A Drip of JavaScript

Determining if a String Contains a Substring in JavaScript

Originally published in the A Drip of JavaScript newsletter.

One of the most basic tasks in any programming language is determining whether a string contains a given substring. Unfortunately, JavaScript's built-in tools for doing so leave quite a bit to be desired. First of all, let's take a look at using `String.prototype`'s `indexOf` method.

```
var philosophers = "Aquinas, Maimonedes, and Avicenna";
var me = "Joshua";

function printPhilosopherStatus (person) {
    if (philosophers.indexOf(person) >= 0) {
        console.log(person + " is a philosopher.");
    } else {
        console.log(person + " is NOT a philosopher.");
    }
}

// Outputs: "Joshua is NOT a philosopher."
printPhilosopherStatus(me);
```

While `indexOf` is often recommended as a simple way to test for the presence of a substring, that's not really its purpose. Its job is to return the index at which a given substring is found. In the event that no match is found, it will return `-1`. That means that we can use it, but the clarity of the code suffers. Ideally, what we're looking for is

a method with a name that matches our intention (determining if x contains y), and returns a simple `true` or `false`.

Looking through the documentation for `String.prototype`, the `search` method looks promising due to its name. Unfortunately, with the exception of matching on a regular expression rather than a string, the behavior is identical to `indexOf`.

However, that does point us toward something else useful. `RegExp.prototype` has a `test` method which returns a boolean. Let's try it out.

```
var philosophers = "Aquinas, Maimonedes, and Avicenna";
var me = "Joshua";

function printPhilosopherStatus (person) {
    var personRegExp = new RegExp(person);
    if (personRegExp.test(philosophers)) {
        console.log(person + " is a philosopher.");
    } else {
        console.log(person + " is NOT a philosopher.");
    }
}

// Outputs: "Joshua is NOT a philosopher."
printPhilosopherStatus(me);
```

This is a bit better because the method itself returns `true` or `false`. The method name also communicates intent more clearly than `indexOf`.

Unfortunately, if we are trying to match a string which uses characters like `?` or `.`, we have a problem. Because they have special meanings in regular expressions, we have to deal with escaping them. That means this isn't a very good general purpose solution. In addition, the code could still use some improvement in clearly communicating its intent.

Finally we come to `String.prototype`'s `contains` method.

```
var philosophers = "Aquinas, Maimonedes, and Avicenna";
var me = "Joshua";

function printPhilosopherStatus (person) {
    if (philosophers.contains(person)) {
        console.log(person + " is a philosopher.");
    } else {
        console.log(person + " is NOT a philosopher.");
    }
}

// Outputs: "Joshua is NOT a philosopher."
printPhilosopherStatus(me);
```

This has all the features that we've been looking for. It returns a boolean value, and the method name clearly conveys the intent of our code.

Unfortunately, there is a problem. The `contains` method is a proposal for the next version of JavaScript (ECMAScript 6) and has only been implemented in FireFox 19+ so far.

If you'd like to use something similar to `contains`, for now your best bet is to use a third-party library like String.js, a "prolly-fill" like ES6 Shim, or wrap `indexOf` in your own custom utility function, like so:

```
function aContainsB (a, b) {
    return a.indexOf(b) >= 0;
}

var philosophers = "Aquinas, Maimonedes, and Avicenna";
var me = "Joshua";

function printPhilosopherStatus (person) {
    if (aContainsB(philosophers, person)) {
        console.log(person + " is a philosopher.");
    } else {
```

```
        console.log(person + " is NOT a philosopher.");
    }
}

// Outputs: "Joshua is NOT a philosopher."
printPhilosopherStatus(me);
```

And that is an overview of some the ways you can determine if a string contains substrings in JavaScript.

Thanks for reading!

Joshua Clanton
