A Drip of JavaScript

Immutable Objects with `Object.freeze`

Originally published in the [A Drip of JavaScript newsletter](#).

One of the more common techniques in JavaScript is the use of an object to hold configuration values. The object might be accessed as a global or passed around as an argument. For example:

```
var artist = {
    name: "Johnny Cash",
    latestAlbum: "American V"
};

function announce (artist) {
    if (artist.name == "Johnny Cash") {
        console.log("The Man in Black");
    } else {
        console.log(artist.name);
    }
}

// Outputs: "The Man in Black"
announce(artist);

// Outputs: {
//     name: "Johnny Cash",
//     latestAlbum: "American V"
// }
console.log(artist);
```

But in either sort of situation, there is a problem. Functions that have access to the configuration object can modify the object, whether intentionally or accidentally. Suppose that you had a coworker modify the `announce` function above to highlight Elvis rather than Cash, but they mistyped the comparison.

```
var artist = {
    name: "Johnny Cash",
    latestAlbum: "American V"
};

function announce (artist) {
    // Whoops! Assigning the name rather than testing equality!
    if (artist.name = "Elvis Presley") {
        console.log("The King");
    } else {
        console.log(artist.name);
    }
}

// Outputs: "The King"
announce(artist);

// Outputs: {
//     name: "Elvis Presley",
//     latestAlbum: "American V"
// }
console.log(artist);
```

Hmm. I'm pretty sure that Elvis didn't record American V.

Some of you may be thinking that this what something like JSHint is for, and you'd be right. But as part of a defence in depth strategy, it would be awfully nice to make sure that our configuration objects don't get changed around after they're created. Fortunately, JavaScript gives us a way to do exactly that.

```
var artist = {
    name: "Johnny Cash",
    latestAlbum: "American V"
};

Object.freeze(artist);

function announce (artist) {
    // Whoops! Assigning the name rather than testing equality!
    if (artist.name = "Elvis Presley") {
        console.log("The King");
    } else {
        console.log(artist.name);
    }
}

// Outputs: "The King"
announce(artist);

// Outputs: {
//     name: "Johnny Cash",
//     latestAlbum: "American V"
// }
console.log(artist);
```

The `Object.freeze` method takes an object and renders it effectively immutable. Its existing properties may not be modified and new properties may not be added. In the example above this means that even though the logical error is still there, our `artist` object remains safe from modification and available for later use.

While in normal mode attempts to modify the object will fail silently, if you use strict mode, an error will be thrown.

```
var artist = {
    name: "Johnny Cash",
    latestAlbum: "American V"
};
```

```
Object.freeze(artist);

(function() {
    "use strict";

    // TypeError: Can't add property firstAlbum, object is not extensib
    artist.firstAlbum = "A Hard Days Night";
})();
```

Of course, we don't want to throw errors all over the place, so JavaScript also gives us a method to detect whether an object is frozen. Appropriately, it is named `Object.isFrozen`.

```
var artist = {
    name: "Johnny Cash",
    latestAlbum: "American V"
};

Object.freeze(artist);

// Outputs: "Frozen Artist!"
if (Object.isFrozen(artist)) {
    console.log("Frozen artist!");
}
```

`Object.freeze` is part of the ECMAScript 5 specification, which means it isn't available in older browsers like IE8 and below. If you need to support those browsers, then you'll need to either avoid it or use feature detection to ensure you use it only in supporting browsers.

I hope this quick introduction to `Object.freeze` helps you write more robust code.
