Introduction

Inside the click increment event, remove the `console.log` statement and replace it with a `PlayersList.update` function:

```
Template.leaderboard.events({
    'click .increment': function() {
        var selectedPlayer = Session.get('selectedPlayer');
        PlayersList.update({_id: selectedPlayer}, {score: 5});
    }
});
```

This update function allows us to modify a document that's already in the collection, which will allow us to increment the selected player's score. We pass the ID of the document and data in the JSON format.

But there is a problem. If you select a player and click the button, the name of that player will disappear while the value of the `score` field will be changed to five. This might seem like a bug, but by default, the `update` function works by deleting the original document and creating a new document with the data that we specify. The value of the `_id` field will remain the same. Because we've only specified the score field inside the update function, that's the only other field that will continue to exist after the document is modified.

Update Function

To account for this, we need to use a Mongo feature that allows us to set the value of the `score` field without deleting the original document:

```
Template.leaderboard.events({
    'click .increment': function() {
        var selectedPlayer = Session.get('selectedPlayer');
        PlayersList.update({_id: selectedPlayer}, {$set: {score: 5}});
    }
});
```

The `$set` operator allows us to modify fields in a document without deleting the original document Because of this change the `update` function won't be completely broken.

At the moment, we're only setting the value of the score field, rather than incrementing it, so no matter how many times we click the button,the value of the score field will never go any higher than five.

INC Operator

To fix this problem, replace the `set` operator with an `inc` operator. By using this special Mongo operator, whenever the `update` function is triggered, the value of the selected player `score` field will now be incremented by a value of five:

```
Template.leaderboard.events({
    'click .increment': function() {
        var selectedPlayer = Session.get('selectedPlayer');
        PlayersList.update({_id: selectedPlayer}, {$inc: {score: 5}});
    }
});
```

What's also neat is how easily we can allow users to decrement the scores of players using very similar code. Inside the leaderboard template, create a "Take 5 Points" button with a class of `decrement`:

```
<button class="decrement">Take 5 Points</button>
```

Then inside the JavaScript file:

```
Template.leaderboard.events({
    'click .decrement': function() {
        var selectedPlayer = Session.get('selectedPlayer');
        PlayersList.update({_id: selectedPlayer}, {$inc: {score: -5}});
    }
});
```

This reverses the effect of the operator, meaning the "Take 5 Points" button will now decrement the selected player's score.