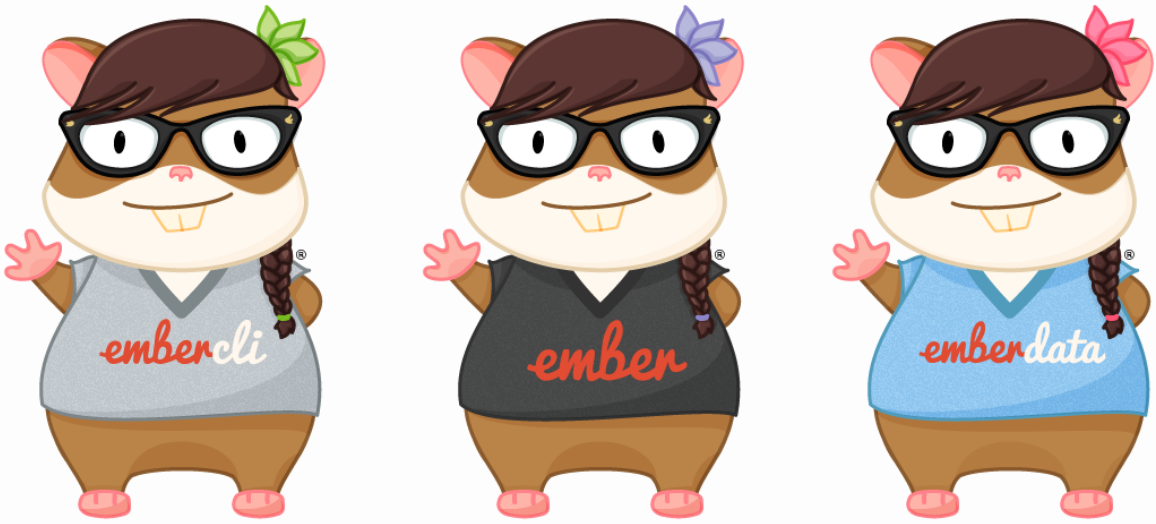
embercli
ember
emberdata

STEPS TO DESIGN SAMPLE EMBER APPLICATION

Prerequisites:

- NodeJs and NPM commands.
- MVC Design Pattern.
- JavaScript, CSS, HTML5 and JSON.
- ES6 and ES5 JavaScript features.
- Jquery basics

Install Ember

You can install Ember with a single command using npm, the Node.js package manager. Type this into your terminal:

```
1  npm install -g ember-cli@2.12
```

Creating a New App

To create a new project using Ember CLI, use the new command. In preparation for the tutorial in the next section, you can make an app called super-rentals.

```
1  ember new super-rentals
```

A new project will be created inside your current directory. You can now go to your super-rentals project directory and start working on it.

```
1  cd super-rentals
```

Directory Structure

The new command generates a project structure with the following files and directories:

```
|--app
|--config
|--node_modules
|--public
|--tests
|--vendor

<other files>

bower.json
ember-cli-build.js
package.json
README.md
testem.js
```

Let's take a look at the folders and files Ember CLI generates.

app: This is where folders and files for models, components, routes, templates and styles are stored. The majority of your coding on an Ember project happens in this folder.

bower.json: Bower is a dependency management tool. It can be used to manage front-end plugins and component dependencies (HTML, CSS, JavaScript, etc). All Bower components are installed in the bower_components directory. If we add front-end dependencies, such as Bootstrap, we will see them listed here, and added to thebower_components directory.
config: The config directory contains the environment.js where you can configure settings for your app.
node_modules / package.json: This directory and file are from npm. npm is the package manager for Node.js.
Ember is built with Node and uses a variety of Node.js modules for operation. The package.json file maintains the list of current npm dependencies for the app. Any Ember CLI add-ons you install will also show up here. Packages listed in package.json are installed in the node_modules directory.

public: This directory contains assets such as images and fonts.

vendor: This directory is where front-end dependencies (such as JavaScript or CSS) that are not managed by Bower go.

tests / testem.js: Automated tests for our app go in the `tests` folder, and Ember CLI's test runner **testem** is configured in `testem.js`.

ember-cli-build.js: This file describes how Ember CLI should build our app.

ES6 Modules

If you take a look at `app/router.js`, you'll notice some syntax that may be unfamiliar to you.

app/router.js

```
1  import Ember from 'ember';
2  import config from './config/environment';
3
4  const Router = Ember.Router.extend({
5    location: config.locationType,
6    rootURL: config.rootURL
7  });
8
9  Router.map(function() {
10 });
11
12 export default Router;
```

Ember CLI uses ECMAScript 2015 (ES2015 for short or previously known as ES6) modules to organize application code.
For example, the line `import Ember from 'ember';` gives us access to the actual Ember.js library as the variable `Ember`. And the `import config from './config/environment';` line gives us access to our app's configuration data as the variable `config`. `const` is a way to declare a read-only variable to make sure it is not

accidentally reassigned elsewhere. At the end of the file, `export default Router;` makes the `Router` variable defined in this file available to other parts of the app.

The Development Server

Once we have a new project in place, we can confirm everything is working by starting the Ember development server:

```
1 ember server
```

or, for short:

```
1 ember s
```

If we navigate to http://localhost:4200, we'll see the default welcome screen.

Building For Production

Now that we've written our application and verified that it works in development, it's time to get it ready to deploy to our users. To do so, run the following command:

```
1 ember build --env production
```

The `build` command packages up all of the assets that make up your application—JavaScript, templates, CSS, web fonts, images, and more.
In this case, we told Ember to build for the production environment via the `--env` flag. This creates an optimized bundle that's ready to upload to your web host. Once the build finishes, you'll find all of the concatenated and minified assets in your application's `dist/`directory.

The Ember community values collaboration and building common tools that everyone relies on. If you're interested in deploying your app to production in a fast and reliable way, check out the Ember CLI Deploy addon.

**********END**********