Introduction

We've finished building the original leaderboard application, but there's still a number of features we can add to the project. In this chapter, we're going to create a form that allows users to add players to the leaderboard and a button that allows users to remove players from the leaderboard.

To begin, create an `addPlayerForm` template inside the HTML file. Then include this template within the leaderboard template. Inside the template, create a form that contains two elements: a text field with the name attribute of `playerName` and a submit button with the `value` attribute of "Add Player":

```
<template name="addPlayerForm">
    <form>
        <input type="text" name="playerName">
        <input type="submit" value="Add Player">
    </form>
</template>
```

And in the same way we created a click event, we can use a submit event type to trigger the execution of code when the user submits a form:

```
Template.addPlayerForm.events = function() {
    'submit form': function() {}
}
```

Creating the Form

You might be wondering, why don't we just use the click event type for the form? Won't most users click the submit button anyway? That might be the case, but it's important to remember that forms can be submitted in a number of ways. You can submit them by clicking the Submit button, but you can also submit them by tapping the `Enter` key on your keyboard. So by using the submit event we are able to trigger the execution of code, no matter how the user submits the form.

To confirm the event is working as expected, place a `console.log` statement inside the events function. But as it turns out, there's a problem, because when we submit the form, the browser refreshes the page. You might see a quick flash of the form submitted message in the console but we've, nevertheless, encountered an error that needs to be fixed.

This happens because, by default, the web browser assumes that the data in the form should be sent to an external page. But when working with Meteor, we don't want the data to be sent to an external page. We want the data to remain on the current page. To account for this, what we need to do is disable the default behavior that web browsers attach to forms. This is something we'll soon fix.