



## Ember 2

### Section 2 - Quick Start

MVVM Design Pattern  
Ember With ES6  
Installing Ember  
Creating a Single Page Web Application



## Section Objectives

- Be able to teach what MVVM is and the responsibilities of each concern
- Be able to write ES6 syntax and recognize it in Ember code
- Be able to install Ember CLI
- Be able to create a simple single-page application using Ember



## Quick Start

## MVVM Design Pattern

**MVVM Design Pattern**

Ember With ES6

Installing Ember

Core Concepts of an Ember Application

Creating a Single Page Web Application



# MVVM Design Pattern

- Model-View-Controller (MVC)
- Model-View-ViewModel (MVVM)
  - Originally defined by Microsoft for use with Windows Presentation Foundation (WPF) and Silverlight
  - Grew in popularity with single-page apps to not only separate concerns on an architectural level but skill-set level as well
  - ViewModel is the “Controller” + the added benefit of data binding
- Difference between Controllers and Components?
  - Controller’s model is defined by its route
  - Component’s model is injected into it and not tied to a route



## Quick Start

## Ember With ES6

MVVM Design Pattern

**Ember With ES6**

Installing Ember

Core Concepts of an Ember Application

Creating a Single Page Web Application



# Ember with ES6(ES2015)

```
// let keyword
let age = 21;

// arrow functions (lexical ‘this’)
function addListener('click', () =>
  console.log('clicked')
);

// import
import Ember from 'ember';

// class
class Person { constructor() {} }

// method shorthand
var obj = { counter: 1, count() {} }
```



## Quick Start

## Installing Ember

MVVM Design Pattern

Ember With ES6

**Installing Ember**

Core Concepts of an Ember Application

Creating a Single Page Web Application



# Installing Ember

## ■ Node.js and npm

```
npm install -g ember-cli@2.11
```



## Quick Start

### Core Concepts of an Ember Application

MVVM Design Pattern  
Ember With ES6  
Installing Ember

**Core Concepts of an Ember Application**  
Creating a Single Page Web Application



# Core Concepts of an Ember Application

- Router
- Route Handlers
- Templates
- Models
- Components
- Controllers



## Quick Start

### Core Concepts of an Ember Application

MVVM Design Pattern  
Ember With ES6  
Installing Ember

Core Concepts of an Ember Application  
**Creating a Single Page Web Application**



# Directory Structure

■ *ember new notes-app*

```
▶ app
  ▶ components
  ▶ controllers
  ▶ helpers
  ▶ models
  ▶ routes
  ▶ styles
  ▶ templates
    app.js
    index.html
    resolver.js
    router.js

  ▶ app
  ▶ config
  ▶ node_modules
  ▶ public
  ▶ tests
  ▶ vendor
    .bowerrc
    .editorconfig
    .ember-cli
    .gitignore
    .jshintrc
    .travis.yml
    .watchmanconfig
    bower.json
    ember-cli-build.js
    package.json
    README.md
    testem.js
```



# Starting the Development Server

- *ember server (ember s)*
- *http://localhost:4200*



## Congratulations, you made it!

You've officially spun up your very first Ember app :-)

You've got one more decision to make: what do you want to do next? We'd suggest one of the following to help you get going:

- Quick Start - a quick introduction to how Ember works. Learn about defining your first route, writing a UI component and deploying your application.
- Ember Guides - this is our more thorough, hands-on intro to Ember. Your crash course in Ember philosophy, background and some in-depth discussion of how things work (and why they work the way they do).

If you run into problems, you can check Stack Overflow or our forums for ideas and answers—someone's probably been through the same thing and already posted an answer. If not, you can post your **own** question. People love to help new Ember developers get started, and our community is incredibly supportive

*To remove this welcome message, remove the `{{welcome-page}}` component from your `application.hbs` file.*

*You'll see this page update soon after!*



# Setting Up Your Development Environment

- Install the following extensions:
  - Angular 2 TypeScript Emmet
  - Ember Colorizer and Theme
- Format – Alt+Shift+F
- Show Integrated Terminal
- Copy 'app.css' from /Labs/setup/quickstart into the app/styles folder.
- Install Font-Awesome add-on:  
***ember install ember-font-awesome***



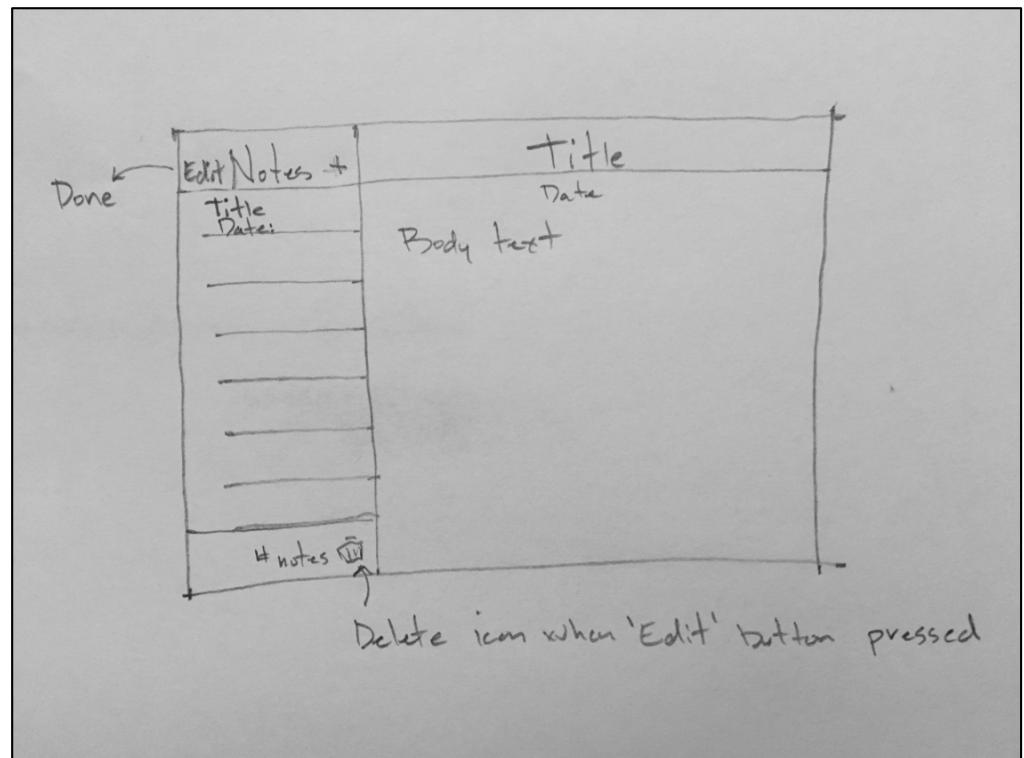
# Setting Up Tests

## Design:

- Always be sure to plan before you code

## Requirements:

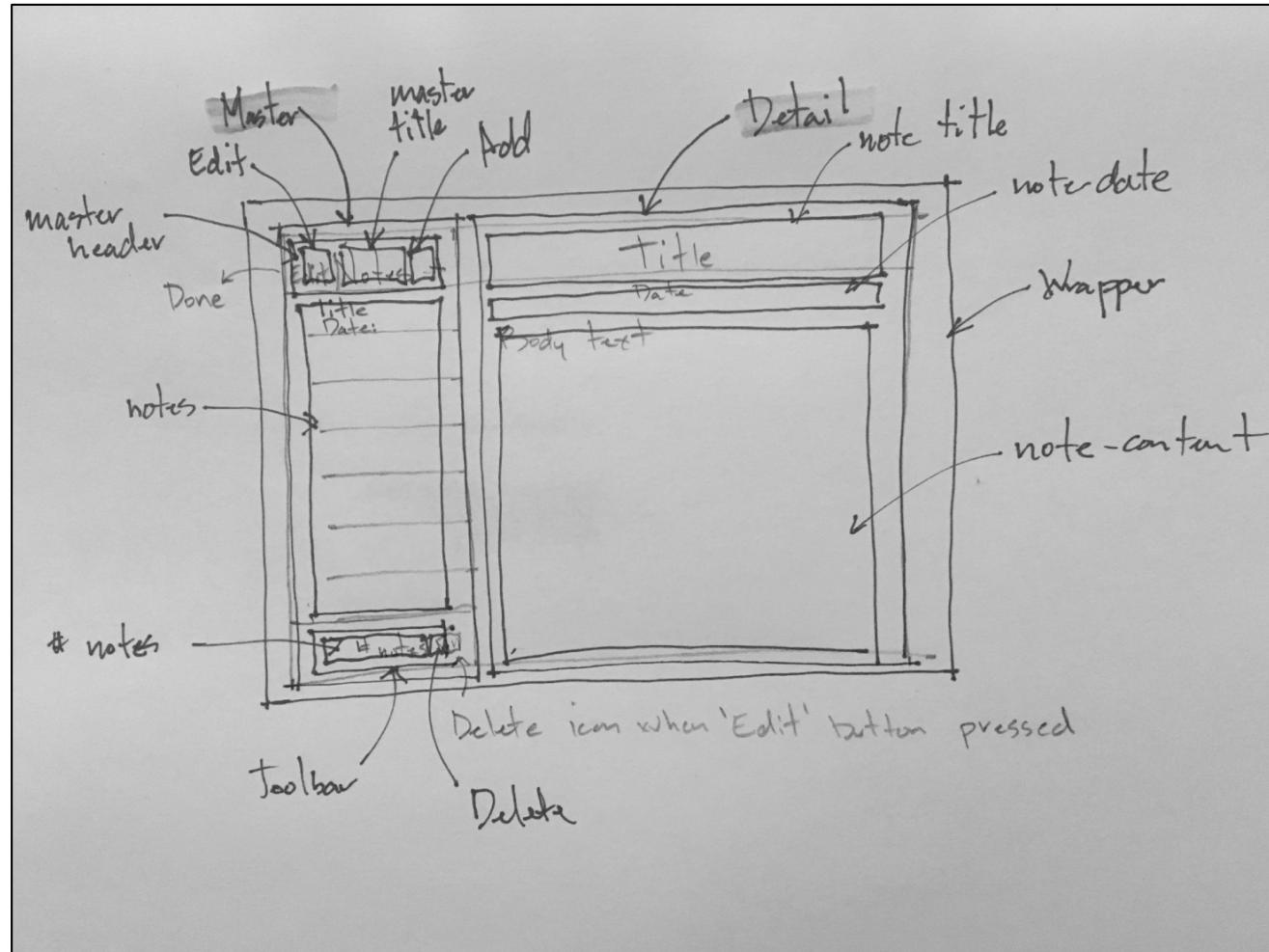
1. Should be able to create a note.
2. Should be able to edit a note.
3. Should be able to delete a note.





# Route - Application

*ember g route application*





# Template - Application

application.hbs

```
<div class="wrapper">
  <div class="master container">
    <div class="master_header">
      <div class="container">&nbsp;</div>
      <div class="master_title container">Notes</div>
      <div class="edit_button container btn">Edit</div>
    </div>
    <div class="notes">
    </div>
    <div class="toolbar">
      <div class="container">&nbsp;</div>
      <div class="notes_count container">{{count}} notes</div>
      <div class="add_button container btn">{{fa-icon "plus-square-o"}}</div>
    </div>
  </div>
  <div class="detail container">
    <div class="detail_header">{{title}}</div>
    {{outlet}}
  </div>
</div>
```



# Model Hook - Application

```
import Ember from 'ember';

export default Ember.Route.extend({
  model() {
    var notes = [];
    var singleNote = notes.length == 1;

    return {
      notes: notes,
      singleNote: singleNote,
      selectedNote: null
    };
  }
});
```

application.js

- Use the ‘notes’ property returned with the model in your template to show the number of notes in the toolbar
- Change detail-header title to use {{model.selectedNote.title}}



## #if Block Helper

- Our label in the toolbar needs to read ‘note’ if there is one note, and ‘notes’ for all other values

```
 {{#if model.singleNote}}  
   note  
 {{else}}  
   notes  
 {{/if}}
```

application.hbs

- Test between one note and two notes



# Ember Data- Index

- Ember Store

- *ember g model Note*

application.hbs

```
var notes = this.get('store').findAll('note');
```

- What happened?

- Change the method call from the Store to ‘peekAll’



# Model - Note

note.js

```
export default DS.Model.extend({
  title: DS.attr('string'),
  content: DS.attr('string'),
  date: DS.attr('date'),
  selected: DS.attr('boolean', {
    defaultValue() {
      return false;
    }
  })
});
```



# Route - Note

## ■ Create a note route

note.js

```
model(params) {  
    return this.get('store').peekRecord('note', params.id);  
}
```



# Template - Note

note.hbs

```
<div class="date">{{model.date}}</div>
<div class="note">
    {{#if model.selected}}
        {{textarea value=model.content}}
    {{/if}}
</div>
```



# Controller - Application

- How do we add functionality to our route? How did we provide the model?
- ***ember g controller application***
- Create an action to add a note
  - Add {{action “addNote”}} to add\_button
  - Create addNote()

application.js

```
actions: {
  addNote() {
    let notes = this.get('store').peekAll('note');
    this.set('model.singleNote', notes.content.length === 0);
    this.get('store').push({
      data: [
        id: notes.content.length,
        type: 'note',
        attributes: {
          title: "New Note",
          date: new Date().toUTCString(),
          content: "",
          selected: false
        }
      ]
    });
  }
}
```



# Component – ‘note-row’

## ■ *ember g note-row*

## ■ How does the component work in the Ember architecture?

- Model
- Functionality
- Template

## ■ *{{note-row note=note showNote=(action “showNote”)}}}*

## ■ Component code

```
import Ember from 'ember';  
  
export default Ember.Component.extend({  
  tagName: 'div',  
  classNames: ['note_row'],  
  note: null,  
  
  actions: {  
    selectNote() {  
  
    }  
  }  
});
```

note-row.js



## #each helper

- Now we need to be able to display multiple {{note-row}} components

```
{ {#each model.notes as |note| } }  
  {{note-row note=note showNote=(action "showNote") }}  
{ {/each} }
```

application.hbs



## #link-to helper

- We are now ready to be able to select a note and have it show up in our detail view
- #link-to allows us to go to a route specified in the router

```
application.hbs  
{{{#each model.notes as |note|}}}  
  {{{#link-to 'note' note}}}  
    {{note-row note=note showNote=(action "showNote")}}  
  {{{/link-to}}}  
{{{/each}}}
```



# #unless helper

- Similar to a guard statement

application.hbs

```
<div class="detail_content">
  {{#unless model.selectedNote}}
    <div class="default_note">
      +<br/>Note
    </div>
  {{/unless}}
  {{outlet}}
</div>
```



# Input Helpers

- We need a way to edit the title and contents of the note in-place

```
<div class="detail_header">  
    {{#if model.selectedNote}}  
        {{input type="text" value=model.selectedNote.title}}  
    {{/if}}  
</div>
```

application.hbs

```
<div class="note">  
    {{#if model.selected}}  
        {{textarea value=model.content}}  
    {{/if}}  
</div>
```

note.hbs



## Section Review

- Be able to teach what MVVM is and the responsibilities of each concern
- Be able to write ES6 syntax and recognize it in Ember code
- Be able to install Ember CLI
- Be able to create a simple single-page application using Ember



# Notes App

Edit	Notes	+	Random Note
	<b>Random Note</b> Fri, 03 Mar 2017 23:29:10 GMT		Fri, 03 Mar 2017 23:29:10 GMT
	<b>New Note</b> Fri, 03 Mar 2017 23:29:11 GMT		A note that I wrote down
	<b>New Note</b> Fri, 03 Mar 2017 23:29:11 GMT		
	<b>New Note</b> Fri, 03 Mar 2017 23:29:12 GMT		
4 notes			



# End of Section