Title: Arguments Passed by Value vs. Arguments Passed by Reference

Answer:

One important property of primitives in JS is that when they are passed as arguments to a method, they are *copied* ("passed by value"). So for example:

Heads up: This is *not* well-formed JavaScript. We're using it to prove a point.

```
 let threatLevel = 1;

function inspireFear(threatLevel){
  threatLevel += 100;
}

inspireFear(threatLevel);
console.log(threatLevel); // Whoops! It's still 1!
```

JavaScript

The threatLevel inside inspireFear() is a *new* number, initialized to the same *value* as the threatLevel outside of inspireFear(). Giving these *different* variables the same name might cause confusion here. If we change the two variables to have different names we get the exact same behavior:

```
 let threatLevel = 1;

function inspireFear(theThreatLevel){
  theThreatLevel += 100;
}

inspireFear(threatLevel);
console.log(threatLevel); // Whoops! It's still 1!
```

JavaScript

In contrast, when a method takes an object, it actually takes a *reference* to *that very object*. So changes you make to the object in the method persist after the method is done running. This is sometimes called a side effect.

```
 const scaryThings = ['spiders', 'Cruella de Vil'];

function inspireFear(scaryThings){
  scaryThings.push('Snakes');
  scaryThings.push('no internet');
  scaryThings.push('low battery and no charger');
}

inspireFear(scaryThings);
console.log(scaryThings);
// ['spiders', 'Cruella de Vil', 'Snakes', 'no internet', 'low battery and n
o charger']
```

Tags: functions / methods