What I needed was a distinct list of things that were valid. But I only had the full list and the list of invalid things.



My thought process took me down the pat of saying "I wish I had a Set and a simple way to get the difference between two lists". A short search later, I was looking at Dr Axel Rauschmayer's Exploring ES6 book (highly recommended reading!), with my answer right there on the page!

ES6 Sets

If you're not familiar with the idea of a set, it's a math concept that says there is a list of distinct things. There can be no duplicates in a set. Integers are a good example of a set: *1, 2, 3, 4, 5*

In this example, there is a list of integers that count from 1 to 5 with no duplication. Each item is distinct. You won't find "1" in the list from 1 to 5 more than once.

This is a set – and I wanted these semantics in a JavaScript list of things. Fortunately, ES6 provides a Set object for just that purpose.

Array-Like, Distinct List Of Items

Think of a Set as an array-like (iterable) object, but with the semantics and behavior of ensuring a distinct list of items. In other words, it is not possible to have duplicates.

There isn't a "set literal" syntax, like there is for an array (the *[square, bracket, syntax]*) but the API for a set is fairly simple.

```
// initial set created from array
var mySet = new Set([1, 2, 3, 2, 4, 1, 3, 5]);

// add more items
mySet.add(1);
mySet.add(3);

// see what it holds
console.log(mySet); // => Set { 1, 2, 3, 4, 5 }
```

1.js hosted with ♥ by GitHub view raw

In this example, I'm creating a set and adding numbers to it – including some duplicates. The resulting list of items does not contain any duplicates.

This solves my first problem – getting a distinct list of the objects that I have, which are invalid. Now I need to solve the second half of my problem.

ES6 Array Filter To Get The Difference

One of the new features of ES6 Array's is the filter method. This method lets you decide which items to exclude from the original array, returning a new array with only the leftover items in it.

In my case, I needed to filter the set of invalid objects out of the original array. To do this, I used the .has method on my "invalid" set. This method is used in the "filter" callback to determine which objects should be removed.

```
var completeList = [/* ... filled in elsewhere */]

```

```
3  var invalidList = [/* ... also filled in elsewhere */]
4
5  // filter the items from the invalid list, out of the cor
6  var validList = completedList.filter((item) => {
7    return !invalidList.has(item);
8  })
9
10 // get a Set of the distinct, valid items
11 var validItems = new Set(validList);
```

2.js hosted with ♥ by GitHub view raw

Note that the filter method keeps everything that returns "true" from the callback. Since I want to remove items that are in my other list, I need to invert the boolean result of the has method. That way my code says

> *If I have this object in the invalid list, return false so the object is removed.*
>
> *If I do not have this object, return true so that the object is kept.*

Having filtered the array down to a "valid" object list, I then converted it into a Set. This probably isn't necessary, but made me a little more comfortable knowing that I would have a distinct list of objects with no duplicates.

At this point, I have both my list of valid and invalid objects!

It's Like A Left Outer Join… For JavaScript!

Ok, that probably sounds dumb… but that's how I was thinking about this problem. I was using a set-based, SQL mindset to think through the problem and found a way to perform the operations with my already in-memory arrays.

If you have a database, use a database with a query language that knows how to deal with sets, like this. If you don't have the database query available, you may be able to take advantage of that mindset, still. JavaScript does not natively work with set based operations in the way SQL does – but it does provide a functional, iterative approach to solving many of the same problems.

(… and yes, there are a lot of potential improvements for this code – including the use of objects that store their own valid/invalid state. These objects don't keep track of that information in their attributes. They just get put in to an invalid bucket when they are invalid, and taken out when they are valid.)

RELATED POST

10 Myths About Docker That Stop Developers Cold

Docker Recipes Update: Speed Up npm install In Mou...

Update and a Bonus Recipe for the Docker Recipes e...

A Sneak Peak at Docker Recipes for Node.js Develop...

Docker Recipes for Node.js: Pre-sale Dates & ...

Filed Under: Arrays, ES6, JavaScript, Sets, SQL

About derickbailey

Derick Bailey is a developer, entrepreneur, author, speaker and technology leader in central Texas (north of Austin). He's been a professional developer since the late 90's, and has been writing code since the late 80's. In his spare time, he gets called a spamming marketer by people on Twitter, and blurts out all of the stupid / funny things he's ever done in his career on his email newsletter.

DERICK BAILEY AROUND THE WEB

Twitter: @derickbailey

Google+: DerickBailey

Screencasts: WatchMeCode.net

eBook: Building Backbone Plugins