A Drip of JavaScript

Building Up Arrays with Array#concat

Originally published in the A Drip of JavaScript newsletter.

Working with arrays is the bread and butter of being a JavaScript developer. And among the most common tasks is building up a new array out of smaller ones. Let's take a look at one way to do this.

Suppose that you are implementing a registry that holds a list of metahumans. We want to be able to add to the list by giving it a list of new metahumans. A quick and simple way to do it might look like this.

```
// Metahuman Registry
var mhr = [];

var heroes = ["Captain Marvel", "Aquaman"];
var villains = ["Black Adam", "Ocean Master"];

mhr = mhr.concat(heroes);

// Outputs: ["Captain Marvel", "Aquaman"]
console.log(mhr);

mhr = mhr.concat(villains);

// Outputs: [
//     "Captain Marvel",
//     "Aquaman",
```

```
//     "Black Adam",
//     "Ocean Master"
// ]
console.log(mhr);
```

As you can see, when we pass in an array `concat` creates a new array which consists of the elements of both `mhr` and whatever we passed in, maintaining the order of the elements. This is useful on its own, but it turns out that `concat` can actually accept multiple arguments. So we could rewrite the example above as follows:

```
// Metahuman Registry
var mhr = [];

var heroes = ["Captain Marvel", "Aquaman"];
var villains = ["Black Adam", "Ocean Master"];

mhr = mhr.concat(heroes, villains);

// Outputs: [
//     "Captain Marvel",
//     "Aquaman",
//     "Black Adam",
//     "Ocean Master"
// ]
console.log(mhr);
```

But `concat` can handle more than just arrays. If `concat` is given a non-array value, it will just drop that value into the new array at the appropriate location.

```
// Metahuman Registry
var mhr = [];

var heroes = ["Captain Marvel", "Aquaman"];
var villains = ["Black Adam", "Ocean Master"];
```

```
mhr = mhr.concat(heroes, villains, "Death");

// Outputs: [
//     "Captain Marvel",
//     "Aquaman",
//     "Black Adam",
//     "Ocean Master",
//     "Death"
// ]
console.log(mhr);
```

So far I've been glossing over an important detail. The `concat` method doesn't modify the array that it is called on. Instead it creates a brand new array. That's why in the examples above I've been assigning the result to `mhr`.

The fact that it is a new array can be quite useful. Suppose that you want to be able to check whether `mhr` has been modified since the last time you checked on it. You could do something like this:

```
// Metahuman Registry
var mhr = [];

var original = mhr;

// Outputs: true
console.log(mhr === original);

var heroes = ["Captain Marvel", "Aquaman"];
mhr = mhr.concat(heroes);

// Outputs: false
console.log(mhr === original);
```

I hope this look at `concat` gave you a deeper understanding of how it works.
