Front-end revolution

with

Ember.js

Preface

My experience with Ember started around two years ago and no other thing in my web development career kept me interested with such intensity and passion for such a long time. It's remarkable how the discovery of Ember.js framework made such a pleasant impact in my life and erode so much of development anxiety I was experiencing, mainly because of fear of missing out technologically in this crazy ecosystem of web development tools.

With Ember I feel like we've reached some kind of a plateau and we finally found the right way to do web applications.

If my words in this book manage to expand your horizons in any possible way, then my goal has been achieved, and I could not be happier.

Yours truly,
Matic Jurglič (@matixmatix on Twitter)
http://www.codeandtechno.com/

Book resources

Repository for Ember.js client code:

https://github.com/matixmatix/red-green-client

Repository for API back-end code (written in Ruby on Rails):

https://github.com/matixmatix/red-green-api

The deployed end product we'll incrementally be building with this book:

https://red-green.pagefrontapp.com

Book's website:

https://leanpub.com/front-end-revolution-with-ember-js

1 About this book

Purpose

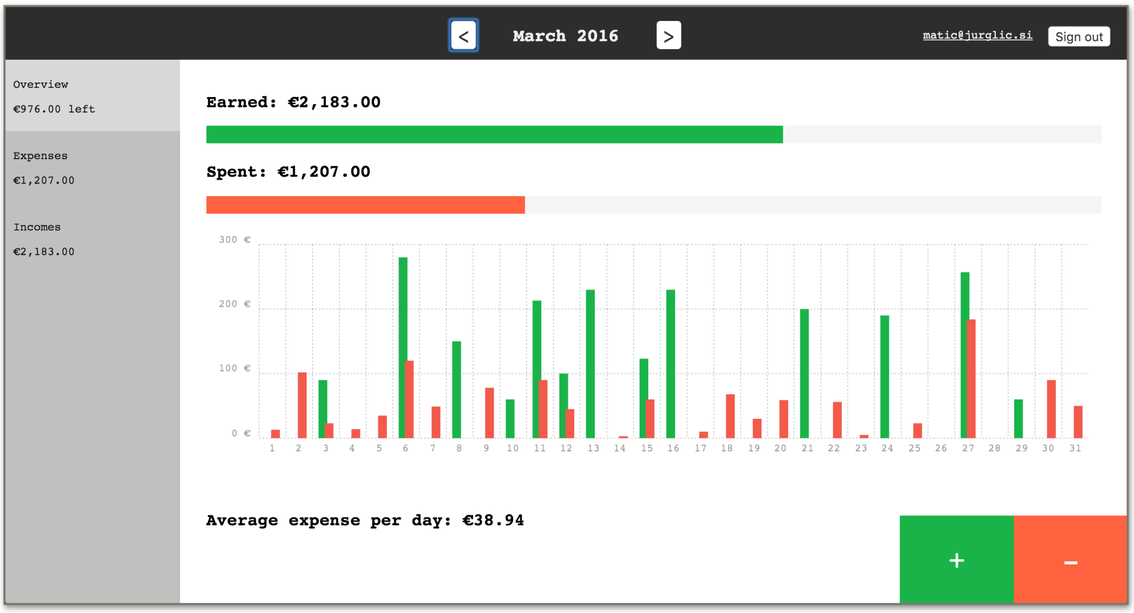
As you already know, Ember.js is a front-end JavaScript framework which comes with everything you need to build ambitious web applications. It's a performant whole-sale solution that brings sanity to the front-end.

Ember continues to be an antidote to the insane madness in the JavaScript ecosystem. It's opinionated for sure, but that means it's very easy to get started with and maintain it.

The purpose of this book is to educate readers about the fundamentals and up-to-date best practices of building Ember.js applications and provide a well-written, concise tutorial on how to build rock-solid web application that don't suck.

Objective

Chapters and examples in this book are tightly narrated by the natural progression of building a real-life application. The application we'll be building is an expense tracking product, called **red**-**green**. In the end, it will look nothing short of amazing:



Overview
€976.00 left

Expenses
€1,207.00

Incomes
€2,183.00

Earned: €2,183.00

Spent: €1,207.00

Average expense per day: €38.94

We'll cover every part in the process; from initial prototyping, laying down the routes, connecting to back-end, logging in, signing up, binding properties to templates, listing, editing, persisting records, animating, to making builds and deploying.

You can take it for a spin here: https://red-green.pagefrontapp.com

Here is the source code: https://github.com/matixmatix/red-green-client

The back-end API is also provided for you to use while developing out application. You can either:

1. Get the code from https://github.com/matixmatix/red-green-api and run it on your machine. It's written in Ruby on Rails.
2. Use already deployed one: https://red-green-api.herokuapp.com/api/v1

Whom is it for

This book has been written with a single noble goal in mind - to help fresh Ember developers to get up to speed with the current proper ways of making Ember applications, in the shortest time possible. It could also serve a an excellent source for people who are dealing with JavaScript fatigue and want to get some insights on how things are done in the Ember world.

There are no extra prerequisites other than basic knowledge of JavaScript and general web development.

Let this book convince you it's worth building your next rich application with Ember.js.

How is it different from other books?

What I've noticed with some other Ember teaching material is that it's often outdated and not providing much input on Ember's near future developments. As a person who spends a lot of time interacting, listening and helping people with Ember related questions, I developed a feel on what people are often confused, or want to know more about. To make this learning experience as thorough as possible, the learning process described here is tightly coupled with my findings on most common mistakes people make in Ember.

Another thing I noticed is that people oftentimes struggle with API authentication and authorization of users in Ember applications - signing up and logging in. A heavy part of this book is dedicated to this topic, offering a concrete implementation while narrating it in an abstract way, introducing higher level concepts you should use if you wish to roll out your own authentication solution.

5 things you need to know before starting with Ember

1. Ember is mature and ready

People used to associate Ember with sad tags like; #buggy, #immature, #tough, #undocumented. Ember has evolved big-time and the tough parts to learn have been eradicated, and all the rest has been simplified. The guides are wonderful and comprehensive (https://guides.emberjs.com).

To name just a few companies that use Ember in production: Apple, Heroku, Twitch, Vine, NBC,…, NASA (check more on http://builtwithember.io/).

You read that right. A space agency uses Ember.

2. Ember liberates you from wasting time with already solved problems

Once you get a feel of the Ember way, there’s a very slim chance you’ll want to abandon it. Tough questions that don’t really matter in the large scheme of things, like, “how do I structure my application folders and files”, “what set of tools should I pick for building code and assets” and “how do I name my files" have already been solved for you, so you can focus on building right away. With its set of conventions Ember developers can jump between different applications and feel like at home with any of them.

3. Ember moves fast

One thing you should keep in mind is that Ember.js is a constantly evolving framework. Sometimes things get deprecated and/or replaced with something

better, but always for good reason Luckily, upgrade procedures are pretty painless (it follows Semantic Versioning), but it's your duty to keep up with the latest releases in order to keep your application sharp and avoid technical debt.
Also, please stay away from Ember resources written more than a year ago, as they are often outdated.

4. Ember makes you live in the future

Things like the next generation of JavaScript (called ES6 or ES2015), managing asynchrony with promises and modular design may still not be very common to all JavaScript developers out there, but Ember developers already use them by default. ES6 syntax makes writing JavaScript suck a lot less and makes your app more future-proof. The code is automatically transpiled with Babel to older JavaScript version that browsers understand. When browsers fully adopt the new generation of JavaScript, Ember apps will already be compatible and take advantage of native ES6 features.

5. We're here for you

Ember has the friendliest community I had ever seen in open source software.
You can go to Slack and chat with Ember (core) developers just about anything, get insights about future developments and get your problems solved in a matter of minutes. There are also hundreds of add-ons made by the community.

It is very likely somebody already solved the challenge you're facing and published it as an add-on. Check them out on https://www.emberaddons.com and https://emberobserver.com.

2 Environment, tools and prerequisites

Ember CLI

You can develop using Ember without Ember CLI (Command Line Interface), but that would be a huge mistake. Let me tell you just a few of the reasons why:

1. It takes care of the whole build process in real-time while coding, does automatic page reloads when changing code, hot-swapping CSS code, transpiling ES6 code, dealing with modules
2. Provides a brilliant asset pipeline using Broccoli
3. Easy installing of add-ons
4. Code generators
5. Provides a localhost server for your application

Find out more on Ember CLI website.

Dependency management

Currently, Ember uses two dependency managers, npm (contained in `package.json`) and Bower (`bower.json`). This sometimes causes confusion. Bower is used for managing purely front-end concerns (jQuery, Ember,…) while npm manages internal dependencies. Given that npm system has improved and is becoming a nice place to store front-end dependencies as well, we will most probably see the end of Bower in Ember in the near future.

Initializing a new application

First, make sure you have installed node. Typing this into your terminal should output node's version:

```
node -v
v5.4.0
npm -v
3.3.12
```

Let’s prepare an initial structure. In your terminal, type in the following:

```
npm install -g ember-cli
npm install -g bower
```

That’s it. We’re ready to initialize our application structure by typing:

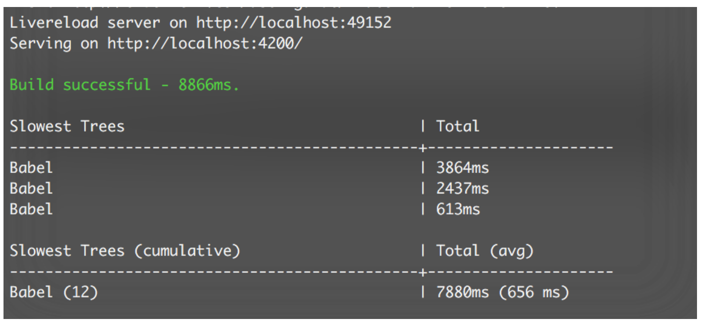
```
ember new red-green-client
```

This does a bunch of things for us. It lays down the application structure in the newly created folder, initializes a git repository, installs initial dependencies.

Now let’s fire up this baby.

```
cd red-green-client
ember server
```

The `ember server` command will invoke development server. We can shorten this command to `ember s`. For a list of all available options, just type `ember`.



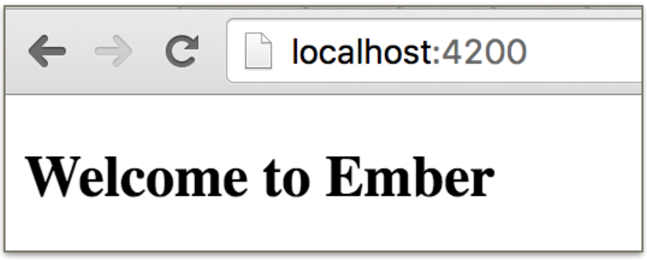
```
Livereload server on http://localhost:49152
Serving on http://localhost:4200/

Build successful - 8866ms.

Slowest Trees                                 | Total
----------------------------------------------+---------------------
Babel                                         | 3864ms
Babel                                         | 2437ms
Babel                                         | 613ms

Slowest Trees (cumulative)                    | Total (avg)
----------------------------------------------+---------------------
Babel (12)                                    | 7880ms (656 ms)
```

ℹ️ When we’re running the server, it continuously gives us info on duration of builds. Initial builds are sometimes a bit slow, but after that when we’re changing code, Broccoli identifies the modules that have been changed, and rebuilds only those, making subsequent builds super fast. It will also output JavaScript linting warnings and HTML tag errors in our templates.
The output told us our application is available on http://localhost:4200/. Visit it:


localhost:4200

Welcome to Ember

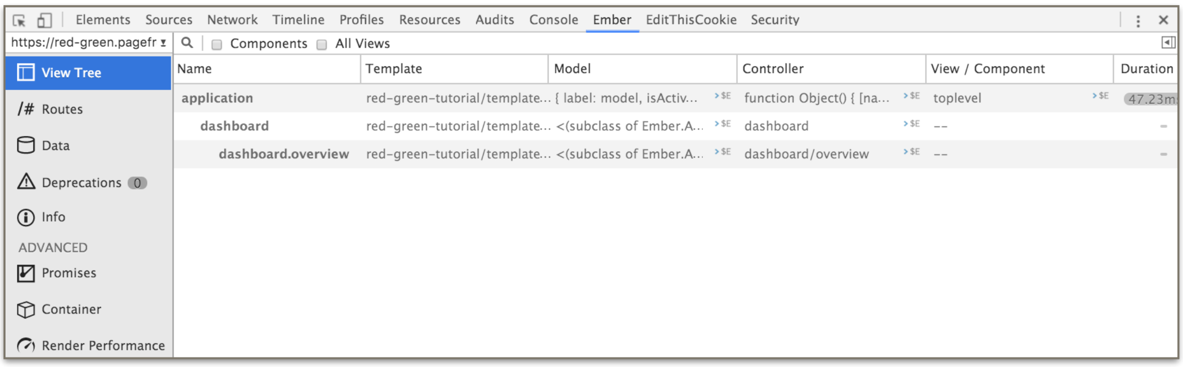
This means our application is ready and we can start fiddling around (also known as “developing”).

If you ever in your life had to configure a JavaScript build pipeline yourself, you know damn well how much of your precious time this task may take. **Too many times** in my professional history I observed teams struggling with finding and agreeing on a project structure, implementing asset chains, integrating transpilers, linters, etc. Full working days are lost on this.

With Ember, it took less than 5 minutes to get everything ready, with the best tools the ecosystem has to offer.

Ember inspector

This is a must-have tool for every Ember developer. It enables you to see under the hood of every Ember application you can navigate to, straight in your browser console. It’s available for Chrome and Firefox. Install it depending on your chosen browser.



Name	Template	Model	Controller	View / Component	Duration
application	red-green-tutorial/template…	{ label: model, isActiv…	function Object() { [na…	toplevel	47.23m
dashboard	red-green-tutorial/template…	<(subclass of Ember.A…	dashboard	--	-
dashboard.overview	red-green-tutorial/template…	<(subclass of Ember.A…	dashboard/overview	--	-

It enables you to inspect every route and template outlet in your Ember application, see what model it holds, provides access to model records and container, displays deprecations, and a lot more.

ES6, modules and the resolver

When writing files in Ember applications you'll come across a common pattern. Like in every mafia gang, it's all about import/export business. Take a look at this example of a JavaScript file in Ember:

`app/controllers/dashboard/expenses/edit.js:`

```
import Ember from 'ember';

export default Ember.Controller.extend({
  session: Ember.inject.service(),
  actions: {
    save(balanceChangeData) {
      this.get('model').setProperties(balanceChangeData);
      this.get('model').save().then((balanceChange) => {
        this.transitionToRoute('dashboard.expenses');
      });
    }
  }
});
```

The `import` and `export` keywords refer to ES6 modules (which are transpiled by Babel into AMD modules). Whenever we need something, we import it. If we want some JavaScript object to be available for importing somewhere else, we need to export it. Everything in Ember is based on this mechanism.

Ember has a special mechanism, the Ember Resolver. This is where the so-called magic happens. Let me disappoint you: there is no magic, just clever conventions. The resolver is just a mechanism that looks up code in your application depending on the naming conventions. The upper example results in a module called `red-green-client/controllers/dashboard/expenses/edit`. When we want to import this module someplace else, The Resolver will search for file with this name and provide whatever it's exported.

Add-ons

Add-ons are Ember applications you merge with your application to extend functionality. These packages live in `npm` and you can install them with a command `ember install addon-name`. Don't forget to restart your server after each install!

❗ When selecting add-ons to install, be extra cautious. Only go with add-ons that are well maintained (you can use score metric in emberobserver), otherwise you might run into trouble when upgrading your application.

Sass

A lot of people like to use Sass CSS preprocessor. To enable it, install the add-on (do it now, we'll need it later):

```
ember install ember-cli-sass
```

Now rename `styles/app.css` to `styles/app.scss.`

❗ The Sass code for the application we'll be building is already written. The scope of this book is not to deal with (S)CSS, so I just copy the style from the GitHub repository and paste it to `styles/app.scss.`

Whenever we make changes to the (S)CSS file, the CSS code in the browser is reloaded on the fly without needing to reload the whole application.