Events

In this chapter, we're going to create our first **event**.

Events allow us to trigger the execution of code when a user clicks on a button, taps a key on their keyboard, or completes a number of other actions. To demonstrate this, add the following code to the `isClient` conditional:

```
Template.leaderboard.events({});
```

The `Template` keyword searches through the templates in our project. The `leaderboard` keyword is the name of the template to which we're about to attach some events. And the `events` keyword is what we use to define a block of events in the JSON format.

Creating an Event

To create an event, first define an event type. In this case, we'll create a click event that will trigger whenever the user clicks anywhere within the bounds of the leaderboard template. Next, attach a function to the event. The code within this function will execute whenever the click event is triggered:

```
Template.leaderboard.events({
    'click': function() { console.log('click'); }
});
```

The problem with the event we've created is that it's too generic. It triggers when the user clicks inside the bounds of the leaderboard template, but it'd be a lot more useful if it triggered when the user did something specific, such as clicking on a particular button.

Event Selectors

To achieve this we'll use **event selectors**. They allow us to attach events to specific HTML elements. If you've ever used jQuery before, this process will be familiar. But if not, then it should still be quite easy to grasp.

Earlier, we placed list item tags inside the leaderboard template. The plan now is to make our event trigger when the user clicks on one of these elements. To do this, place a reference to the `li` tag right after the click event type:

```
Template.leaderboard.events({
    'click li': function() { console.log('click'); }
});
```

Because of this change, you will now see that clicking on any of the list item elements will output a message to the console.

The problem with this code is that if we had other list item elements inside this template that were not a part of the list of players, then the event would trigger when it wasn't supposed to trigger. To account for this, attach a class of `player` to the list item elements inside the `each` block:

```
<li class="player"></li>
```

Make it so the event will trigger when any elements with the class of `player` are clicked:

```
Template.leaderboard.events({
    'click .player': function() { console.log('click'); }
});
```