A Drip of JavaScript

Preventing Object Extensions in JavaScript

Originally published in the A Drip of JavaScript newsletter.

In the last two drips, we've discussed safeguarding your objects with `Object.seal` and `Object.freeze`. In this issue we look at our final object protection mechanism: `Object.preventExtensions`.

Suppose that you have an object recording the populations of the continents. You probably don't want new continents added willy-nilly.

```
var continentPopulations = {
    africa: 1100000000,
    antarctica: 5000,
    asia: 4300000000,
    australia: 36000000,
    europe: 739000000,
    northAmerica: 529000000,
    southAmerica: 387000000
};

Object.preventExtensions(continentPopulations);

continentPopulations.atlantis = 50000;

// Outputs: undefined
console.log(continentPopulations.atlantis);
```

As you can see, the effect of `Object.preventExtensions` is to prevent the adding of new properties to an object. If you are using strict mode it will throw an error, but in normal mode it will fail silently.

It is important to remember that `Object.preventExtensions` does not prevent manipulating the values of existing properties.

```
// The antarctic population drops during winter
continentPopulations.antarctica = 500;

// Outputs: 500
console.log(continentPopulations.antarctica);
```

Nor does it prevent deleting existing properties.

```
// In the year 2248 Antarctica was destroyed
// by spacefaring dinosaurs from another dimension
delete continentPopulations.antarctica;

// Outputs: undefined
console.log(continentPopulations.antarctica);
```

To detect whether an object will accept new properties, use the corresponding `Object.isExtensible` method like so:

```
// Outputs: false
console.log(Object.isExtensible(continentPopulations));
```

`Object.preventExtensions` and `Object.isExtensible` are both part of the ECMAScript 5 specification, so older browsers like IE8 don't support them. And due to the limitations of previous versions of JavaScript, these features cannot be polyfilled.

That's it for our series on JavaScript's built-in object protection methods.