Sessions

When a user clicks on one of the player list items, a function is executed, and when this happens, we want to change the background color of that player's list item. This will create the effect of that player being selected. To achieve this, we're going to use a feature of Meteor known as `sessions`.

Sessions allow us to store small pieces of data that are not saved to the database and will not be remembered on return visits. This sort of data might not sound immediately useful, but as we'll see, it's a surprisingly versatile way to solve a lot of common problems. To begin using sessions, we need to first install a package.

Installing a Package

In Meteor, a `package` is simply a plugin that adds a range of useful features to a project in a matter of seconds and reduces the amount of code we need to write. By default, every Meteor project has local access to a number of official packages. These packages are designed by the Meteor Development Group, and they contain functionality that will be used by most developers at some point or another, but not necessarily for every project.

Because of this, the functionality is made available as an optional extra. Prior to Meteor version 1.3, the functionality for creating sessions was included with every project by default, but it has since been removed from the core software and turned into a package. To install this package, switch to the command line and write the following command

```
meteor add session
```

Creating a Session

The best way to understand how sessions work is to create one. So to begin, add the following code to the `click` event:

```
Template.leaderboard.events({
    'click .player': function() {
        Session.set('selectedPlayer', 'test');
}
});
```

The first argument is the name of the session. This is a reference we'll use to retrieve the value of the session at a later time. The second argument is whatever data we want to store inside the session.

To then retrieve the value of this session, use a `Session.get` and pass through the name of a session to retrieve the value that's stored within that session.

```
Template.leaderboard.events({
    'click .player': function() {
        Session.set('selectedPlayer', 'test');
        var selectedPlayer = Session.get('selectedPlayer');
        console.log(selectedPlayer);
}
});
```

We could have done this without using sessions, but now that we're familiar with the syntax, we can do something that's a bit more interesting.