
## Introduction

In an earlier version of this course, I suggested that methods should be made to run on the server from inside the `isServer` conditional, but this is not actually the best approach. We can place methods inside the `isServer` conditional and then call those methods from the `isClient` conditional. And in some cases, this is necessary, but in the majority of cases, it's not the best option. As such, I just want to take a moment to explain why methods should be placed outside of these conditionals, since it's a very important part of what makes Meteor so interesting.

As we talked about before, code that is placed outside of the `isClient` and `isServer` conditionals, will execute on both the client from inside the user's browser and on the server, where the application is hosted. That code that can also behave differently depending on the environment. This is something that happens in the case of Meteor methods.

Let's focus on the `createPlayer` method that we've created. When this method is executed on the server, it behaves as you would expect it to behave. It grabs the unique ID of the currently logged in user and inserts some data into the `PlayersList` collection. But when this method is executed on the client, it creates an instantaneous simulation of the code that appears inside this method. Basically, when the method runs on the client, it guesses what the method on the server is trying to do and instantly reflects those changes from inside the browser.

This is partly why Meteor is realtime by default. When we're adding a player to the list for instance, the change will happen inside the interface without any delay. But behind the scenes, it actually takes a little bit longer for the data to be inserted into the database. Because of the simulation happening in the web browser though, users will never see this delay. As far as they're concerned, everything is happening instantly, which is great for the user's experience, but there's actually a lot more happening than they will ever realize.

## Optimistic UI

This feature is part of what Meteor refers to as **Optimistic UI**. The creators of Meteor wrote a great article about it if you're interested in learning more. At this point though, it doesn't really matter if you don't understand the precise details of what's happening behind the scenes. The beauty of Meteor is that, even as a developer, all of this is relatively transparent. It is, however, useful to have at least a general understanding, so you know why you're putting certain code in certain places. That will only make it easier to understand the practical details.