ember-cli 101

by **Adolfo Builes**

ember-cli 101

Learn Ember.js with ember-cli.

Adolfo Builes

This book is for sale at http://leanpub.com/ember-cli-101

This version was published on 2014-12-28

Tweet This Book!

Please help Adolfo Builes by spreading the word about this book on Twitter!

The suggested hashtag for this book is #ember-cli-101.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#ember-cli-101

Contents

Why

Before getting into the specifics, I'd like to explain why ember-cli was created and how it is different from other tools.

The main objective of ember-cli is to reduce what we call glue code and allow developers to focus on what is most important for them: building their app.

Glue code refers to those things that are not related to your application but that every project requires. For example, you need to test your code, compile your assets, serve your files in the browser, interact with a back-end API, perhaps use third party libraries, and so on. All those things can be automated and, as it is done in other frameworks, some conventions can provide a common ground to begin building your applications.

Having a tool that does that for you not only eases the process of writing your app but also saves you time and money (you don't have to think about problems that are already solved).

`ember-cli` aims to be exactly that tool. Thanks to `Broccoli`[1], waiting time is reduced while your assets compile. `QUnit`[2] allows you to write tests, which can then be run with `Testem`[3]. If you need to deploy your build to production, you'll get fingerprint, compression, and some other features for free.

`ember-cli` also encourages the use of ES6(ECMAScript 6)[4]. It provides built-in support for modules and integrates easily with other plugins, allowing you to write your applications using other ES6 features.

Next time you consider wasting your day wiring up all those things I just mentioned, consider `ember-cli`. It will make your life easier and you will get support from a lot of smart people who are already using this tool.

---

[1] https://github.com/broccolijs/broccoli

[2] http://qunitjs.com/

[3] https://github.com/airportyh/testem

[4] https://people.mozilla.org/~jorendorff/es6-draft.html

Anatomy

In this chapter we will learn about the main components of ember-cli.

ember-cli is a **Node.js** command line application that sits on top of other libraries.

Its main component is **Broccoli**, a builder designed to keep builds as fast as possible.

When we run `ember server`, **Broccoli** compiles our project and puts it in a directory where it can be served using **Express.js**[5], a **Node.js** library. **Express** not only serves files but also extends **ember-cli**'s functionality using its **middlewares**. An example of this is **http-proxy**, which supports the `--proxy` option that allows us to develop against our development backend.

Testing is powered by **QUnit** and **Testem**. By navigating to **http:/localhost:4200/tests**, our tests run automatically. We can also run Testem in **CI** or `--development` mode with the **ember test** command. Currently, only **QUnit** is supported and it's done via an **ember-cli add-on**. We will probably see support for other testing frameworks and runners as more people become familiar with the add-on system.

ember-cli uses its own resolver and has a different naming convention from **Ember.js's** defaults.

ember-cli makes us write our application using **ES6 Modules**. The code is then transpiled (compiled)[6] to **AMD**[7] and finally loaded with the minimalist **AMD**[8] loader, **loader.js**.

You can use **CoffeeScript** if you want, but you are encouraged to use plain JS and ES6 modules where possible. In subsequent chapters, we'll explore its syntax and features.

Finally, we need to cover plugins that enhance the functionality of **Broccoli**. Each transformation your files go through is done with a **Broccoli** plugin, e.g. transpiling, minifying, finger-printing, uglifying. You can have your own **Broccoli** plugins and plug them wherever you like throughout the build process.

---

[5] http://expressjs.com/

[6] The transpiling process is done with es6-module-transpiler.

[7] To know more about **AMD** checkout their wiki

[8] To know more about **AMD** checkout their wiki