Introduction

In this section, we're going to **publish the data** that's inside the `PlayersList` collection. Conceptually, you can think of publishing data as specifying what data should be available to users.

Publishing Data

To achieve this, delete the `console.log` statement from the `isServer` conditional and replace it with a `Meteor.publish` function. Then, as the first argument, pass through a string of `thePlayers`. This argument is simply a name that we'll use as a reference. As the second argument, pass through a function, and within this function we can specify what data should be available to users of the application. In this case, we'll return all of the data from the `PlayersList` collection using a `return` statement and the `find` function:

```
if (isServer) {
    Meteor.publish('thePlayers', function() {
        return PlayersList.find();
    })
}
```

This will duplicate the functionality of the autopublish package so it's not exactly what we want, but it's a step in the right direction. By publishing data, we're essentially transmitting that data from the server and out into the ether.

But for the data to become available to users, we need to subscribe to that data from the client side. This is easier to understand in practice. Inside the `isClient` conditional write:

```
Meteor.subscribe('thePlayers');
```

Save the file, switch back to the browser, and notice that the `find` and `fetch` functions once again retrieve all of the data from the collection. This means we've finished duplicating the functionality of the autopublish package, so now, we can work on making everything a lot more secure.