Object Enhancements

Object Shorthand

ES2015 provides quite a few enhancements for JS objects!

When the keys are the same name as the variable values, (this happens a lot), you don't have to repeat yourself.

```
let firstName = "Mary";
let lastName = "Malarky";

// ES5 (Oldschool)
let instructor = {
    firstName: firstName,
    lastName: lastName
}
```

```
let firstName = "Mary";
let lastName = "Malarky";

// ES6
let instructor = {
    firstName,
    lastName
}
```

Object Methods

A nice shorthand when a key in an object represents a function.

```
// ES5
let instructor = {
  sayHello: function () {
    return "Hello!";
  }
}
```

```
// ES2015 - do NOT use arrow functions here
let instructor = {
  sayHello() {
    return "Hello!";
  }
}
```

Computed Property Names

ES2015 allows us to create an object with a key that JavaScript can compute at definition.

Here's what we mean by that!

```
// ES5
let firstName = "Mary";
let instructor = {};
instructor[firstName] = "That's me!";

instructor.Mary; // "That's me!"
```

```
// ES2015
let firstName = "Mary";
let instructor = {
    [firstName]: "That's me!"
}

instructor.Mary; // "That's me!"
```

Current usage

- These new shorthand methods are everywhere!
- Object shorthand and methods allow for writing less code

- Computed property names are everywhere in modern web frameworks.

Computed property names in the wild

- This appears when you work with multiple inputs or DOM elements and you want to change the value in an object based on a specific interaction,
- It's impossible to know upfront what key you are changing in the object without hardcoding the key, so instead we can use the ***event*** object for a browser interaction.

```
function changeValueInObj(obj, event){
  return {
    ...obj,
    [event.target.name]: [event.target.value]
  }
}
```