Lesson Weekend

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Comparing and Cloning Arrays

Text Cheat sheet

Unlike other data types we've seen so far, we can't compare two arrays by using the `===` comparison operator. In JavaScript, two primitive data type values can be equal. For example, "abc" is the same as any other "abc". But arrays are different — **no two arrays are the same, even if they have the exact same elements inside**.

Comparing Arrays

If you need to compare the *contents* of two arrays, one way to accomplish this is to transform the arrays into strings using the `toString()` method on each and comparing the return values. Be careful, though, because there are edge cases where this doesn't work correctly. An **edge case** in computer programming is a possible outcome of an operation that leads to unexpected or inconsistent results. Let's take a look at an example where we can effectively compare two arrays — and then let's tweak one of the arrays to make it a gotcha:

```
> const a = [1,2,3];
> const b = [1,2,3];
> a === b;
false
> a.toString();
"1,2,3"
> a.toString() === b.toString();
true
```

Great! This works. But what if one of the arrays includes strings?

```
> const c = [1,2,"3"];
> const d = [1,2,3];
> c === d;
false
> c.toString();
"1,2,3"
> d.toString();
"1,2,3"
> c.toString() === d.toString();
true
```

According to the rules of strict equality (`===`), strings and numbers are different — as well they should be. That means that the string `"3"` inside array `c` should not be considered the same as the number `3` inside array `d` . However, the `Array.prototype.toString()` method does not account for differing data types within each array: they are all turned into strings! So this is an example of when comparing the equality of two arrays by turning them into strings with `Array.prototype.toString()` does not work effectively.

As with many things in computer programming, we've found a fairly effective solution that doesn't work all of the time. As a developer, a big part of your job is to find things that work — but also to

understand the little ways and circumstances where things don't work.

Cloning Arrays

Here is another scenario where arrays operate differently than we might expect.

Suppose we have an array that we want to keep in its original form. We also need to manipulate this array in some way. We might have one variable hold the original array and another variable with a copy of the same array that we can change.

Try this in the console:

```
> const originalArray = [1,2,3,4,5];
> originalArray;
[1, 2, 3, 4, 5]
> let cloneArray = originalArray;
> cloneArray;
[1, 2, 3, 4, 5]
> cloneArray.push(6);
6
> cloneArray;
[1, 2, 3, 4, 5, 6]
```

What does `originalArray` look like now? We haven't touched it, so we may assume it would retain its original value of `[1,2,3,4,5]`. But if we check it again in the console, we'll see the following:

```
> originalArray;
[1, 2, 3, 4, 5, 6]
```

In JavaScript, the array is stored in memory. The variables `originalArray` and `cloneArray` are called **pointers**. A pointer references an object in memory but is *not* the object itself. Therefore, as the array is changed, the new value is reflected in all of the variables that point to it.

The way to clone an array *without* simply making another pointer is to create a new variable set to a new array:

```
const cloneArray = originalArray.slice();
```

Note that we use the `Array.prototype.slice()` here in a slightly different way than we learned in the last lesson — without arguments. This just "slices" the whole array so we can save it in a new variable. This ensures that the new array really is an entirely separate entity instead of simply a pointer to the original array.

Additional Resources

For more details on how the `Array.prototype.slice()` method works, check out MDN's JavaScript documentation (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice).

Previous (/introduction-to-programming/arrays-and-looping/array-methods)
Next (/introduction-to-programming/arrays-and-looping/document-query-methods-that-return-collections)

(http://www.epicodus.com)