A Drip of JavaScript

The Problems with for...in and JavaScript Arrays

Originally published in the A Drip of JavaScript newsletter.

We've talked in the past about different ways of iterating over arrays. But in this drip we'll take a look at one way **not** to do it.

JavaScript's `for` ... `in` loop iterates over the enumerable properties of an object like so:

```
var tMinus = {
    two: "Two",
    one: "One",
    zero: "Blast off!"
};

var countdown = "";

for (var step in tMinus) {
    countdown += tMinus[step] + "\n";
}

console.log(countdown);
// => "Two
//    One
//    Blast Off!
//    "
```

Because `for ... in` operates on any JavaScript object, it is also possible to use it with arrays. For example:

```
var tMinus = [
    "Two",
    "One",
    "Blast off!"
];

var countdown = "";

for (var step in tMinus) {
    countdown += tMinus[step] + "\n";
}

console.log(countdown);
// => "Two
//    One
//    Blast Off!
//    "
```

However, there are three problems with using this approach on arrays. First, the `for ... in` also iterates over an object's prototype properties if those properties are enumerable. For example:

```
Array.prototype.voice = "James Earl Jones";

var tMinus = [
    "Two",
    "One",
    "Blast off!"
];

var countdown = "";

for (var step in tMinus) {
    countdown += tMinus[step] + "\n";
```

```
}

console.log(countdown);
// => "Two
//    One
//    Blast Off!
//    James Earl Jones
//    "
```

That can be solved by using `hasOwnProperty` to exclude prototype properties.

Array.prototype.voice = "James Earl Jones";

```
Array.prototype.voice = "James Earl Jones";

var tMinus = [
    "Two",
    "One",
    "Blast off!"
];

var countdown = "";

for (var step in tMinus) {
    if (tMinus.hasOwnProperty(step)) {
        countdown += tMinus[step] + "\n";
    }
}

console.log(countdown);
// => "Two
//    One
//    Blast Off!
//    "
```

Second, according to the specification `for` ... `in` loops may iterate over an object's values in an **arbitrary order**.

That's not really a problem for an ordinary object whose values are inherently unordered anyway. But you probably don't want your JavaScript engine handing back array values in a random order, because you could get unexpected results like this:

```
console.log(countdown);
// => "Blast Off!
//    One
//    Two
//    "
```

The third problem is that `for ... in` iterates over **all enumerable properties**, not just the array's elements. As we've discussed before, it is possible to store additional properties on an array. This can also lead to unexpected results.

```
var tMinus = [
    "Two",
    "One",
    "Blast off!"
];

tMinus.announcer = "Morgan Freeman";

var countdown = "";

for (var step in tMinus) {
    if (tMinus.hasOwnProperty(step)) {
        countdown += tMinus[step] + "\n";
    }
}

console.log(countdown);
// => "Two
//    One
//    Blast Off!
//    Morgan Freeman
//    "
```

Because of these problems you will almost never want iterate over arrays with `for ... in` loops. Instead, use an ordinary `for` loop, or one of the built-in array iteration methods like `forEach` or `map`.

Now you know one more quirk to avoid as you write safe and clean JavaScript.

Thanks for reading!

Josh Clanton
