A Drip of JavaScript

Avoiding Problems with Decimal Math in JavaScript

Originally published in the A Drip of JavaScript newsletter.

One of the more unintuitive aspects of JavaScript, particularly for new developers, is the fact that decimal math doesn't always work as you'd expect it to. Suppose that Worf has exactly $600.90 of 21st century American dollars to trade for a wine of excellent vintage.

```
var worfsMoney = 600.90;
var bloodWinePrice = 200.30;
var worfsTotal = bloodWinePrice * 3;

// Outputs: false
console.log(worfsMoney >= worfsTotal);

// Outputs: 600.9000000000001
console.log(worfsTotal);
```

As you can see, simply checking whether Worf has enough money fails because his total doesn't come to exactly $600.90. But why is that?

In JavaScript all numbers are IEEE 754 floating point numbers. Due to the binary nature of their encoding, some decimal numbers cannot be represented with perfect accuracy. This is analagous to how the fraction 1/3 cannot be accurately represented with a decimal number with a finite number of digits. Once you hit the limit of your storage you'll need to round the last digit up or down.

Your first thought might be to try rounding to the second decimal place. Unfortunately, JavaScript's built-in rounding functions only round to the nearest integer.

Your second thought might be to do something like this:

```
// Outputs: true
console.log(worfsMoney.toFixed(2) >= worfsTotal.toFixed(2));
```

But even though we get the correct result, using `toFixed` means that we are just comparing formatted strings rather than actual numbers. And what we really want is a way to get an accurate numeric representation of our numbers.

Fortunately there is an easy way to do that in JavaScript. Instead of representing our money's value as decimal numbers based on dollars, we can just use whole integers of cents.

```
var worfsMoney = 60090;
var bloodWinePrice = 20030;
var worfsTotal = bloodWinePrice * 3;

// Outputs: true
console.log(worfsMoney >= worfsTotal);

// Outputs: 60090
console.log(worfsTotal);
```

Because integers can be represented perfectly accurately, just shifting our perspective to treat cents as the basic unit of currency means that we can often avoid the problems of comparisons in floating point arithmetic.

Of course, this doesn't solve all problems. Division of integers and multiplication by decimals may still result in inexact values, but basic integer addition, subtraction, and multiplication will be accurate. (At least until you hit JavaScript's upper limit for numbers.)

Thanks for reading!

Josh Clanton
