Arnaud Weil

Learn Meteor

Node.js and MongoDB JavaScript platform

Learn Meteor - Node.js and MongoDB JavaScript platform

Be ready for coding away next week using Meteor

Arnaud Weil

This book is for sale at http://leanpub.com/learnmeteor

This version was published on 2017-09-11

To my wonderful family. Your love and support fueled this book.

To my Verallia and Bruitparif clients, whose demanding projects helped me discover and challenge Meteor.

Contents

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor

1. Introduction

1.1 What this book is not

I made my best to keep this book small, so that you can learn Meteor quickly without getting lost in petty details. If you're looking for a reference book where you'll find answers to all the questions you may have within the next 4 years of your Meteor practice, you'll find other heavy books for that.

My purpose is to swiftly provide you with the tools you need to code your first Meteor application and be able to look for more by yourself when needed. While some authors seems to pride themselves in having the thickest book, in this series I'm glad I achieved the thinnest possible book for my purpose. Though I tried my best to keep all of what seems necessary, based on my 14 years experience of teaching.

I assume that you know what Meteor is and when to use it. In case you don't, read the following *Why Meteor ?* chapter.

1.2 Prerequisites

In order for this book to meet its goals, you must :

- Have basic experience creating applications with JavaScript.
- Have working knowledge of HTML.
- Know what a Web application is.

1.3 How to read this book

This book's aim is to make you productive as quickly as possible. For this we'll use some theory, several demonstrations, plus exercises. Exercises appear like the following:

Do it yourself: Time to grab your keyboard and code away to meet the given objectives.

1.4 Tools you need

The only tools you'll need to work through that book are:

- A Windows, Linux or OS X machine
- A text editor. My own favorite is the free Visual Studio Code[1], just pick the one you like

[1] https://code.visualstudio.com

1.5 Source code

All of the source code for the demos and do-it-yourself solutions is available at https://bitbucket.org/epobb/learnmeteorexercises

It can be downloaded as a ZIP file[2], or if you installed GIT you can simply type:

```
git clone https://bitbucket.org/epobb/learnmeteorexer\
cises.git
```

[2] https://bitbucket.org/epobb/learnmeteorexercises/get/c50195f7592a.zip

2. Why Meteor ?

If you're in a hurry, you can safely skip this chapter and head straight to the Beginning with Meteor chapter. This *Why Meteor* chapter is there for those that want to know why Meteor should be used.

2.1 How I felt in love with Meteor

I've been coding a lot of data-centric applications using various technology stacks. They usually involve displaying and updating data from some store (mainly a database) through a user interface and APIs, adding some functional logic in the way.

Problem is: most technology stacks involve writing quite some code for basic things, and often require different coding logics on the client and server.

One of my clients presented me with a challenging application to build. After rough evaluation of the time needed, writing a proof of concept required dozens of days. That was too much, so I took a full day and looked for available technology stacks. By the end of the day I stumbled upon Meteor. It promised a lot, but would it hold to its promises?

Well, it did. In five days I had the proof-of-concept coded and deployed on a staging server. A few days later it

appeared that Meteor could also meet the requirements for another client's project, except that the requirements needed low-level network I/O. Turns out Meteor allowed me to leverage Node.js in a breeze, and I had the first draft running in hours.

I love tools that make me productive for common tasks while also allowing for greater power and customization when needed. Meteor does just that, providing simple ways to answer most requirements of today's applications.

2.2 Why is Meteor so productive?

Meteor makes writing applications a much faster process than many other Javascript environments. Here are some of the reasons why:

- Everything a developer needs is installed in a breeze, on any major OS (Linux, Windows, OS X);
- JavaScript code on the server and client, some of the code may even be shared between the two;
- Native support for MongoDB collections (if you don't know MongoDB we'll talk about it a bit later);
- Automated synchronization of the data between client and server;
- Very little code to write;
- Most of the functionality a developer usually needs is provided as easy-to-install packages;
- A very helpful command-line utility.

- Out-of-the-box effortless minification and file combine;
- Simple creation of packages;
- Integrated debugging of server code right in the browser;
- Straightforward unit-testing of created packages.

2.3 Where is the catch?

Too often you invest time and energy in a technology stack, only to find out its drawbacks a while later.

After some extensive use, Meteor drawbacks came when deploying. This is personal experience, but I was faced with two problems.

First, as long as you are willing to host your apps in Meteor's cloud solution (*Galaxy*), everything goes well. If you want to host Meteor apps in your custom environment, you'd better use a Linux machine. In which case things go quite smoothly when you don't require a server farm.

As of writing, deploying to a production Windows server was so difficult that I stopped in the process. That's a real pain, to say the less. This is strange since the Meteor team worked hard to bring excellent support for developing on a Windows environment. However, my problems were solved when using Docker for deployment. Dockerized Meteor applications can be deployed on any platform that support Docker, which means Linux, Windows or cloud with no surprise.

Second, the applications deployed had performance problems. There are many things that can be done in order to improve Meteor application performance, but it's quite a disappointment to spend a lot of time on performance improvements when the development stage was so quick.

All in all, though you can learn Meteor very quickly (that's the purpose of this book), bear in mind that fully understanding Meteor takes time. It's a quick way to create applications thanks to bold assertions from the Meteor stack and many things going on under the hood. You'll be able to create great Meteor applications within a few days, but you'll need to invest some time to fully understand Meteor's engine.

What I mean is that most tasks that usually take hours of coding will take minutes using Meteor, but during the first days be prepared to get stuck several times on some documentation reading. All in all you should gain a lot of time anyway, but you'll need to invest yourself in order to really master the Meteor stack.

3. Beginning with Meteor

3.1 Setting up your development machine

You can develop Meteor applications using a Windows, Linux or OS X machine, and this book applies to all of them.

In order to setup your machine, just point your browser to the install page[1].

For Windows machines, there is an installer. On a Linux or OS X machine, just type:

```
curl https://install.meteor.com/ | sh
```

3.2 Creating a meteor application

Meteor boosts your productivity as a developer. While most JavaScript development environments rely on the help of scaffolding tools like Yeoman in order to hide the fact that you need many files for even a basic application, Meteor needs only three files to code your application.

[1] https://www.meteor.com/install

You read well: three files. That's all there is to it. Of course you can use a full folder hierarchy and as many files as needed to organize your application, but that's optional. Simply put, Meteor applies a good coding pattern from the start: KISS (keep it simple).

Although you need only three files, Meteor even makes that process easy. Let me show you how I create my *demos* application. I'll use my demos to showcase the techniques I want you to learn, before you get some practice for yourself in the exercises. You'll soon create your own application, but for now please bear with me: sit and relax, and look at my demonstration.

I want my application to be called *demos* so I simply open a command line and type:

```
meteor create demos
cd demos
```

That's it! The first line creates a "demos" directory and adds three files to it. It also adds a `.meteor` folder, but it's just a local storage you won't need to manage or go into.

Time to run my application. Guess what? Yes, you're right: Meteor makes it simple. Again, using my command-line I type:

```
meteor
```

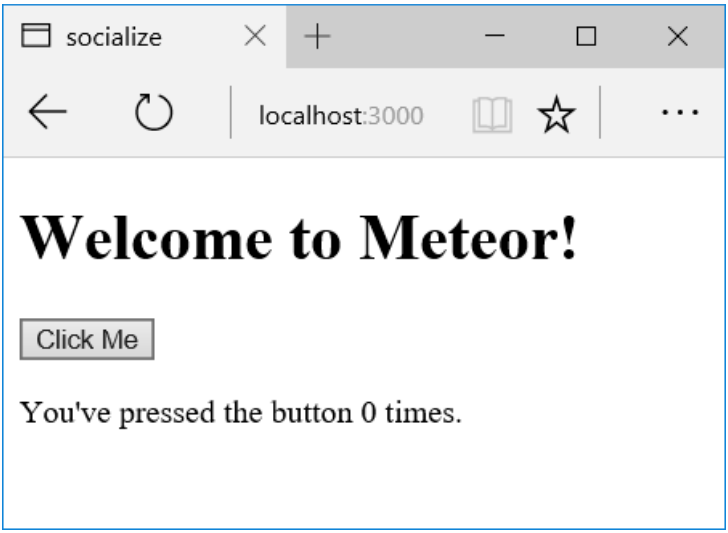
That's a shortcut for `meteor run`. It builds my application, runs a local MongoDB database server, and runs an HTTP server (using Node.js) that serves the application.

Now I can simply open a Web browser and type in the application URL:

```
http://localhost:3000
```

> In case you don't like the default port (3000), you can use the `--port` argument of the `meteor run` command.

What do I see in my browser? Well, not much but that's a good start:



Alright, I didn't code anything yet but I already have a local database and server that render an HTML page. Out of the box, that page doesn't do more than increase a counter when you click the button, but it won't take long to fill that page with data and functionality.

3.3 What are those files ?

As mentioned earlier, there are currently three files that make up my application. In fact, I could code my whole application using those three files. Of course, for maintenance purposes it would be a good idea to split them up, but that's fully optional.

Let's take a look at those files. We have:

- demos.html contains the HTML that renders on the client;
- demos.js contains code that executes on the server and/or the client, and may be used by the HTML code;
- demos.css is, well, the CSS that applies to your HTML;

Sounds obvious? Great! In fact it is, but we just overlooked a time-saving feature of Meteor: you can mix server **and** client code in the same file. More about that later.

3.4 Meteor is listening to your code

Time for some magic: if I change whatever of those files, the result will automatically appear in my browser after a few seconds. If you never used a good development stack this may not feel so natural.

When I typed the `meteor` command in my command-line, it didn't just build the application and listen for HTTP queries. It also stays there listening for file changes. When a change is detected it builds the application again, and it also sends a message to the connected browsers so that they reload.

As a developer, this means all you have to do is change your files, and your browser refreshes. Best of all, this also works in production: any connected browser to your application will automatically refresh with the latest version whenever you publish a new version.

3.5 The HTML file

Let me open the `demos.html` file. Here's what we find inside:

```
<head>
  <title>demos</title>
</head>

<body>
  <h1>Welcome to Meteor!</h1>

  {{> hello}}
</body>

<template name="hello">
  <button>Click Me</button>
  <p>You've pressed the button {{counter}} times.</p>
</template>
```

You can see three base elements: `head`, `body` and `template`. Those are the only three elements that can be at the root of your HTML. The template element will be repeated, but any other HTML element should be inside one of those three elements.

Let's detail those three elements.

head

When Meteor runs your application, it does a lot of stuff for you, including referencing the JavaScript and CSS file, plus bundling and minification for those. For this, it will locate the `head` element and add links to those generated resources there.

body

That's where you put the HTML that renders at first to the user. Sure, you may feel that this is the easiest place to put your HTML elements, but that would make maintenance a nightmare.

Since you're a clean developer (I know it, because you're still reading), you won't want to put more than a few lines inside the `body` element. Later, we'll mostly use it as display for our routing engine. But hey, we don't necessarily need a routing engine to be clean. So what I recommend you do here is only use references to your templates.

Did I say *templates*. Time to look at the third element.

template

As I said, this element will be repeated. You create as many templates as you want, provided you assign a different `name` attribute to each.

Then you reference your template elements from the `body` element or from other templates using the mustache syntax with a `>` element:

```
{{> templateName}}
```

> Alright, this is not pure HTML, we added some *mustache* syntax. More about this a bit later. Just accept it like that, for now.

body + template + template + ...

In fact, if we look again at the `demos.html` file, we can see such a reference:

```
<body>
  ...
  {{> hello}}
</body>

<template name="hello">
  ...
</template>
```

What this means is that the content of the `template` element named *hello* must go inside the `body` element.

Which means I can write:

```
<body>
  {{> header}}
  {{> someGreatInfo}}
  {{> footer}}
</body>

<template name="header">
  ...
</template>

<template name="someGreatInfo">
  ...
</template>

<template name="footer">
  ...
</template>
```

I could also reference a template from within another template:

```
<body>
  {{> home}}
</body>

<template name="home">
  {{> latestNews}}
  ...
</template>

<template name="latestNews">
```

```
  ...
</template>
```

Sure, I could elaborate on that syntax but I'll do that later. For now, you already learned a bunch of new things and it's time to put them into practice.

3.6 It's your turn to code: do-it-yourself

Now is your turn to grab the keyboard and code away. Oh, just let me explain you how that works, in case you're not familiar with my *Learn* books.

About exercises in this book

All of the exercises are linked together: you're going to build a small e-commerce application. You'll allow users to browse for your products, add them to their basket, and you'll also create a full back-end where the site administrators will be able to list, create, modify, and delete products.

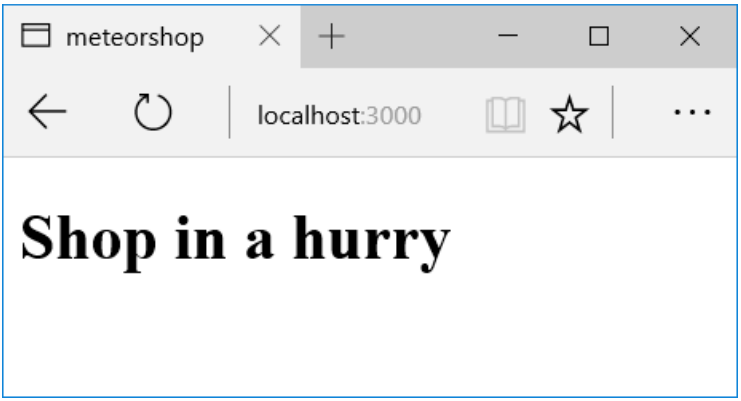
In case you get stuck

You should be able to solve the exercise all by yourself. If you get stuck or don't have a computer at hand (or you don't have the prerequisites for that book, which is fine with me!), here's the solution. I'll provide the solution for all of the exercises in this book, right after each of them.

3.7 Exercise - Create the application

Create a new Meteor application named *meteorshop*. Add a <h1> title to the page that reads “Shop in a hurry”. Remove the button and counter.

Your application should look like the following:



I know, it’s basic, but you need to learn some more things before you can do more. Beginner badge unlocked: let’s proceed to the next level.

3.8 Exercise solution

- Open a command-line and navigate to an empty development folder or create one.

- In the command-line, type `meteor create meteor-shop`.
- In the command-line, type `cd meteorshop`.
- Open the `meteorshop.html` file. Locate the following code:

```
<body>
    ...
</body>
```

- Replace that code with the following one :

```
<body>
    <h1>Shop in a hurry</h1>
</body>
```

- In the command-line, type `meteor`.
- Wait for the application to generate and run.
- Open your browser and type the following URL:

  http://localhost:3000

4. Blaze, Spacebars and reactivity

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor

5. Managing data from a MongoDB database

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor

6. Packages: admin dashboard, navigation, validation, forms generation

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor

7. Accounts: user management

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor

8. Accounts: user management

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor

Definitions

This is just a sample of the full book.

If you like it, get your full version here: https://leanpub.com/learnmeteor