A Drip of JavaScript

Detecting Arrays (and other subtypes) vs. Objects in JavaScript

Originally published in the A Drip of JavaScript newsletter.

When writing JavaScript, it is often necessary to detect whether a certain variable is an array or an ordinary object so that you can perform a different set of actions. For instance, consider a function that can be called with an object representing a marathon, or an array of objects representing multiple marathons:

```
function getListOfMarathonNames (marathons) {
    if (marathons instanceof Object) {
        // Return an array containing the name of the only
        // marathon given.
        return [marathons.name];
    } else if (marathons instanceof Array) {
        // Return an array containing all marathon names.
        return marathons.map(function(race) {
            return race.name;
        });
    }
}
```

At a glance, this seems like perfectly reasonable code. Unfortunately, it is hiding a major bug. Let's test it out to see how it behaves.

```
var londonMarathon = {
    name: "London Marathon",
    date: "April 13, 2014"
};

console.log(getListOfMarathonNames(londonMarathon));
// -> ["London Marathon"]
```

So far so good.

```
var moreMarathons = [
    {
        name: "New York City Marathon",
        date: "November 2, 2014"
    },
    {
        name: "Chicago Marathon",
        date: "October 12, 2014"
    }
];

console.log(getListOfMarathonNames(moreMarathons));
// -> [undefined]
```

What's going on here? The problem is that our function's array detection logic is reversed. Because in JavaScript arrays are just a special type of object, it is impossible for our `else if` to ever be triggered. And since the array we passed in doesn't have a `name` property, we end up returning an array containing only an `undefined` element.

In order to correct this logic we need to work in the other direction, first checking whether the input is of the `Array` subtype before proceeding to check whether it is part of the broader `Object` type.

```
function getListOfMarathonNames (marathons) {
    if (marathons instanceof Array) {
        // Return an array containing all marathon names.
        return marathons.map(function(race) {
            return race.name;
        });
    } else if (marathons instanceof Object) {
        // Return an array containing the name of the only
        // marathon given.
        return [marathons.name];
    }
}

console.log(getListOfMarathonNames(londonMarathon));
// -> ["London Marathon"]

console.log(getListOfMarathonNames(moreMarathons));
// -> ["New York City Marathon", "Chicago Marathon"]
```

And now we have everything working as it should.

Of course, this type of error can also occur when trying to detect other types of objects as well, like `Date`, `RegExp`, etc. The general rule is to check for the subtype first, only handling `Object` afterwards.

It's also worth noting that this isn't a comprehensive test for whether something is an array. While it will work in most cases, if you are working with iframes or objects from another "domain," `instanceof` isn't all that useful, and you'll probably want to use a utility library to handle the checking for you.

Thanks for reading!

Josh Clanton
