OVERVIEW

Ember CLI is the Ember.js command line utility that provides a fast Broccoli-powered asset pipeline, a strong conventional project structure, and a powerful addon system for extension. Get started.

Ember CLI has support for:

- Handlebars
- HTMLBars
- Emblem
- LESS
- Sass
- Compass
- Stylus
- CoffeeScript
- EmberScript
- Minified JS & CSS

In addition to the above, Ember CLI's addon system provides a way to create reusable units of code, extend the build tool, and more. To view a complete list of addons, please visit emberaddons.

Modules

Ember CLI uses babel, which turns ES2015 module syntax into AMD (RequireJS-esq) modules.

Using a custom resolver, Ember CLI can automatically import modules when needed. For example, the route in `routes/post.js` will know to use the controller in `controllers/post.js` and the template in `templates/post.hbs`. You are not limited to automatic resolution so if your application needs to explicitly include a module, it's only an `import` away. Learn more about ES2015 modules here.

Testing using the CLI

All apps built with Ember CLI are preconfigured to use QUnit and Ember QUnit. While these are the default, you are free to use other options such as Mocha and Ember Mocha.

Dependency Management

Ember CLI uses two package managers: Bower, for keeping front-end dependencies (including jQuery, Ember, and QUnit) up-to-date, and npm, for managing internal dependencies. You can use both package managers to introduce your own dependencies.

Runtime Configuration

Ember CLI's runtime is configurable via a file named `.ember-cli`. The JSON-formatted file, which must be placed in your home directory, can include any command-line options whose names must be in camel case form. For example:

```
# ~/.ember-cli
{
  "skipGit" : true,
  "port" : 999,
  "host" : "0.1.0.1",
  "liveReload" : true,
  "environment" : "mock-development",
  "checkForUpdates" : false
}
```

Content Security Policy

Ember CLI comes bundled with the ember-cli-content-security-policyaddon which when running the development server, enables Content Security Policy in modern browsers. When enabled, Content Security Policy mitigates certain types of attacks including Cross Site Scripting (XSS) and data injection. While browser support is not yet universal, Ember CLI makes it easy to build your app with CSP in mind. For example, enabling it on your production stack is as simple as adding these headers.

Community

Ember CLI is continuously evolving. It’s an on-going community effort. We welcome your issues and PRs for features, bug fixes, and anything that would improve your quality of life as an Ember developer.

Talk to us here:

- Slack: Get your invite
- IRC: #ember-cli on freenode
- Issues: ember-cli/issues

Node

Currently, Ember CLI supports Node (4.0 recommended) and npm (2.x).

WHY?

A good browser framework is important. But building ambitious web applications also requires excellent tooling and processes. Ember CLI wraps all of this (and much more) into a developer-friendly command-line package.

Among the principal features of Ember CLI are:

- Project/Addon creation: create new projects quickly without having to worry about project structure;
- Build pipeline: asset compilation, finger-printing and more out of the box;
- Generators: use the built-in generators and get files that follow the latest practices, and matching tests;
- Ember Addons: extend both your application and Ember CLI itself with community solutions.

By sharing the tooling infrastructure that Ember CLI and its addons provide, members of the Ember.js ecosystem can spend more time on the things that make their individual applications unique.