Event Object

When an event is triggered from within a Meteor application, we're able to access information about that event as it occurs. That might sound a bit weird, but here's a quick demonstration:

```
Template.addPlayerForm.events = function() {
    'submit form': function(event) {
        console.log(event.type);
    }
}
```

You don't have to use the word `event`, but whatever word you use as the first parameter can be used to reference the event.

But because we haven't fixed the original problem of the form refreshing the page with each submission, it still won't work as expected. To fix our problem, refer to the event from inside the `events` function and use a `preventDefault` function. This function prevents the default behavior of an event from occurring, which means the console.log statements will work as expected and will have complete control over the form.

```
Template.addPlayerForm.events = function() {
    'submit form': function(event) {
        event.preventDefault();
        console.log(event.type);
    }
}
```

Next, let's grab the value of the player name text field whenever the submit form event is triggered.

Creating a Variable

We can then use the value of that text field when we add a player to the database.

```
Template.addPlayerForm.events = function() {
    'submit form': function(event) {
        event.preventDefault();
        var playerName = event.target.playerName;
        console.log(playerName);
    }
}
```

By using `event.target.playerName` we're grabbing the `playerName` text field.

Notice that when the form is submitted, the raw HTML of the text field appears in the console, rather than the value from the text field. This is because we need to explicitly retrieve the `value` property of the text field:

```
Template.addPlayerForm.events = function() {
    'submit form': function(event) {
        event.preventDefault();
        var playerName = event.target.playerName.value;
        console.log(playerName);
    }
}
```

To insert the submitted player into the database, we can use the `insert` function that we talked about earlier:

```
Template.addPlayerForm.events = function() {
    'submit form': function(event) {
        event.preventDefault();
        var playerName = event.target.playerName.value;
        PlayersList.insert({name: playerName, score: 0});
    }
}
```

You'll notice that after submitting the form, the player's name remains in the text field. To fix this, reference the text field from the bottom of the event and reset its value property to an empty string:

```
Template.addPlayerForm.events = function() {
    'submit form': function(event) {
        event.preventDefault();
        var playerName = event.target.playerName.value;
        PlayersList.insert({name: playerName, score: 0});
        event.target.playerName.value = '';
    }
}
```