What is ‘bind()’

Syntax:

```
func.bind(thisArg[, arg1[, arg2[, ...]]])
```

What is a Function?

- Usually has a name, optional arguments, and a sequence of statements
- It is a “first-class” object in Javascript
- It is attached to a declaration context (aka ‘this’)

Is ‘this’ broken?

```
console.log('BAD - This will not work');
thing = {
  special: function() {
    console.log('special');
  },
  getAsync: function(callback) {
    callback('async data');
  },
  print: function() {
    this.getAsync(function(data) {
      try {
        console.log(data);
        this.special();
      } catch (e) {
        console.log(`${e.name}: ${e.message}`);
      }
    });
  }
}
thing.print();
```

Output:

```
[nodemon] starting `node bind-this.js`
BAD - This will not work
async data
TypeError: Cannot read property 'special' of undefined
```

How do we fix ‘this’?

```
console.log('GOOD - Manually assign "this" to var "self"');
thing = {
  special: function() {
    console.log('special');
  },
  getAsync: function(callback) {
    callback('async data');
  },
  print: function() {
    var self = this;
    this.getAsync(function(data) {
      try {
        console.log(data);
        self.special();
      } catch (e) {
        console.log(`${e.name}: ${e.message}`);
      }
    });
  }
}
thing.print();
```

Output:

```
GOOD - Manually assign "this" to var "self"
async data
special
```

Best to automatically bind() ‘this’...

```
console.log('BEST - Use function bind()');
thing = {
  special: function() {
    console.log('special');
  },
  getAsync: function(callback) {
    callback('async data');
  },
  print: function() {
    this.getAsync(function(data) {
      try {
        console.log(data);
        this.special();
      } catch (e) {
        console.log(`${e.name}: ${e.message}`);
      }
    }.bind(this));
  }
}
thing.print();
```

Output:

```
BEST - Use function bind()
async data
special
[nodemon] clean exit - waiting for changes before restart
```

Partially Applied Functions

```
function print(a, b) {
  console.log(a, b);
}

// We can bind one argument at a time
var printA = print.bind(null, 'Ahh');
printA();

var printAandB = printA.bind(null, 'Boo');
printAandB();

// We can chain bound functions
var oneLiner = print
  .bind(null, 'Aah')
  .bind(null, 'Boo');
oneLiner();
```

Output: