Lesson Introduction

Let's talk a little bit about some use cases for closures. You're probably already familiar with how you associate methods and properties with a function and, just to give you a recap on how that works, let's imagine that we have a JavaScript program and we're going to create a variable called `tracking`:

```
var tracking = {};
```

Now create a function:

```
function trackActions(item, choice) {
  if (choice) {
    tracking[item] = choice;
  }
  return tracking[item];
}
```

Now use it:

```
trackActions("red", "click");
console.log(trackActions("red")); //"click"
console.log(tracking.red); //"click"
```

Closures for Private Storage

So this function is modifying the variable `tracking` that exists outside of itself. But closures let you associate a set of data directly with a function. Because of this, closures can be used for private storage. Private data will only be accessible to functions defined in the same scope as the data/ Since JavaScript is scoped by functions, that closure is defined within a function:

```
function actionTracker(choice) {
  var tracking = {};
  return function(action) {
    if (action) {
```

```
      tracking[choice] = action;
    }
    return tracking[choice];
  };
}

var redTracker = actionTracker("red");
var blueTracker = actionTracker("blue");

console.log(blueTracker()); //undefined
blueTracker("click");
console.log(blueTracker()); //"click"
redTracker("touch");
console.log(redTracker()); //"touch"
```

The function returns another function.

Another cool thing is that you can extend it to create private methods:

```
var actionTracker = function(choice) {
  var tracking = {};
  return {
    setAction: function(action) {
      if (action) {
        tracking[choice] = action;
      }
    },
    getAction: function() {
      return tracking[choice];
    }
  };
};

var redTracker = actionTracker("red");
redTracker.setAction("click");
console.log(redTracker.getAction()); //"click"
```

We return an object with a couple of methods in it acting as getter and setter.