Creating Methods

Up until this point, the `insert`, `update` and `remove` functions have been only sitting inside our events. This has been the easy approach, but it's also why our application has been insecure. We've been placing these sensitive database related functions on the client side, inside the user's web browser.

The secure approach is to move these functions into **methods**. Methods are simply blocks of code that can be triggered from elsewhere in an application. To demonstrate how methods work, let's create one. Write the following code outside of the `isClient` and the `isServer` conditionals

```
Meteor.methods = {
    createPlayer: function() {
        console.log('create');
    }
};
```

The syntax is ultimately very similar to both helpers and events.

Next, return to the submit form event and place the following statement after the insert function:

```
Meteor.call('createPlayer');
```

This means whenever the form is submitted, the code inside the `createPlayer` method will be executed.

Demonstrating Methods

To demonstrate this, switch back to the browser and use the form. The `insert` function still won't work, but we'll see that the `console.log` message from the method appears inside the console.

To get the `insert` function working again, move both it and the `currentUserId` variable from the submit form event into the `createPlayer` method, but replace the reference to `playerName` with a value of "David". This isn't exactly what we want, but it's a step in the right direction. Based on these changes, the "Add Player" form will now kind of work. If we submit the form, a player of David will be added to the list.

Notice that if we try to use the `insert` function from inside the console, we still won't have permission to do so. This is because we're starting to gain controlover how users interact with the database. We removed the insecure package, so users can't use any of the database functions inside the console. They can only interact with the database by executing the method. And since we're the ones defining how methods work, it becomes much easier to control exactly what users are able to do.