Introduction

When a user clicks on one of the player's list item, we want to change the `background-color` property for that list item. This will create the effect of that player being selected.

To achieve this, open the project's CSS file and create a class named `.selected`:

```
.selected { background-color: yellow; }
```

Next, switch to the JavaScript file, and create a `selectedClass` helper for the leaderboard template. As we covered before, it's possible to create multiple helper functions within a single block of code, but we must remember to separate them with commas:

```
Template.leaderboard.helpers = {
    'selectedClass': function() { return 'selected' }
}
```

For this to work the string that's being returned must be equal to the name of the class in the CSS file. Back in the HTML file, place a reference to the `selectedClass` function inside the list items `class` attribute:

```
<li class="player {{selected}}"></li>
```

Because of this, the `selectedClass` will now be applied to every list item. This will change the background colors of these items to yellow. This isn't exactly what we want, but it's a step in the right direction.

selectedClass

Before we continue though, I want to demonstrate something. Inside the `selectedClass` function, comment out the `return` statement and replace it with another `return` statement that returns `this._id`:

```
Template.leaderboard.helpers = {
    'selectedClass': function() { return this._id }
}
```

Here, we're using this._id to retrieve the unique IDof the player that's currently being iterated through by the each block.

It's important to know that because the selected class function is inside the leaderboard template's each block, it has access to all of the data associated with each document, including the `name` field, the `score` field, and the `_id` field.

selectedPlayer Variable

Knowing this, we'll do a few things. First, remove the `return` statement we just wrote. Second, uncomment the `return` statement from before. Third, create a `playerId` variable that holds the value of `this._id`. Fourth, create a `selectedPlayer` variable that retrieves the value of the selected player from session. And fifth, wrap the `return` statement in a conditional that checks to see if the `playerId` and `selectedPlayer` variables contain equal values.

```
Template.leaderboard.helpers = {
    'selectedClass': function() {
        var playerId = this._id;
        selectedPlayer = Session.get('selectedPlayer');
        if (playerId === selectedPlayer) { return 'selected' }
    }
}
```

When a user clicks on one of the players list items, the unique ID of that player is stored inside the `selectedPlayer` session. The ID in that session is then matched against all of the IDs of the players in the list. Because the player's ID will always be unique, there can only be one match. And when that match occurs, the static text of selected will be returned by the `selectedClass` function, and placed inside the class attribute for that player's list item. Then, as a result of the selected class being applied, the background color of that list item will change to yellow. This creates the effect of players being selected whenever they're clicked.