METHODS & *THIS*

Objects Pt. 2

THE SECRET LIFE OF OBJECTS

- We've seen the basics of object literals, but there is so much more to talk about with OBJECTS!

GOALS

- Add methods to objects
- Use new object shorthand syntax
- Use computed properties
- Explain how the %$@# *this* works.

SHORTHAND PROPERTIES

```
const reviews = [4.5, 5.0, 3.44, 2.8, 3.5, 4.0, 3.5];
const max = Math.max(...reviews);
const min = Math.min(...reviews);
const sum = reviews.reduce((sum, r) => sum + r);
const avg = sum / reviews.length;

const stats = {min, max, sum, avg}
stats; //{min: 2.8, max: 5, sum: 26.74, avg: 3.82}
```

COMPUTED PROPERTIES

```
const user = 'Jools';

const userRoles = {
  [user]: 'Admin'
}
userRoles; //{Jools: "Admin"}
```

We can use a variable as a key name in an object literal property!

METHODS

```
const math = {
    multiply : function(x, y) {
        return x * y;
    },
    divide   : function(x, y) {
        return x / y;
    },
    square   : function(x) {
        return x * x;
    }
};
```

We can add functions as properties on objects.

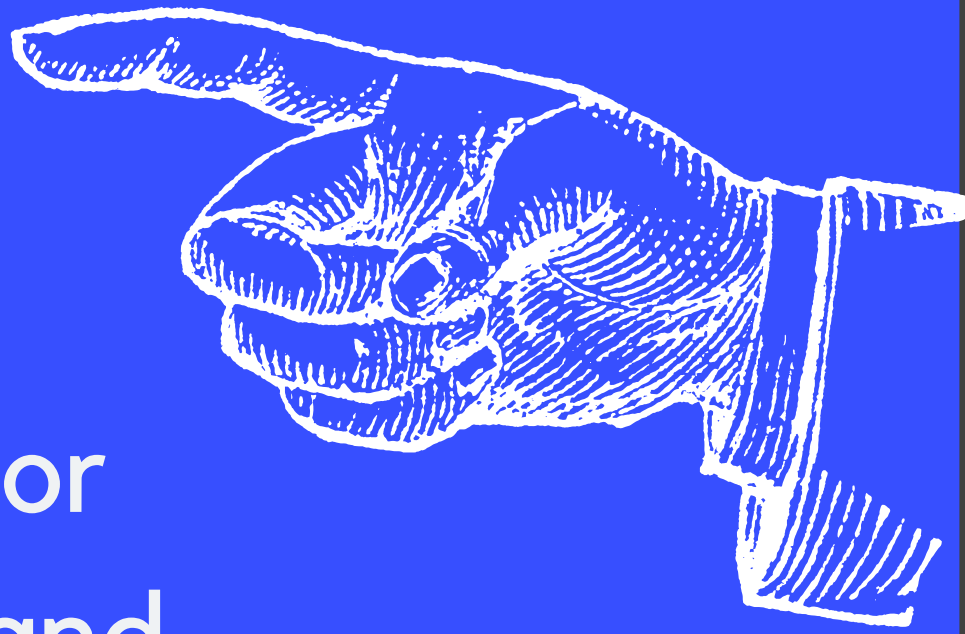
We call them methods!

SHORTHAND

```
const math = {
  blah: 'Hi!',
  add(x, y) {
    return x + y;
  },
  multiply(x, y) {
    return x * y;
  }
}
math.add(50, 60) //110
```

We do this so often that there's a new shorthand way of adding methods.

THIS



The keyword *this* can be a major point of confusion and misery and hardship and general suffering in the life of a new JS developer.

COURAGE

"One of the most courageous decisions you'll ever make is to try and learn the JavaScript keyword *this*."

fingers

THIS IN METHODS

Use the keyword ***this*** to access other properties on the same object

```
const person = {
  first: 'Robert',
  last: 'Herjavec',
  fullName() {
    return `${this.first} ${this.last}`
  }
}
person.fullName(); //"Robert Herjavec"
person.last = "Plant";
person.fullName(); //"Robert Plant"
```

The value of ***this*** depends on the invocation context of the function it is used in.

SAME FUNCTION

```
const person = {
  first: 'Robert',
  last: 'Herjavec',
  fullName() {
    return `${this.first} ${this.last}`
  }
}
```

```
person.fullName();
//"Robert Herjavec"
```

DIFFERENT RESULT???

```
const func = person.fullName;
func()
//"undefined undefined"
```

The value of ***this*** depends on the **invocation context** the function it is used in.

(ALMOST) EVERYTHING IS AN OBJECT (KIND OF)

"HI"

Primitive

Object Wrapper