

# BUILDING WEB APPS WITH EMBER.JS

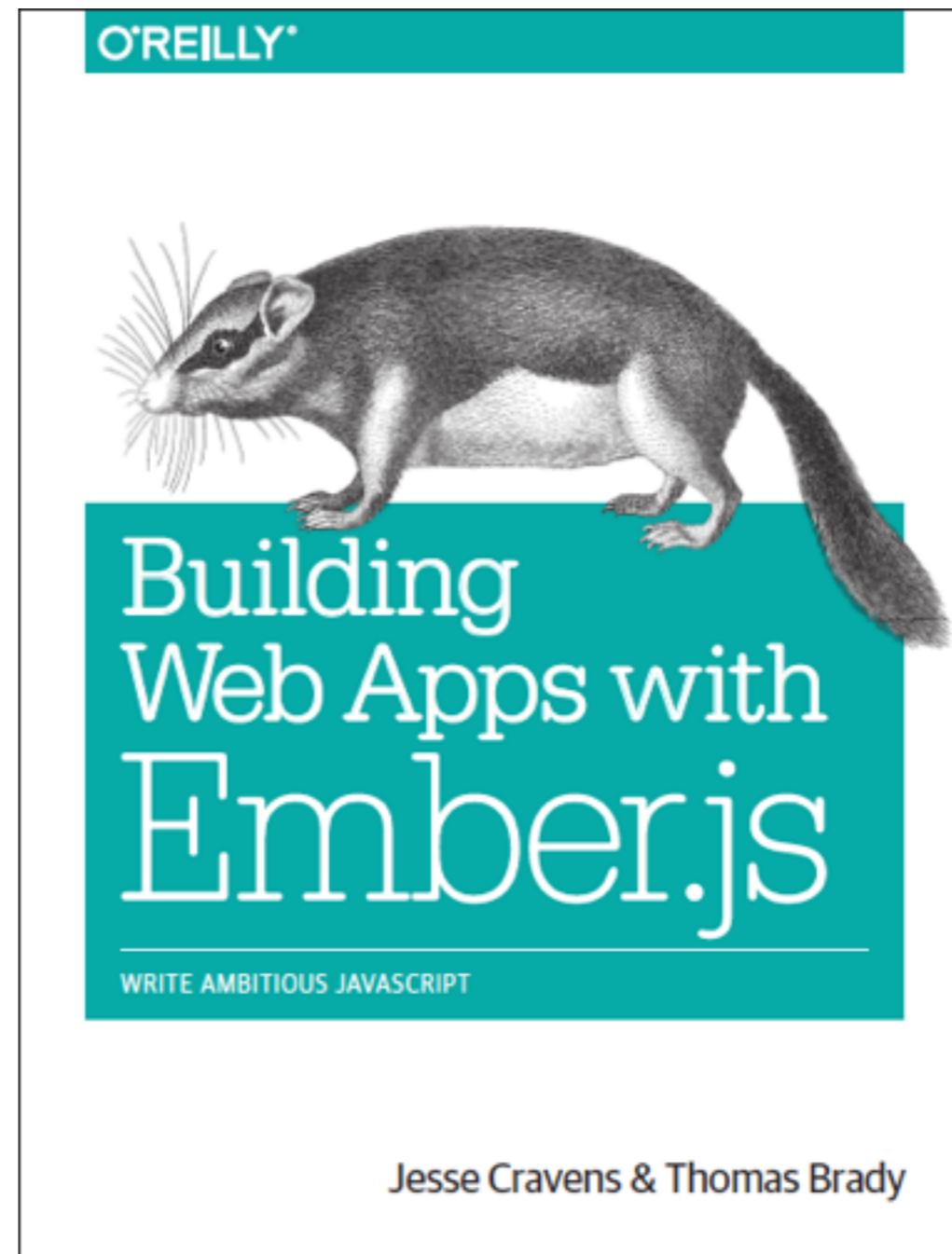
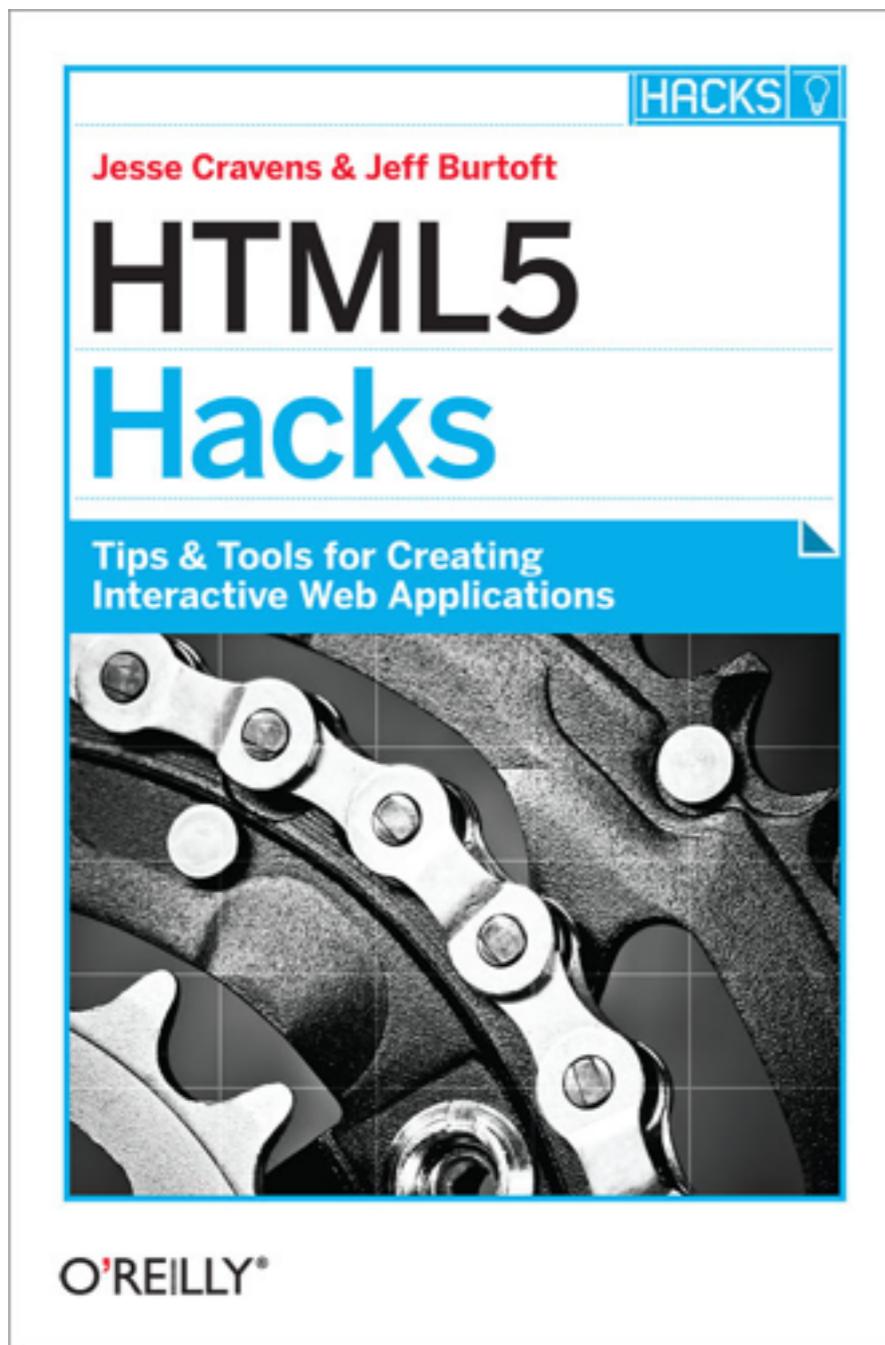
UX DEVELOPMENT at frog



**@jdcravens  
github.com/jessecravens  
jessecravens.com  
principal web engineer | frog Austin**



# O'REILLY MEDIA

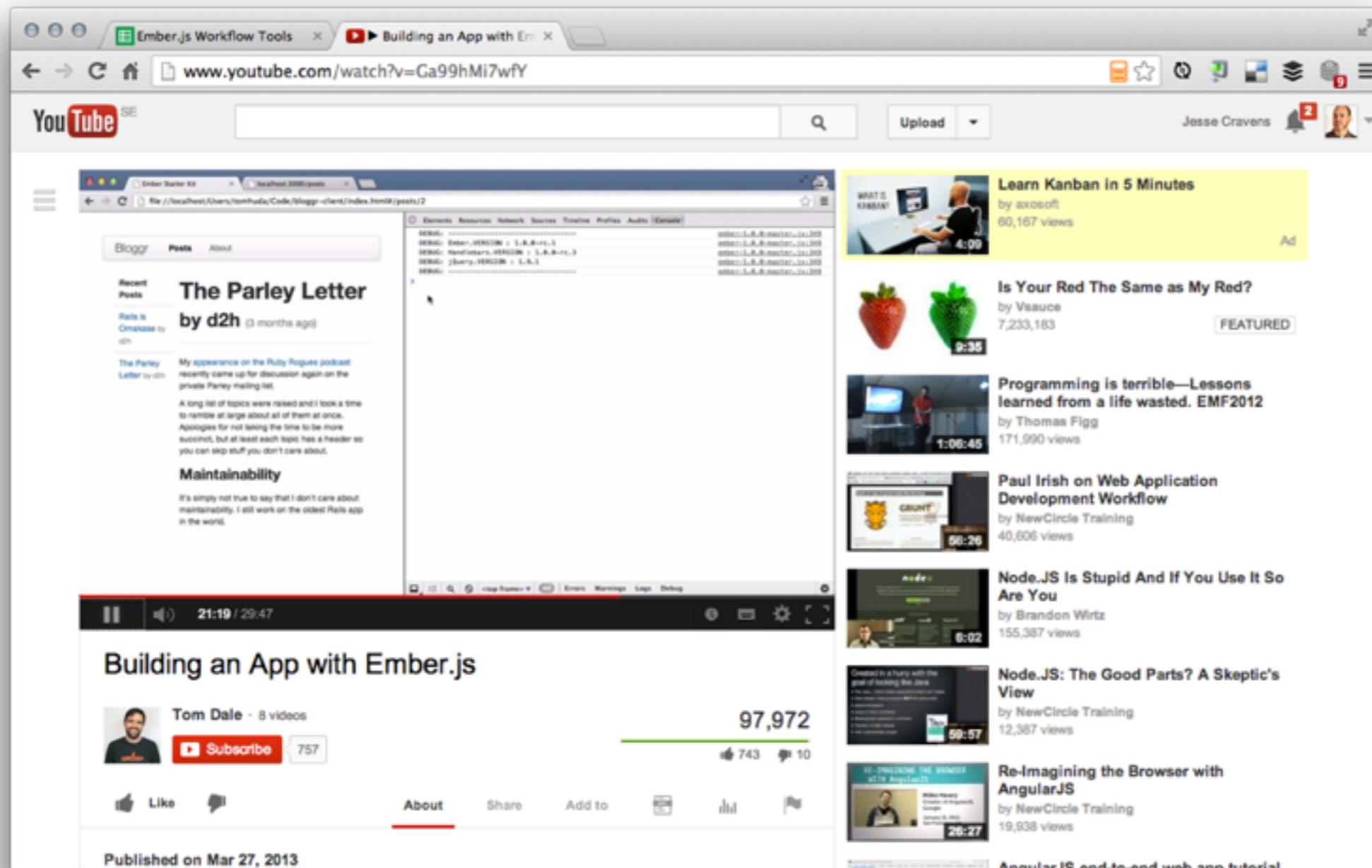


# GETTING STARTED

## chapter 1



# BLOGGR-CLIENT



# EMBER GUIDES

## TODO APP

The screenshot shows a web browser displaying the Ember.js Guides website at [emberjs.com/guides/](http://emberjs.com/guides/). The page has a red header with the Ember logo and navigation links for ABOUT, GUIDES (which is active), API, COMMUNITY, BLOG, and BUILDS. A search bar and a 'FORK US!' button are also in the header. The main content area is titled 'EMBER.JS GUIDES' and includes a welcome message about the documentation's purpose and a link to a 30-minute screencast. Below the welcome message is a video player showing a screencast titled 'Building an Ember.js Application'. The video player shows a code editor with Ember.js code and a progress bar at 00:00 / 27:29. A YouTube logo is visible in the bottom right corner of the video player.

Welcome to the Ember.js guides! This documentation will take you from total beginner to Ember expert. It is designed to start from the basics, and slowly increase to more sophisticated concepts until you know everything there is to know about building awesome web applications.

To help you get started, we've also made a 30-minute screencast that will guide you through building a full-featured Ember.js application:

**Building an Ember.js Application**

```
App = Ember.create();
App.Router.map(function() {
  this.resource('about');
  this.resource('posts');
});

var posts = [
  {
    id: '1',
    title: "Rails Is Deadass",
    author: { name: "d3n" },
    date: new Date('12-27-2012'),
    excerpt: "There are lots of à la carte software environments in this world. Places where you can or want this for my ORM, I want this for my template language, and let's finish it off with",
    body: "I want this for my ORM, I want this for my template language, and let's finish it off with"
  },
  {
    id: '2',
    title: "The Parley Letter",
    author: { name: "d3n" },
    date: new Date('12-24-2012'),
    excerpt: "My appearance on the Ruby Rogues podcast (http://rbrogues.com/056-rr-david-heinzerling)",
    body: "A long list of topics were raised and I took a time to ramble at large about all of them"
  }
];
```

Source code for the app we build in the video is available at <https://github.com/tildedio/bloggr-client>

Most of these guides are designed to help you start building your first app. If you'd like to know more,



# REPOS | FIDDLES | GISTS



# TECH.PRO

Jesse Cravens blogs - Tech PRO

tech.pro/jdcravens/blog

 Jesse Cravens posted 7 months ago INTERMEDIATE Ember  
**Ember.js Views and Live Templates with Handlebars.js Part 2**  
In my previous tutorial, Ember.js Views and Live Templates with Handlebars.js Part 1, we explored the Handlebars.js template library when used with...

 Jesse Cravens posted 8 months ago INTERMEDIATE Ember  
**Ember.js Views and Live Templates with Handlebars.js Part 1**  
This is an exploration of Handlebars.js template library when used with Ember.js views. Many of the Handlebars.js tutorials on the web discuss the...

 Jesse Cravens posted 9 months ago INTERMEDIATE Yeoman Ember  
**Modern Ember.js Application Workflow with Yeoman and Mocha**  
The following tutorial will provide an overview for building Ember.js applications with Yeoman. Keep in mind, Yeoman is a framework agnostic...

 Jesse Cravens posted 10 months ago BEGINNER Ember  
**Getting Started with Ember.js**  
This is a guide that is intended for JavaScript developers that want to get started



# FIDDLES

The screenshot shows the jsfiddle.net user dashboard interface. On the left, there are three fiddle examples:

- Ember Component**: A screenshot of an Ember component definition. The code includes:

```
App = Ember.Application.create();  
App.TimeOutComponent = Ember.Component.extend({  
  active: false,  
  messageShown: false,  
  inactiveTimeout: 10000, // this should happen after the ...});
```
- Testing Ember.js (BDD with Mocha)**: A screenshot of an Ember.js test using BDD and Mocha. The code includes:

```
App = Ember.Application.create();  
App.IndexRoute = Ember.Route.extend({  
  model: function() {  
    return this.store.find('team');  
  }  
});  
App.TeamAdapter = DS.FixtureAdapter ...
```
- Ember.js Template Handlebars helpers: Partial**: A screenshot of an Ember.js template using Handlebars partials. The code includes:

```
var App = Em.Application.create();  
App.Router.map(function () {  
  this.resource('home');
```

The right side of the dashboard shows the user's profile information: "Edit your profile" and "Sign out". Below that is a profile picture of a man with a mustache and the username "jdcravens".



**WAIT...**



**“SHIP CODE!”**



# 4 THEMES



**ADDRESS THE MORE COMPLEX  
USE CASES**

**WORK FRONT TO BACK**

**DESIGN WITH YOUR MEDIUM**

**CONSIDER PROD EARLY**



# 3 SECTIONS



**I. INTRODUCTION**

**II. PROTOTYPING WITH EMBER**

**III. PRODUCTION EMBER**



# 10 CHAPTERS



**PRELUDE**

# **I. INTRODUCTION**

**GETTING STARTED  
EMBER BASICS**

# **II. PROTOTYPING WITH EMBER**

**EMBER TOOLING, BOILERPLATE, AND WORKFLOW**

**TEMPLATES**

**THE ROUTER, ROUTES AND MODELS**

**CONTROLLERS, VIEWS, DATA-BINDING AND EVENTS**

# **III. PRODUCTION EMBER**

**PERSISTING DATA**

**BUILDING AN EMBER BACKEND**

**EMBER COMPONENTS**

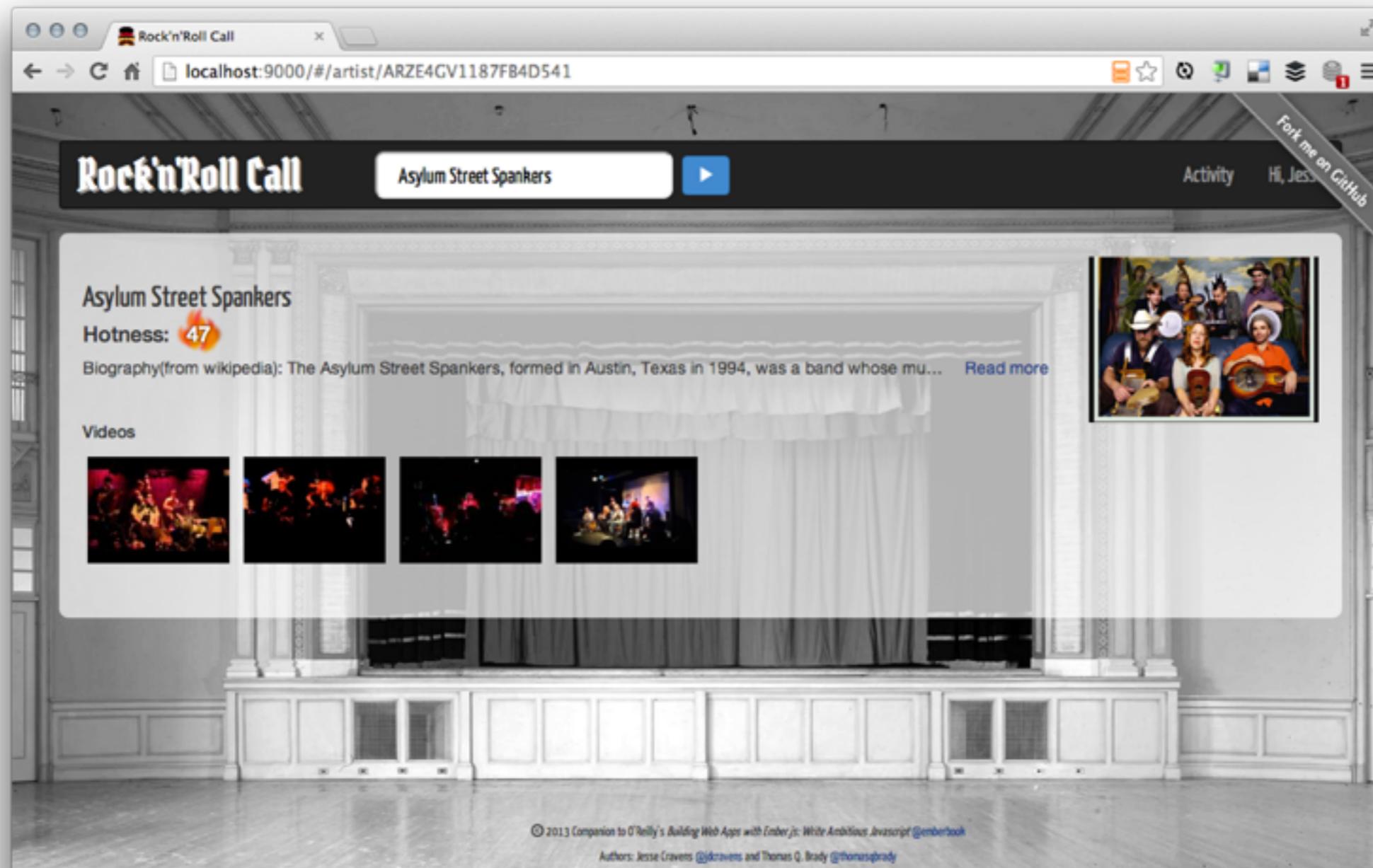
**EMBER TESTING**



**ADDRESS THE  
MORE COMPLEX  
USE CASES**



# ROCKNROLLCALL

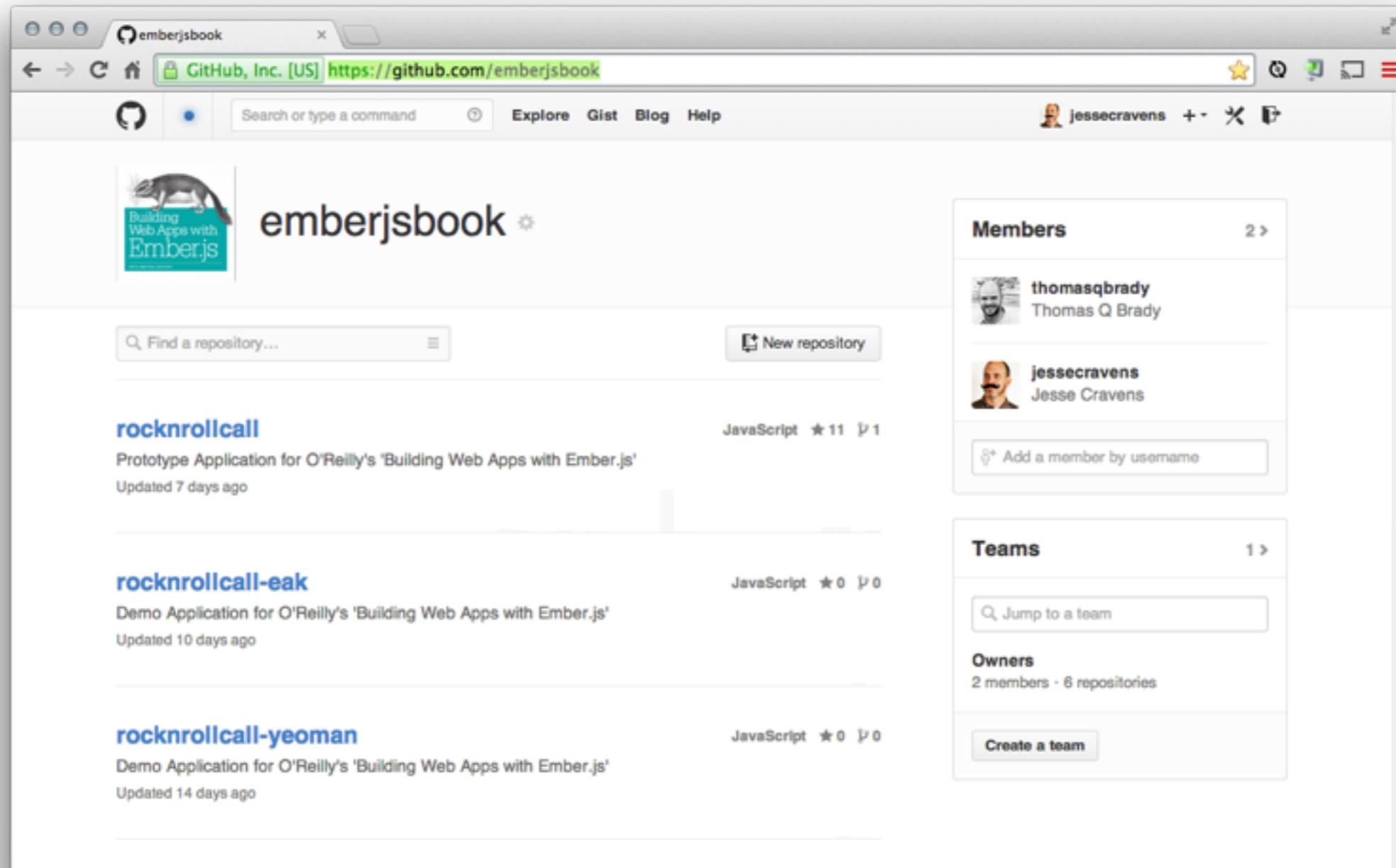


# MOVIN' BEYOND THE BASICS

- \* Ember Tooling
- \* Ember Inspector
- \* Ember Boilerplates / Generators
- \* App Initializers
- \* RSVP Promises
- \* Needs
- \* Ember Components
- \* 3rd Party Integration (jQuery, D3)
- \* Ember-Testing
- \* Ember-Data
- \* JSON Serialization and Format



# COMPANION CODE



**WORK FRONT TO  
BACK**



# RAPID DEV

## 1ST

Early Design: HTML, Diagram Controllers

URL Driven: State Manager and Routes First

Populate the Controllers with Dummy Data

Models, Fixtures, FixtureAdapter

## 2ND

RESTAdapter

Build Server-Side Routes/Endpoints

Serialization and Formatting JSON

Remote Data Store and DB Sync

What is an efficient  
order of execution?

Working front to back  
aids Agile Iterations.

Prototype the  
Router  
and States  
of the Application

By using Dummy data,  
you are able to organize  
the Apps Route Handlers  
and Controllers

Web and Front End  
Devs work in parallel



# RAPID DEV

C  
O  
N  
C  
E  
P  
T  
U  
A  
L

D  
E  
P  
L  
O  
Y

WEB/SERVICES DEVELOPMENT

FRONT END DEVELOPMENT

TEMPLATES

ROUTER

ROUTE HANDLERS

CONTROLLERS

VIEWS/COMPONENTS

MODELS

FIXTURE ADAPTER

REST ADAPTER

SERVICES (API DESIGN)

MVC, SPA (Bootstrap Object)

SERVICES (IMPLEMENTATION)



# EMBER BASICS

## chapter 2



# MINIMAL VIABLE APP

- \* Dummy Data
- \* Bindings
- \* Handlebars View helper
- \* Application Initialization



**CONSIDER PROD  
EARLY**



# AUTOMATION



# WORKFLOW



# EMBER TOOLING

chapter 3



# TOOLS MATRIX

The screenshot shows a Google Sheets document with the title "Ember.js Workflow Tools". The document compares five tools across various features:

	A	B	C	D	E	F	G
Yeoman		Yeoman <a href="https://github.com/yeoman/yeoman">https://github.com/yeoman/yeoman</a> <a href="https://github.com/yeoman/generator-ember">https://github.com/yeoman/generator-ember</a> Anthony Bull	Ember-Tools <a href="https://github.com/rflorenc/ember-tools">https://github.com/rflorenc/ember-tools</a> Ryan Florence	Ember-Rails <a href="https://github.com/emberjs/ember-rails">https://github.com/emberjs/ember-rails</a> Ember Core team	Ember App Kit <a href="https://github.com/stefanpenner/ember-app-kit">https://github.com/stefanpenner/ember-app-kit</a> Stefan Penner		
Ember App Kit				Can be used with Ember-Tools / generator-ember			
Ember Tools			ember g, requires in application.js components generators and directory ember -w		Advanced project setup Uses ES6 Module Transpiler		
Ember Rails				Asset Pipeline		grunt-contrib-concat grunt-contrib-imagemin grunt-contrib-handlebars grunt-contrib-coffee grunt-contrib-less grunt-contrib-sass grunt-contrib-jshint grunt server emberTesting, Karma	
Ember App Kit Rails					Ember, Ember Data and Handlebars, gem "ember-script-rails" gem "active_model_serializers" gem "active_model_serializers"		

Annotations highlight specific features:

- "Advanced project setup" is annotated over the "ember g" row.
- "Uses ES6 Module Transpiler" is annotated over the "ember -w" row.
- "Asset Pipeline" is annotated over the "Asset Pipeline" row.
- "Friendly integration with testem" is annotated over the "ember -w" row.
- "Ember, Ember Data and Handlebars, ..." is annotated over the "Ember App Kit Rails" row.



# EMBER INSPECTOR

The screenshot shows the Ember Inspector tool integrated into a browser window for a "Rock'n'Roll Call" application. The browser address bar shows "localhost:9000/#/". The Ember Inspector tab is active, displaying a table of routes and a preview of the application's view.

**Naming Conventions Table**

**Inspecting Route's properties**

**The View Tree**

**Ember - Data**

**The Application Object**

View Tree	Route Name	Route	Controller	Template	URL
/# Routes	application	ApplicationRoute	ApplicationController	application	
Data	activity	ActivityRoute	ActivityController	activity	/activity
	search-results	SearchResultsRoute	SearchResultsController	search-results	/search/:term
	artist	ArtistRoute	ArtistController	artist	/artist/:enid
	song	SongRoute	SongController	song	/song/:sid
	index	IndexRoute	IndexController	index	/



# PROTOTYPING W/ EMBER

**section II**



# REQs AND EARLY DESIGN



# BASIC REQs

- \* Search by Keyword to Find Transient Data (Songs and Artists)
- \* Persist the Activity of Selecting Songs and Artists into LocalStorage
- \* Display Detail information to the User
- \* Show a Visualization of the ‘Hotness’ of your Searches



# BASIC FLOW

- \* Application Route, application.hbs, view App.SearchBox
- \* Transitions to the SearchResultsRoute passing the query
- \* Calls the searchByName(), searchByTitle on Artists/Songs Controllers
- \* Populates the artists, songs arrays on the SearchResultsController
- \* In search-results template iterate over those arrays
- \* Click on takes the user to the Detail page



# TEMPLATES

## chapter 4

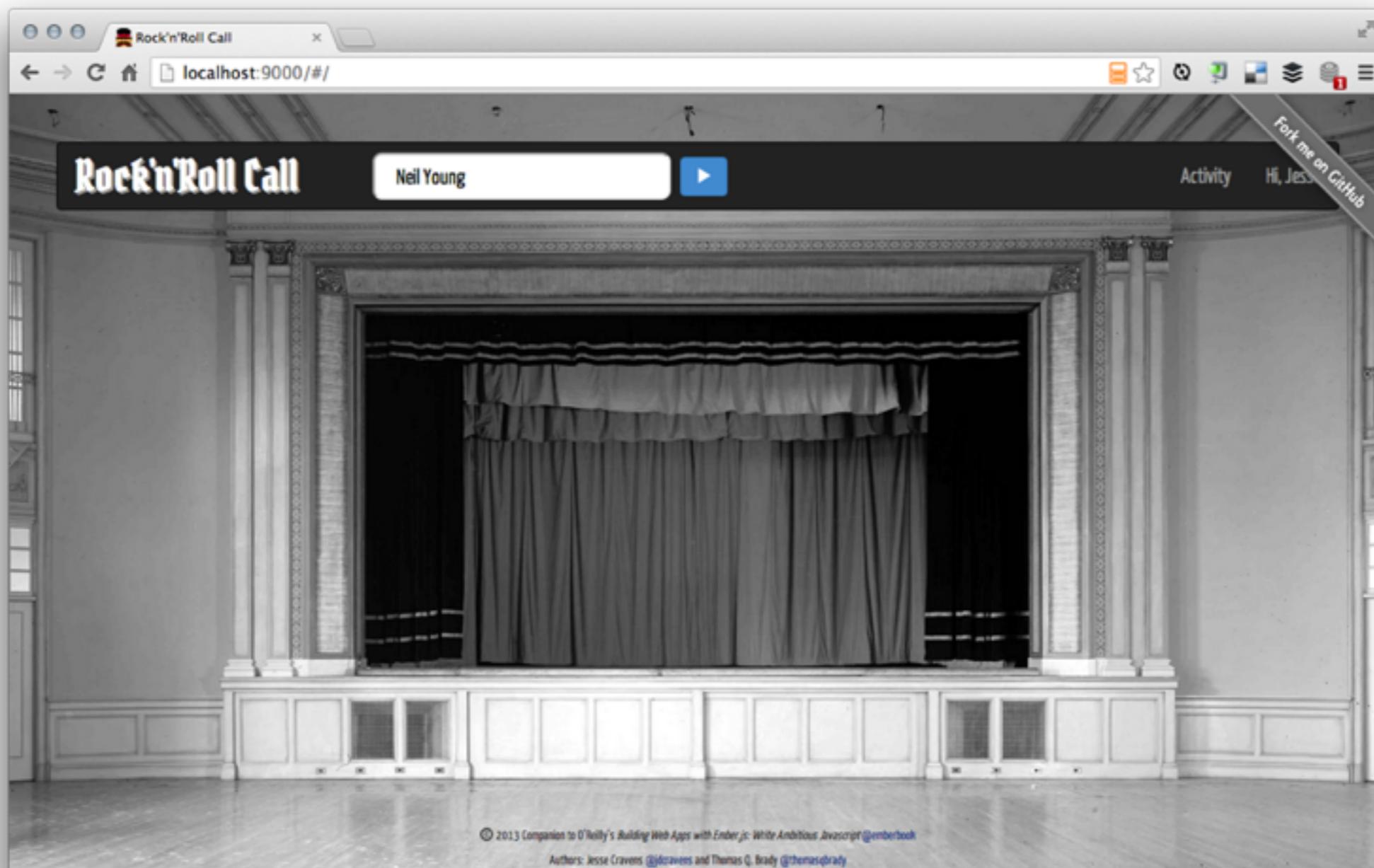


**DESIGN WITH  
YOUR MEDIUM**



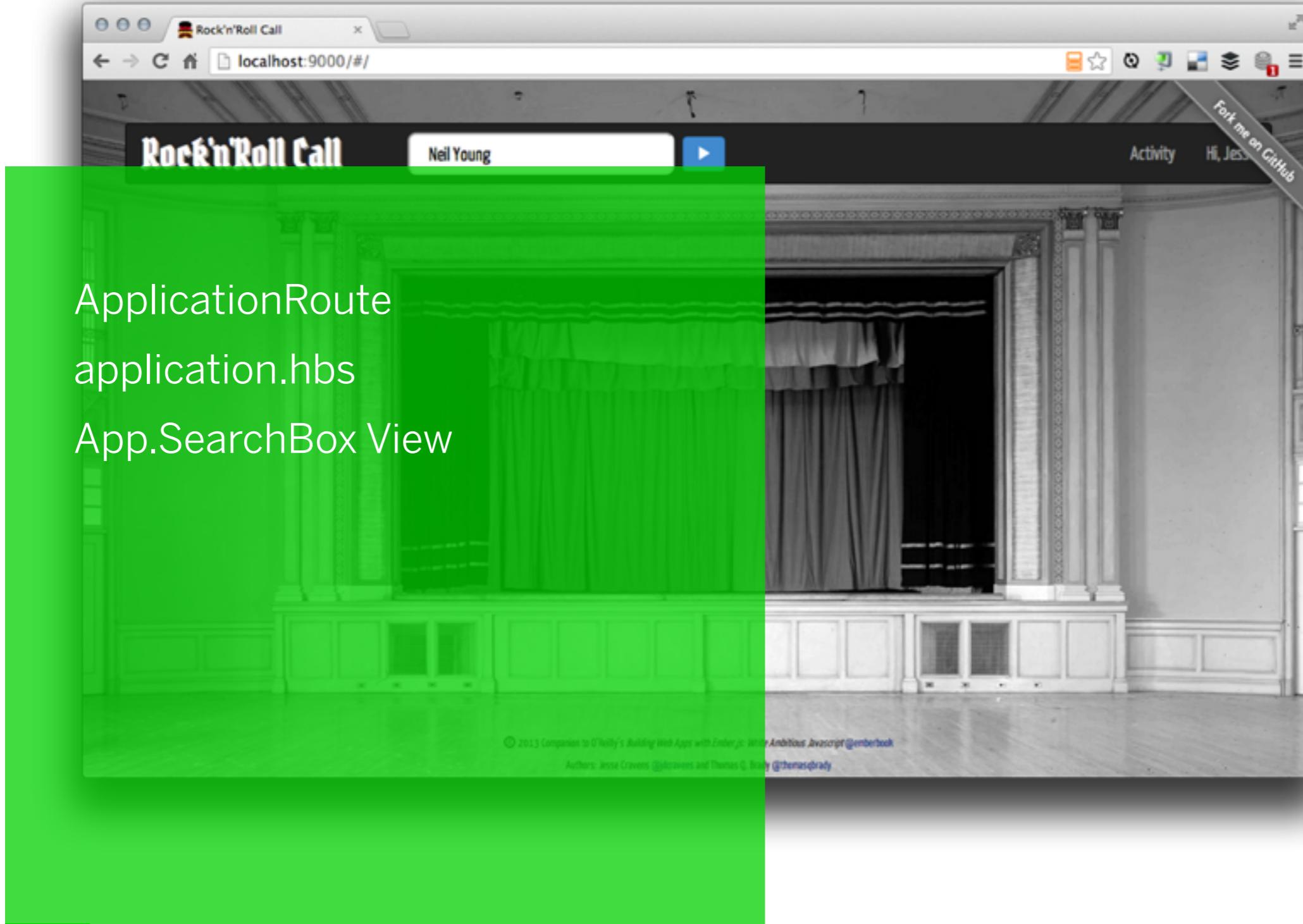
# DESIGN IN .HBS

## APPLICATION / SEARCH STATE



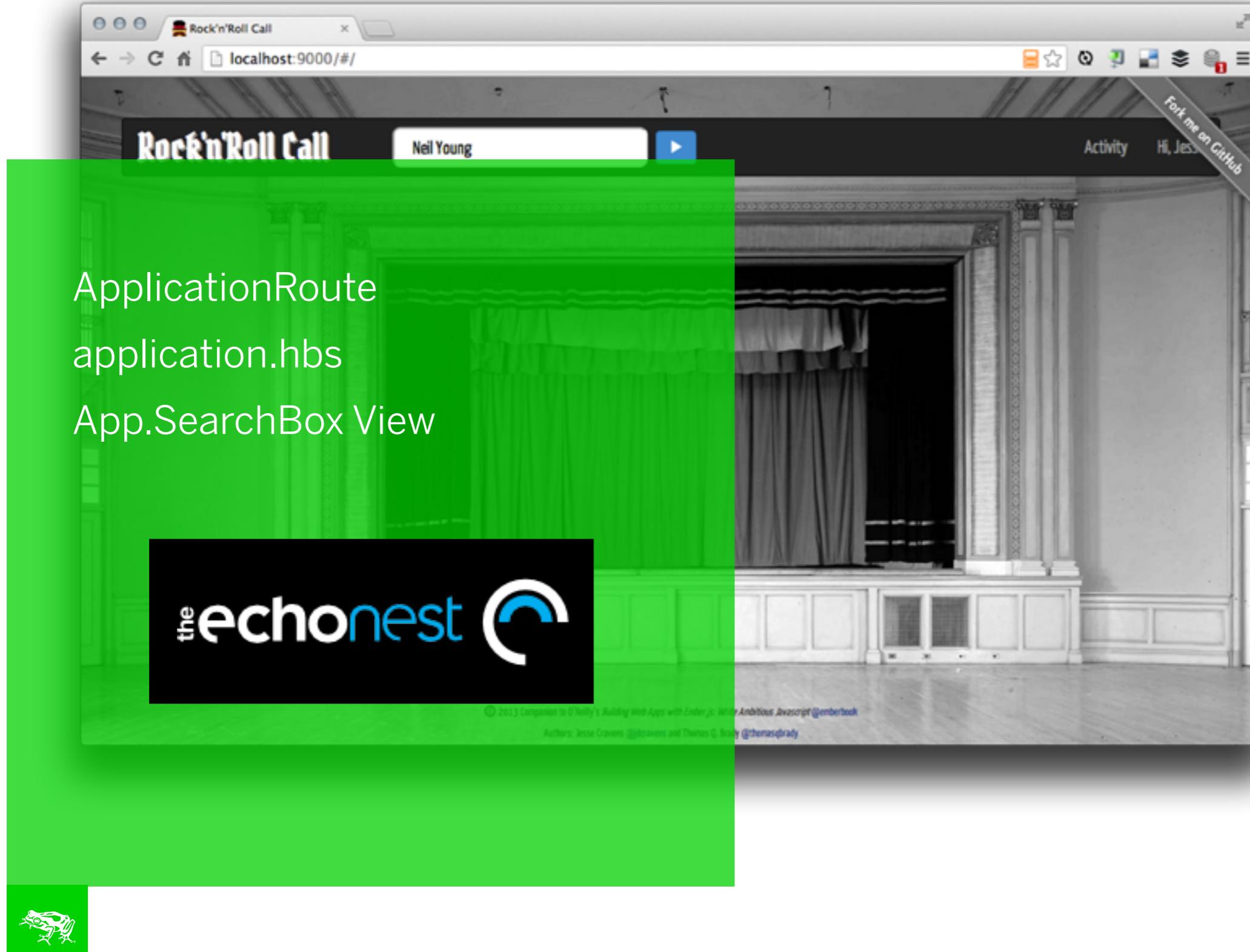
# DESIGN IN .HBS

## APPLICATION / SEARCH STATE



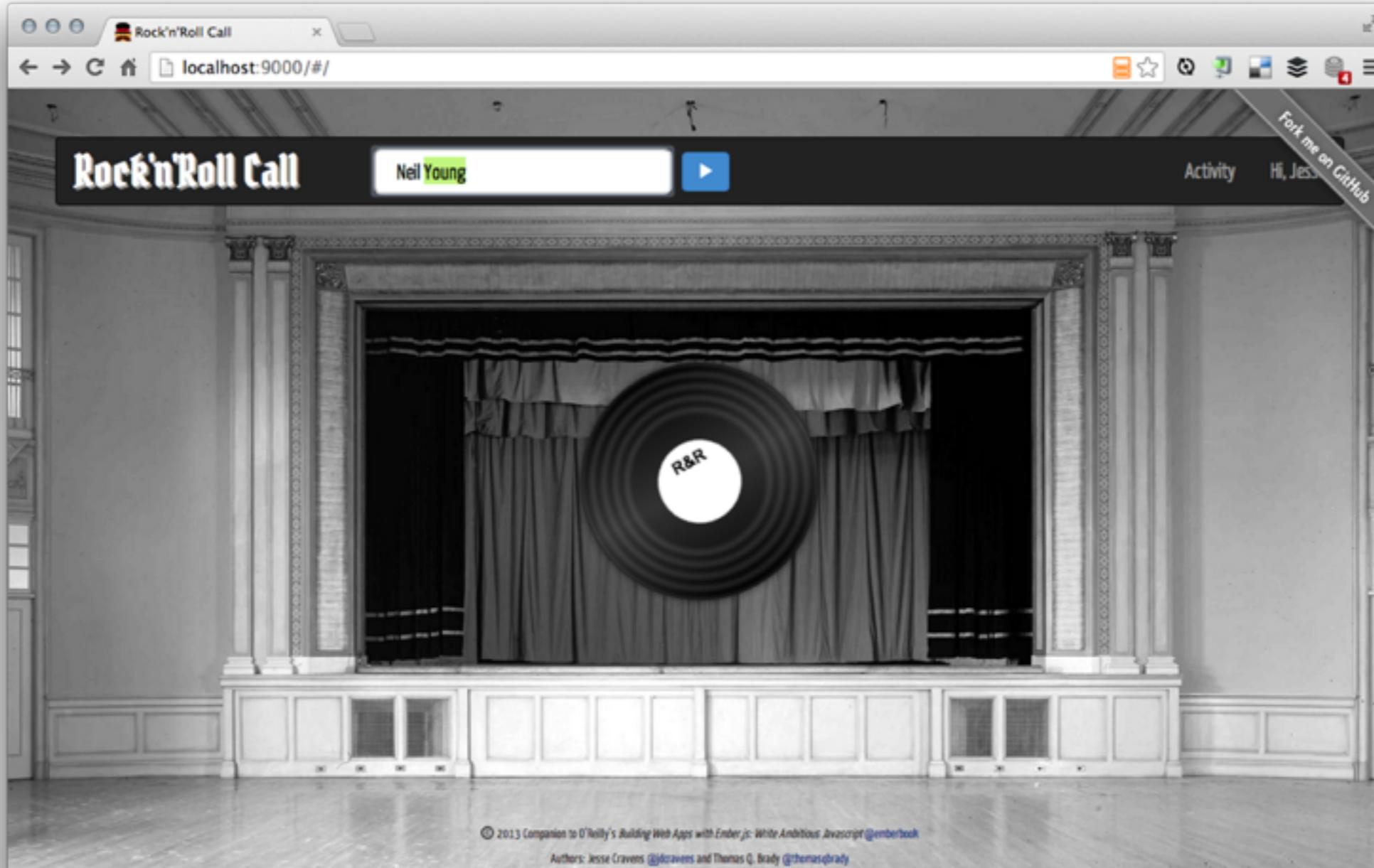
# DESIGN IN .HBS

## MUSIC WEB SERVICE



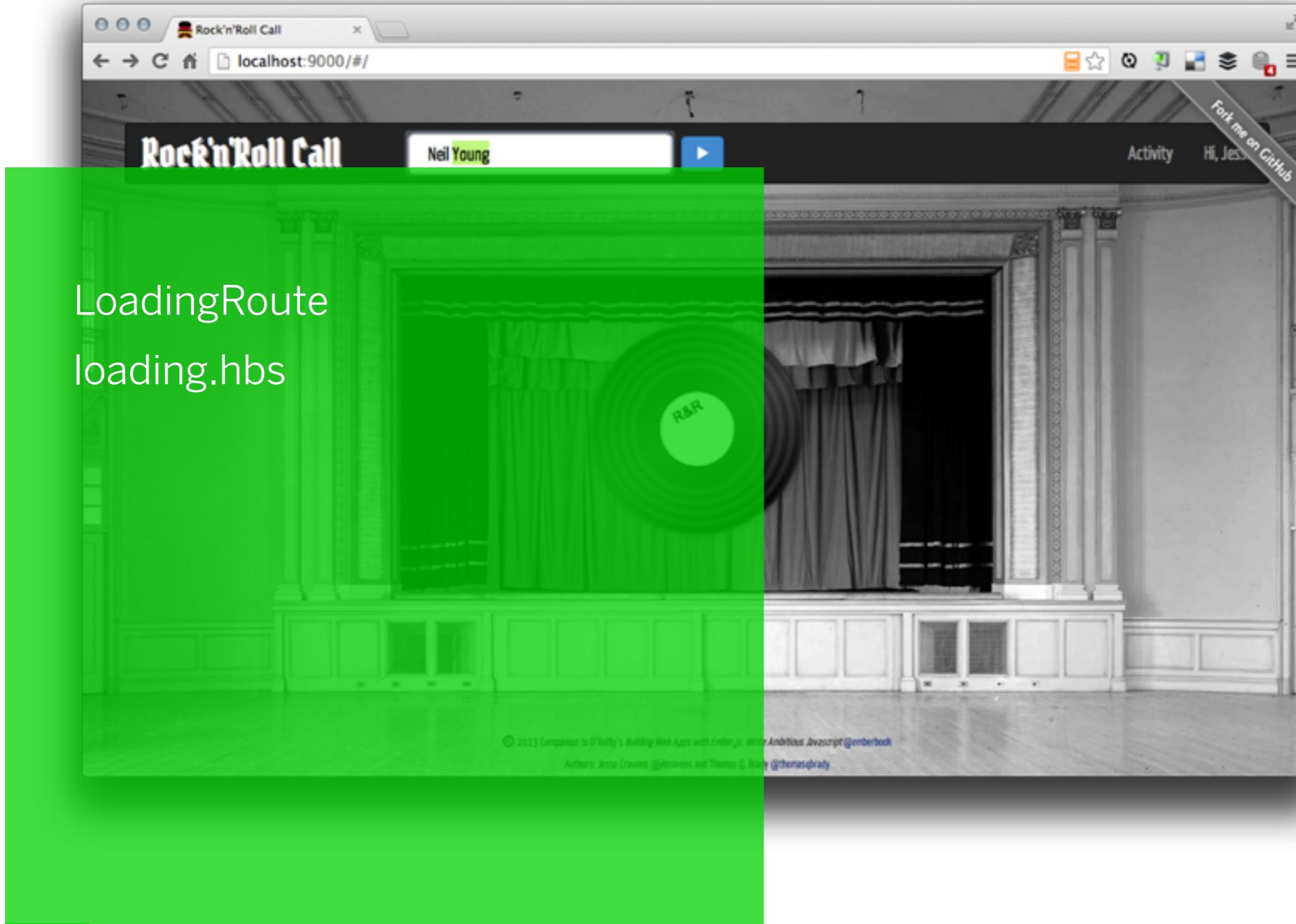
# DESIGN IN .HBS

## LOADING STATE



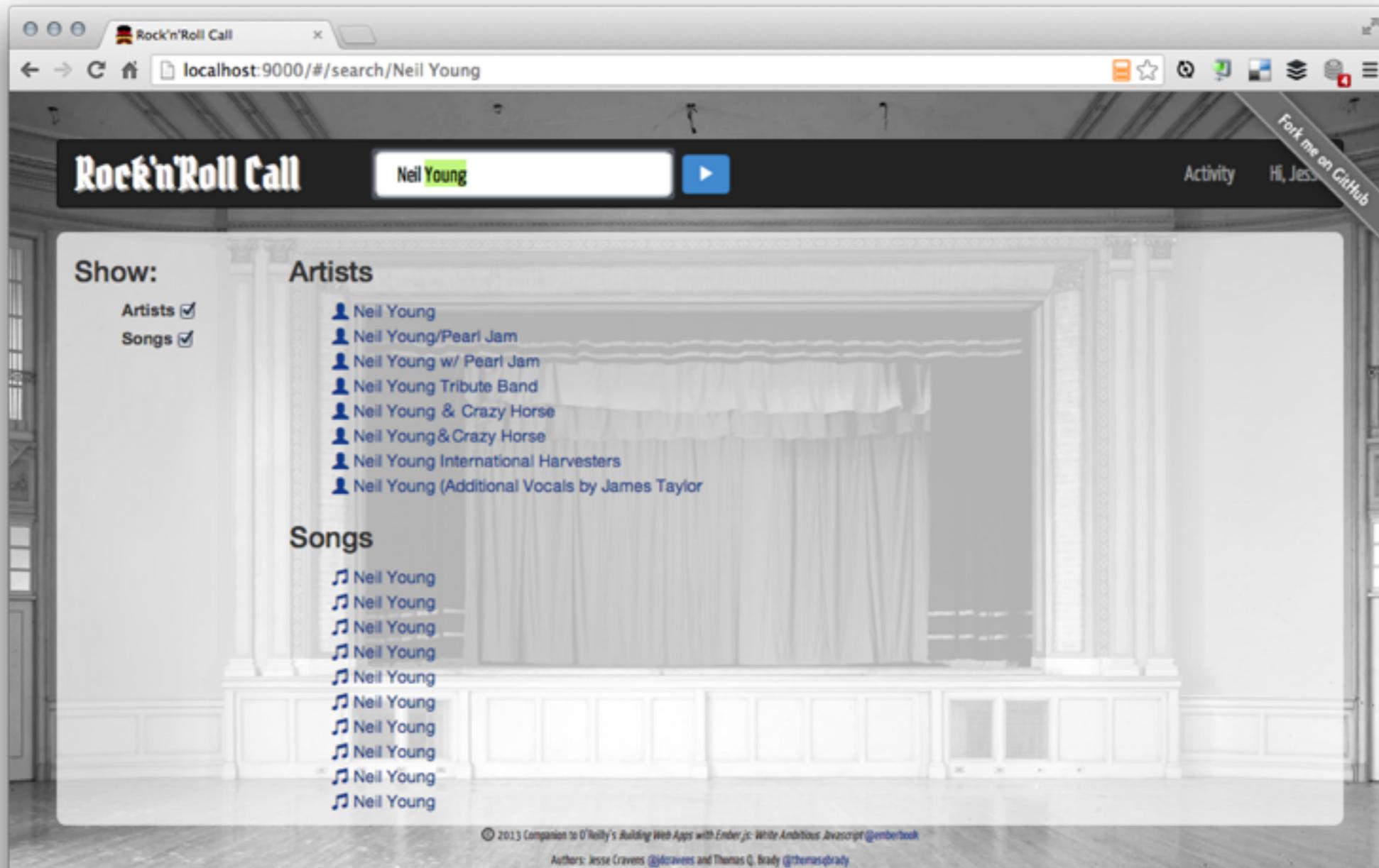
# DESIGN IN .HBS

## LOADING STATE



# DESIGN IN .HBS

## SEARCH RESULTS STATE



# DESIGN IN .HBS

## SEARCH RESULTS STATE

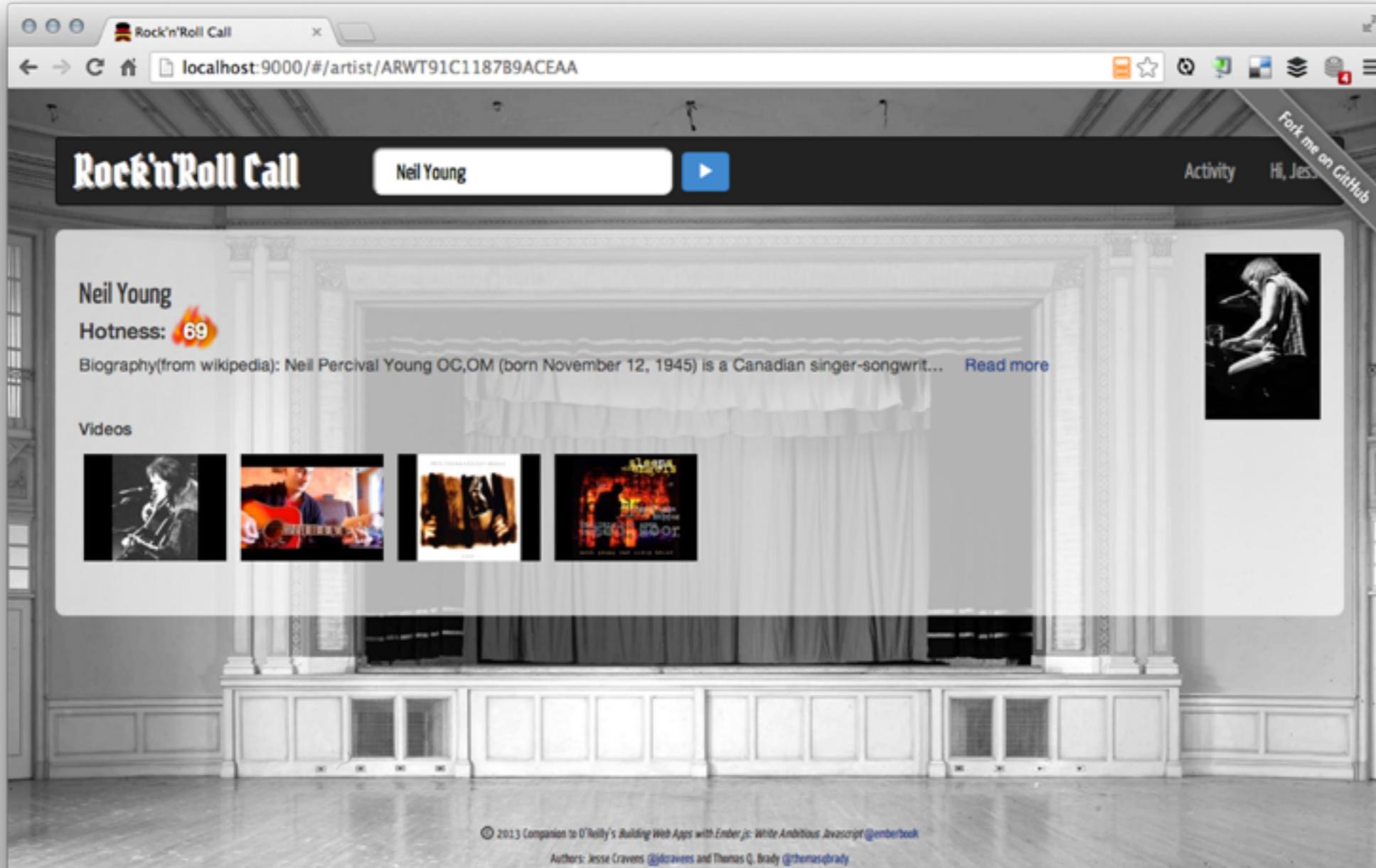
The screenshot shows a web browser window with the title "Rock'n'Roll Call". The address bar displays "localhost:9000/#/search/Neil Young". The main content area is a search results page for the artist "Neil Young". The page has a green header with the title "Rock'n'Roll Call" and a search input field containing "Neil Young". Below the header, there are two sections: "Artists" and "Songs". The "Artists" section lists several entries, each with a small profile picture and the name: Neil Young, Neil Young w/ Pearl Jam, Neil Young Tribute Band, Neil Young & Crazy Horse, Neil Young International Harvesters, and Neil Young (Additional Vocals by James Taylor). The "Songs" section lists ten entries, each with a small musical note icon and the name: Neil Young, and Neil Young. The background of the page is a grayscale photograph of a stage or concert venue with a piano and curtains. In the top right corner of the browser window, there is a GitHub integration interface with the text "Fork me on GitHub", "Activity", and "Hi, Jesse". The bottom of the page contains copyright information: "© 2013 Companion to O'Reilly's Building Web Apps with Ember.js: Write Ambitious JavaScript @emberbook" and "Authors: Jesse Cravens @jcravens and Thomas Q. Brady @thomasqbrady". On the left side of the slide, there is a vertical stack of text labels: "SearchResults Route", "SearchResultsController", "search-results.hbs", "artists()", "songs()", "actions", and "bindings".

SearchResults Route  
SearchResultsController  
search-results.hbs  
artists()  
songs()  
actions  
bindings



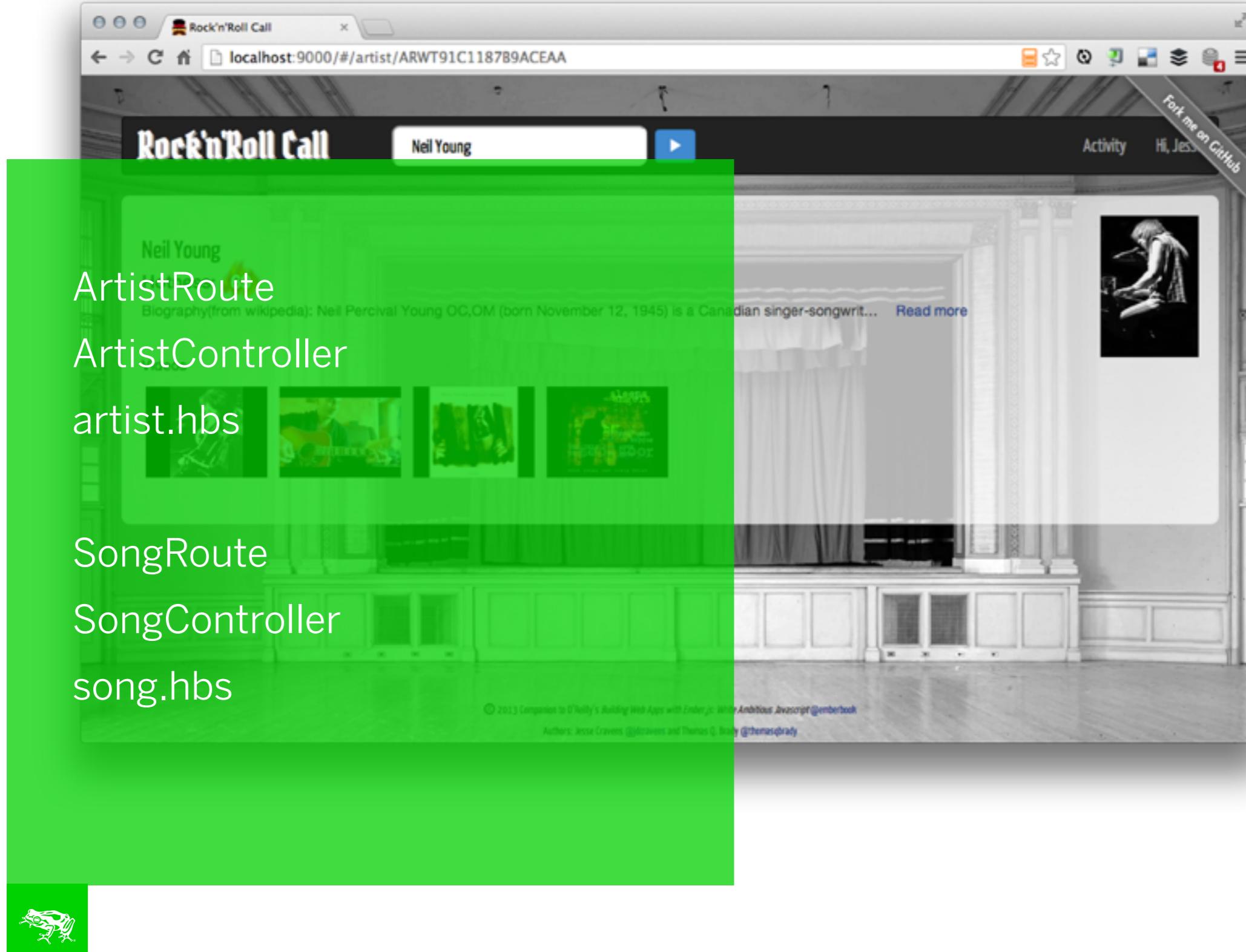
# DESIGN IN .HBS

## ARTIST/SONG DETAIL STATE



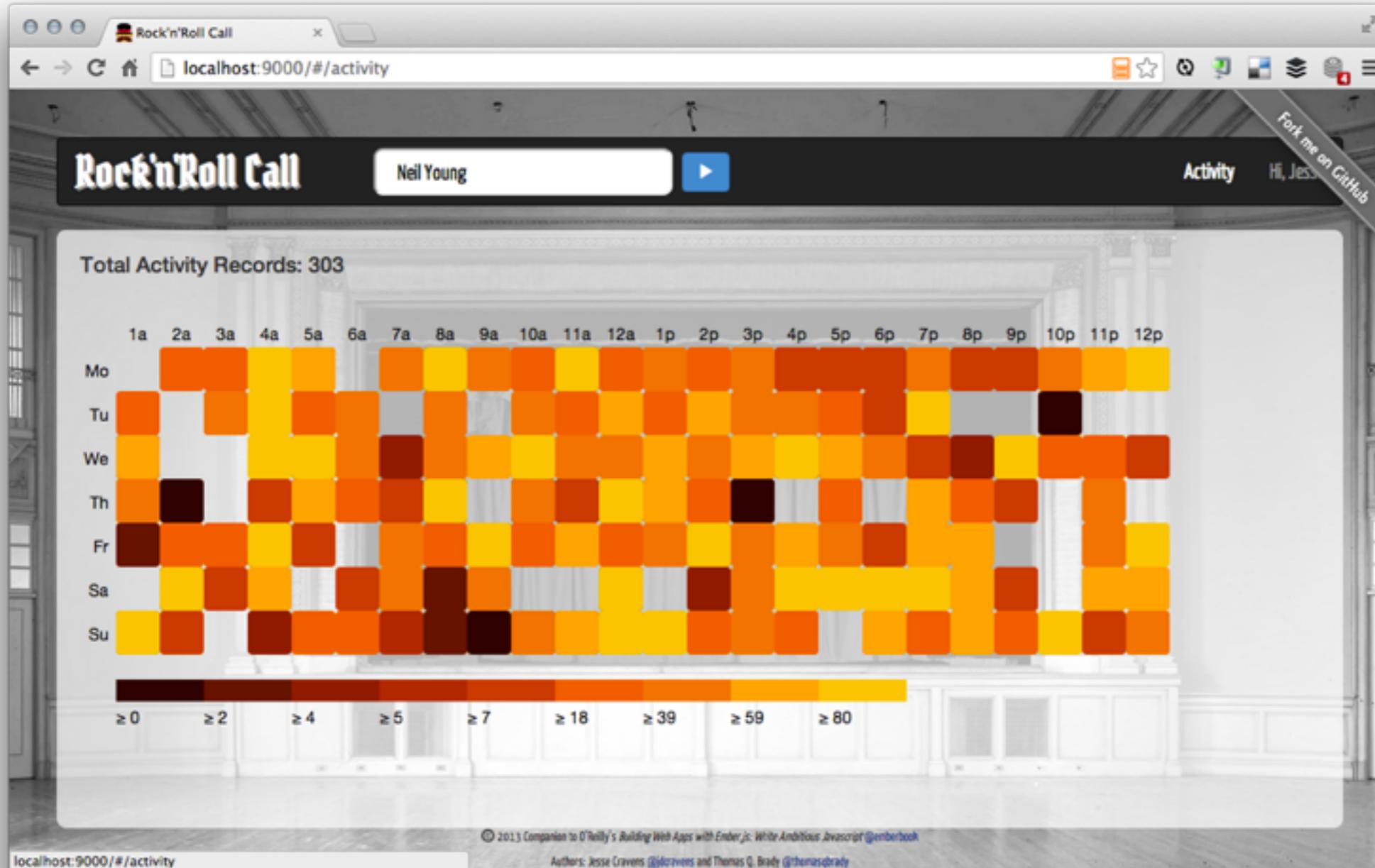
# DESIGN IN .HBS

## ARTIST/SONG DETAIL STATE



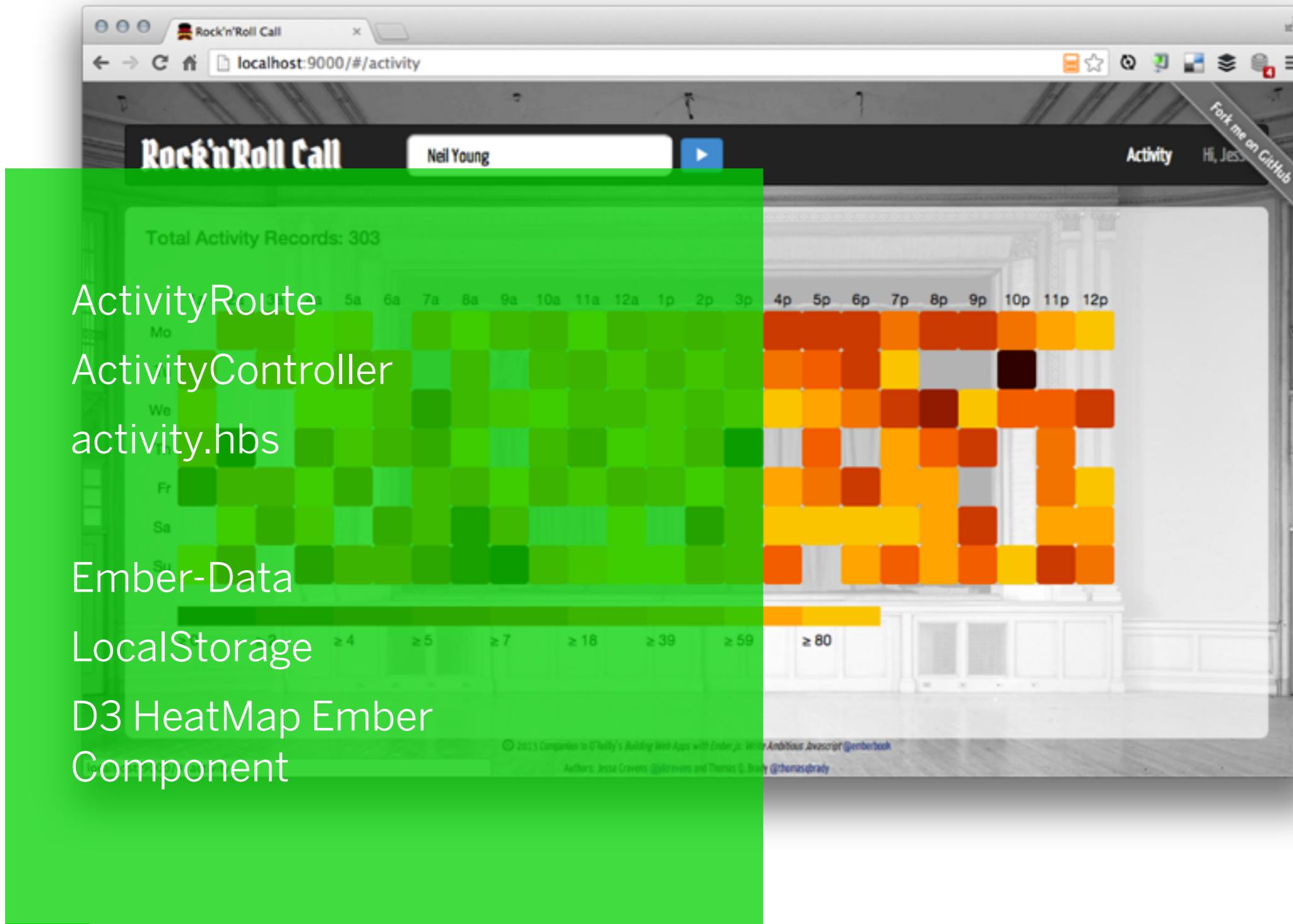
# DESIGN IN .HBS

## ACTIVITIES STATE



# DESIGN IN .HBS

## ACTIVITIES STATE



# HANDLEBARS BASICS

## chapter 4



# HANDLEBARS INLINE

```
<script type="text/x-handlebars" data-template-name="application"    <!-- template code here -->  
</script>
```



# HANDLEBARS .HBS

```
grunt.initConfig({
  yeoman: yeomanConfig,
  watch: {
    emberTemplates: {
      files: '<%= yeoman.app %>/templates/**/*.hbs',
      tasks: ['emberTemplates', 'livereload']
    }
  }
});
```



# HANDLEBARS VARIABLES

```
<a href="#" class="navbar-brand">
```

```
  {{App.applicationName}}
```

```
</a>
```



# HANDLEBARS LINKS

```
{{#linkTo "index" class="navbar-brand"}}
```

```
{App.applicationName}}
```

```
{{/linkTo}}
```



# HANDLEBARS LISTS

```
<ul class="search-results artists">  
{{#each} artist in App.dummySearchResultsArtists}}
```

```
 {{#each} nickname in artist.nicknames}}
```

```
<|i>  
 <a href="#">  
   {{nickname}}, AKA "{{artist.name}}"  
 </a>  
</|i>
```

```
 {{/each}}
```

```
 {{/each}}  
</ul>
```



# HANDLEBARS LOGIC

```
<ul class="search-results artists">
  {{#each artist in App.dummySearchResultsArtists}}
    {{#if artist.nicknames}}
      {{#each nickname in artist.nicknames}}
        <li>
          <a href="#">{{nickname}}</a>
        </li>
      {{/each}}
    {{else}}
      <li><a href="#">{{artist.name}}</a></li>
    {{/if}}
  {{/each}}
</ul>
```



# HBS BOUND ATTRIBUTES

```
<a {{bind-attr href=video.url}}>
```

Watch now on Vimeo

```
</a>
```



# HANDLEBARS HELPERS

```
Ember.Handlebars.helper('hotttnesss-icon',
function(value, options) {
  var h = parseFloat(value);
  var hotttnesss = Math.ceil(h * 10) - 1;
  var html = "<h4>Hotness: ";
  if (hotttnesss > 0) {
    html += '<i class="hotttnesss">';
    ...
  }
  ...
} else {
  html += "0</h4>";
}
return new Handlebars.SafeString(html);
});
```



# THE ROUTER

## chapter 5



# URL DRIVEN DESIGN

- \* Think 'STATE'
- \* Router is a StateManager
- \* NO MODELS: return data in .model() hook
- \* NO VIEWS: templates first



# SAVED GAME PASSWORDS



# ROUTER

- \* Connect URLs to Code
- \* Manages 'states' of the Ember application



# LOG TRANSITIONS

```
App = Ember.Application.create({
```

**LOG\_TRANSITIONS: true**

```
});
```



# MAP

```
App.Router.map(function() {
```

```
    // routes
```

```
});
```



# MAPPING ROUTES

```
App.Router.map(function() {
```

```
    this.route('search-results');  
    this.route('artist');  
    this.route('song');  
    this.route('activity');
```

```
});
```



# DYNAMIC ROUTES

```
App.Router.map(function() {  
  this.route('search-results', {  
    path: 'search/:term'  
  });  
  this.route('artist', {  
    path: 'artist/:enid'  
  });  
  this.route('song', {  
    path: 'song/:enid'  
  });  
});
```



# ROUTE HANDLERS

```
App.ArtistRoute = Ember.Route.extend({
```

```
  // handle a transition to this Route
```

```
});
```



# THE MODEL HOOK

```
App.ArtistRoute = Ember.Route.extend({  
  model: function(params) {  
    }  
  });
```



# ROUTE HANDLERS

```
App.ArtistRoute = Ember.Route.extend({
  model: function(params) {
    XHR("some URL", {"id":params.enid}, function
    callback(response){
      var artist = App.Artist.create({
        name: response.name,
        hotttnesss: response.hotttnesss,
      });
    });
  }
});
```



# EMBER OBJECT

```
App.Artist = Em.Object.extend({  
  id: null,  
  name: null,  
  enid: null,  
  biography: null,  
  hotttnesss: null,  
  image: null,  
  videos: null  
});
```



# PROMISES

**Promise.all**([afunction(),another(), yetAnother()])

```
.then(function() {  
  console.log("They're all finished, success is ours!");  
}, function() {  
  console.error("One or more FAILED!");  
});
```



# ROUTE HANDLERS

**App.ArtistRoute** = Ember.Route.extend({

```
// handle a transition to Artist Detail State  
// call Echo Nest API
```

**App.SongRoute** = Ember.Route.extend({

```
// handle a transition to Song Detail State  
// call Echo Nest API
```



# LOADING ROUTE

**App.LoadingRoute** = Ember.Route.extend({});

```
<div id="lp"></div>
<audio
  src="sounds/tuning.mp3"
  autoplay="autoplay">
</audio>
```



# CONTROLLERS

chapter 6



# CONTROLLERS

- \* Manipulate the data within the application's Models
- \* Store transient data, whether stand-alone or made up of data retrieved from Models
- \* Listen to events dispatched by and dispatch events intended for other Controllers, Views and Templates
- \* Instigate the transition from one route to another



# APPLICATION CONTROLLER

```
App.ApplicationController =  
Em.ObjectController.extend({  
  
  searchTerms: "",  
  
  applicationName: "Rock'n'Roll Call",  
  
  actions: {  
    submit: function() {  
      this.transitionToRoute('search-  
results',this.get('searchTerms'));  
    }  
  }  
  
});
```



# VALUE BINDING

```
{{view Ember.TextField  
  class="search-input"  
  placeholder="Search for artists or song names"  
valueBinding="controller.searchTerms"}}}
```

```
<button class="btn btn-primary">  
  <i class="glyphicon glyphicon-play"></i>  
</button>
```



# .GET() AND .SET()

```
App.ApplicationController =  
Em.ObjectController.extend({  
  
  searchTerms: '',  
  
  applicationName: function() {  
    var st = this.get('searchTerms');  
    if (st) {  
      return st + "???"  
    } else {  
      return "Rock'n'Roll Call"  
    }  
  }.property('searchTerms'),  
  
  ...  
});
```



# COMPUTED PROPERTIES

```
App.ApplicationController =  
Em.ObjectController.extend({  
  ...  
  applicationName: function() {  
    var st = this.get('searchTerms');  
    if (st) {  
      return st + "???"  
    } else {  
      return "Rock'n'Roll Call"  
    }  
  }.property('searchTerms'),  
  ...  
});
```



# ACTIONS

```
App.ApplicationController =  
Em.ObjectController.extend({
```

```
...
```

```
actions: {
```

```
...
```

```
}
```

```
});
```



# SUBMIT

```
{{view Ember.TextField  
  class="search-input"  
  placeholder="Search for artists or song names"  
action="submit"  
  valueBinding="controller.searchTerms"}}
```

```
<button {{action "submit"}} class="btn btn-primary">  
  <i class="glyphicon glyphicon-play"></i>  
</button>
```



# TRANSITIONTOROUTE

```
App.ApplicationController =  
Em.ObjectController.extend({  
  
  ...  
  
  actions: {  
    submit: function() {  
      this.transitionToRoute('search-  
results',this.get('searchTerms'));  
    }  
  }  
  
});
```



# CHECKED BINDING

```
App.SearchResultsController =  
Em.ObjectController.extend({
```

```
...
```

```
  artistsIsChecked: true,  
  songsIsChecked: true
```

```
...
```

```
});
```

```
{{view Ember.Checkbox  
  checkedBinding=“artistsIsChecked”}}
```

```
  {{#if artistsIsChecked}}
```

```
  ...
```

```
  {{/if}}}
```



# EMBER-DATA

## chapter 7



# ACTIVITY TEMPLATE

```
<h4>Total Activity Records: {{model.length}}</h4>
```

```
<ul>
{{#each model}}
<li class="activity">{{this.id}}</li>
{{/each}}
</ul>
```



# ACTIVITY ROUTE

```
App.Router.map(function() {
```

```
  ...
```

```
    this.route('activity');
```

```
});
```

```
App.ActivityRoute = Ember.Route.extend({
```

```
  model: function () {
```

```
    return this.store.find('activity');
```

```
  }
```

```
});
```



# ACTIVITY MODEL

```
App.Activity = DS.Model.extend({  
  display_id: DS.attr('string'),  
  type: DS.attr('string'),  
  display_name: DS.attr('string'),  
  hotttnesss: DS.attr('number'),  
  timestamp: DS.attr()  
});
```



# ACTIONS

```
{#{if artistsIsChecked}}
{#{if artists.length}}
<h3>Artists</h3>
<ul class="search-results artists">
{#each artists}
<li>
<a {{action 'viewedArtist' this }}>{{name}}</a>
</li>
{/each}
</ul>
{/if}
{/if}
```



# CREATERECORD()

```
actions: {
    viewedSong: function(model) {
        var date = Date.now();
        var activity = this.store.createRecord('activity', {
            display_id: model.sid,
            type: model.type,
            display_name: model.artist_name,
            hotttnesss: model.hotttnesss,
            timestamp: date
        });
        ...
    }
},
```



# SAVE()

```
actions: {  
  viewedSong: function(model) {  
    ...  
    var activity = this.store.createRecord('activity', {  
      display_id: model.sid,  
      type: model.type,  
      display_name: model.artist_name,  
      hotttnesss: model.hotttnesss,  
      timestamp: date  
    });  
    activity.save();  
  }  
},
```



# FILTER()

```
store.filter('activity', function(activity){  
    return activity.get('type', 'song');  
});
```



# ALL()

**store.all('activity');**



# ADAPTERS

```
App.ApplicationAdapter = DS.FixtureAdapter.extend({  
  namespace: 'rocknrollcall'  
});
```

```
App.ApplicationAdapter = DS.LSAdapter.extend({  
  namespace: 'rocknrollcall'  
});
```

```
App.ActivityAdapter = DS.LSAdapter.extend({  
  namespace: 'rocknrollcall'  
});
```

```
App.ApplicationAdapter = DS.RESTAdapter.extend({  
  namespace: 'rocknrollcall'  
});
```



# APP INITIALIZER / SEEDS

```
Ember.Application.initializer({  
  name: "DBseeds",  
  initialize: function(container, application) {  
  
    localStorage.clear();  
  
    ...  
  }  
});
```



# BUILDING A BACKEND

chapter 8



# EMBER APP KIT

```
{  
  "name": "app-kit",  
  "namespace": "appkit",  
  "APIMethod  ...  
}
```



# ROUTES.JS

```
module.exports = function(server) {  
  
    // Create an API namespace, so that the root does not  
    // have to be repeated for each end point.  
    server.namespace("/api", function() {  
  
        // Return fixture data for "/api/activities"  
        server.get("/activities", function(req, res) {  
            var activities = [ ];  
            };  
            res.send(activities);  
        });  
    });  
};
```



# RESTADAPTER

```
export default DS.RESTAdapter.extend({  
  namespace: 'api'  
});
```



# EMBER APP KIT

```
{  
  "name": "app-kit",  
  "namespace": "appkit",  
  "APIMethod": "stub",  
  ...  
}
```

```
{  
  "name": "app-kit",  
  "namespace": "appkit",  
  "APIMethod": "proxy",  
  "proxyURL": "http://whatever.api:3232",  
  ...  
}
```



# RAILS AND EMBER

- \* Rails helps you manage your Ruby code, server side MVC.
- \* Active Record provides an ORM for interacting with your data.
- \* Rails Asset Pipeline, and the ember-rails gem, help you manage your assets.
- \* Active Model Serializers provide the REST API that the Ember Data RESTAdapter expects by default.
- \* Testing is built-in.



# ACTIVE:MODEL SERIALIZERS

```
class ActivitySerializer < ActiveModel::Serializer
```

```
# look up 'typ' on the model, but use 'type' in the JSON
```

```
def type
```

```
  object.typ
```

```
end
```

```
attributes :id, :display_id, :type, :display_name, :timestamp
```

```
end
```



# COMPONENTS

chapter 9



# WEB COMPONENTS

- \* HTML Imports ... Em.TEMPLATES
- \* Shadow DOM
- \* Model Driven Views
- \* Custom Elements
- \* For more on Web Components: <http://jessecravens.com/blog/2013/07/25/building-next-generation-widgets-with-web-components/>



# ANATOMY

## ACTIVITY LOG

- \* File System: templates/components/activity-log.hbs
- \* Custom Elements naming convention: {{activity-log}}

## HEAT MAP

- \* File System: templates/components/heat-map.hbs
- \* Custom Elements naming convention: {{heat-map}}



# REUSE ACTIVITY-LOG.HBS

```
<p> display_id: {{display_id}}</p>
<p> type: {{type}}</p>
<p> display_name: {{display_name}}</p>
<p> hotttnesss: {{hotttnesss}}</p>
<p> timestamp: {{timestamp}}</p>
<hr>
```

And in activities.hbs:

```
{{#each}}
{{activity-log}}
{{/each}}
```



# SCOPE ISSUE

`{#each}`

`{{activity-log display_id=display_id type=type  
display_name=display_name hotttnesss=hotttnesss  
timestamp=timestamp }}`

`{/each}`



# SUBCLASS

```
App.HeatMapComponent =  
Ember.Component.extend({});
```



# DIDINSERTELEMENT()

```
App.HeatMapComponent = Ember.Component.extend({
```

```
  didInsertElement: function(){
    var data = this.get('controller.data.content');
    this.draw(data);
  }
});
```



# 3RD PARTY PATTERN: D3.JS

```
App.HeatMapComponent = Ember.Component.extend({
```

```
  width: 900,  
  height: 280,
```

```
  draw: function(myData){  
    // draw the heat map  
  },
```

```
  didInsertElement: function(){  
    var data = this.get('controller.data.content');  
    this.draw(data);  
  }  
});
```



# TESTING

## chapter 10



# TESTING SETUP - EAK

```
document.write('<div id="ember-testing-  
container"><div id="ember-testing"></div></div>');
```

```
Ember.testing = true;
```

```
window.startApp = require('appkit/tests/helpers/  
start_app')['default'];
```

```
window.isolatedContainer = require('appkit/tests/  
helpers/isolated_container')['default'];
```

```
... // test-helpers
```

```
window.exists = exists;
```

```
window.equal = equal;
```

```
window.strictEqual = strictEqual;
```



# STARTAPP()

```
import Application from 'appkit/app';

function startApp(attrs) {
  var App;

  ...

  Ember.run(function(){
    App = Application.create(attributes);
    App.setupForTesting();
    App.injectTestHelpers();
  });
}

return App;
}

export default startApp;
```



# HELPERS

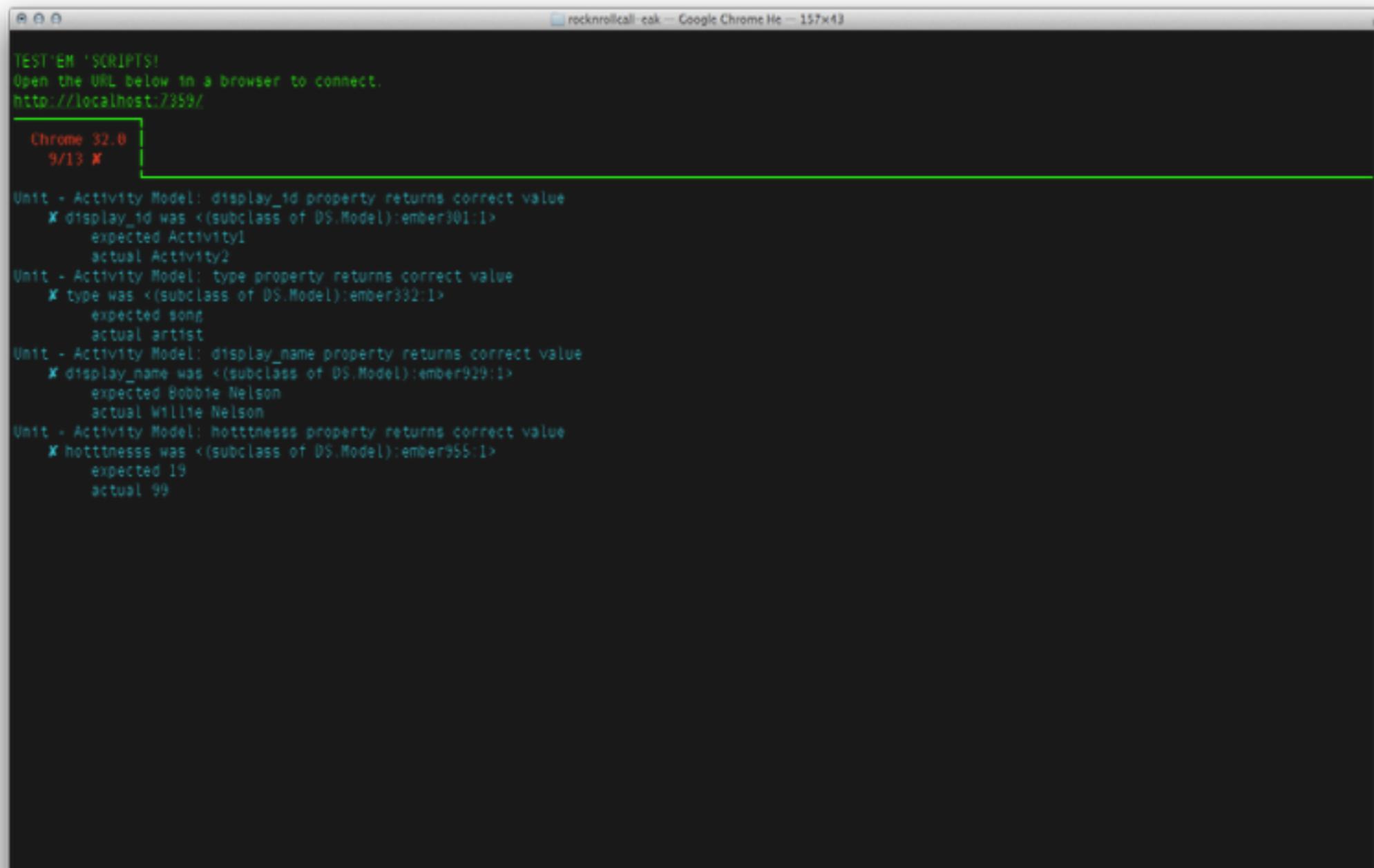
```
function exists(selector) {  
    return !!find(selector).length;  
}
```

```
function equal(actual, expected, message) {  
    message = getAssertionMessage(actual, expected,  
        message);  
    QUnit.equal.call(this, actual, expected, message);  
}
```

```
function strictEqual(actual, expected, message) {  
    message = getAssertionMessage(actual, expected,  
        message);  
    QUnit.strictEqual.call(this, actual, expected, message);  
}
```



# TESTEM TEST RUNNER



rocknrollcall-eak — Google Chrome He — 157x43

```
TESTEM 'SCRIPTS!
Open the URL below in a browser to connect.
http://localhost:7359/

Chrome 32.0
  9/13 X

Unit - Activity Model: display_id property returns correct value
  ✘ display_id was <(subclass of DS.Model):ember301:1>
    expected Activity1
    actual Activity2
Unit - Activity Model: type property returns correct value
  ✘ type was <(subclass of DS.Model):ember302:1>
    expected song
    actual artist
Unit - Activity Model: display_name property returns correct value
  ✘ display_name was <(subclass of DS.Model):ember929:1>
    expected Bobbie Nelson
    actual Willie Nelson
Unit - Activity Model: hotttnesss property returns correct value
  ✘ hotttnesss was <(subclass of DS.Model):ember955:1>
    expected 19
    actual 99
```



# QUNIT TEST RUNNER

The screenshot shows a web browser window displaying the QUnit Test Runner. The title bar says "App (9/13)" and the address bar says "localhost:7359/9124/tmp/result/tests/index.html". The main area is titled "App" and contains the following information:

Module: < All Modules >

Tests completed in 982 milliseconds.  
15 assertions of 19 passed; 4 failed.

Test results:

1. Acceptances - Activities: activities renders (0, 2, 2) 180 ms
2. Acceptances - Component: component output is rendered (0, 2, 2) 00 ms
3. Acceptances - Helper: helper output is rendered (0, 1, 1) 80 ms
4. Acceptances - Index: index renders (0, 3, 3) 73 ms
5. Unit - Activity Model: display\_id property returns correct value (1, 0, 1) 113 ms  
  1. display\_id was <subclass of DS.Model>ember301:15  
    **Expected:** "Activity1"  
    **Result:** "Activity2"  
    **Diff:** "Activity1" "Activity2"  
    **Source:**   at equal ([http://localhost:7359/tests/test\\_helper.js](http://localhost:7359/tests/test_helper.js):18:15)  
          at <http://localhost:7359/tests/test.js>:186:11  
          at invokeCallback (<http://localhost:7359/vendor/ember/ember.js>:9892:19)  
          at publish (<http://localhost:7359/vendor/ember/ember.js>:9882:19)  
          at Promise.publish#fulfillment (<http://localhost:7359/vendor/ember/ember.js>:9982:7)  
          at Object.DeferredActionQueue.flush (<http://localhost:7359/vendor/ember/ember.js>:6027:24)  
          at Object.Backburner.end (<http://localhost:7359/vendor/ember/ember.js>:6118:27)
6. Unit - Activity Model: type property returns correct value (1, 0, 1) Run  
  1. type was <subclass of DS.Model>ember332:15  
    **Expected:** "song"  
    **Result:** "artist"  
    **Diff:** "song" "artist"  
    **Source:**   at equal ([http://localhost:7359/tests/test\\_helper.js](http://localhost:7359/tests/test_helper.js):18:15)  
          at <http://localhost:7359/tests/test.js>:179:11  
          at invokeCallback (<http://localhost:7359/vendor/ember/ember.js>:9892:19)  
          at publish (<http://localhost:7359/vendor/ember/ember.js>:9882:19)  
          at Promise.publish#fulfillment (<http://localhost:7359/vendor/ember/ember.js>:9982:7)  
          at Object.DeferredActionQueue.flush (<http://localhost:7359/vendor/ember/ember.js>:6027:24)  
          at Object.Backburner.end (<http://localhost:7359/vendor/ember/ember.js>:6118:27)
7. Unit - Activity Model: display\_name property returns correct value (1, 0, 1) Run  
  1. display\_name was <subclass of DS.Model>ember322:15



# INTEGRATION MODULES

```
var App;  
  
module('Acceptances - Activities', {  
  setup: function(){  
    App = startApp();  
  },  
  teardown: function() {  
    Ember.run(App, 'destroy');  
  }  
});  
  
// tests  
});
```



# INTEGRATION TESTS

```
module('Acceptances - Activities', {
```

```
  ...
```

```
});
```

```
test('activities renders', function(){
```

```
  visit('/activities').then(function(){
```

```
    var title = find('h4');
```

```
    var list = find('ul li.activity');
```

```
    equal(title.text(), 'Total Activity Records: 2');
```

```
    equal(list.length, 2);
```

```
  });
```

```
});
```



# TESTING ROUTES

```
export default Ember.Route.extend({  
  model: function() {  
    return ['red', 'yellow', 'blue'];  
  }  
});
```

```
export default Ember.Route.extend({  
  model: function() {  
    return this.get('store').find('activity');  
  }  
});
```



# UNIT TESTING ROUTES

```
import Activities from 'appkit/routes/activities';
```

```
var route;
```

```
module('Unit - ActivitiesRoute', {
  setup: function() {
    var container = isolatedContainer[
      'route:activities'
    ]);
    route = container.lookup('route:activities');
  }
});
```



# ISOLATED CONTAINER

```
import Resolver from 'resolver';

function isolatedContainer(fullNames) {
  var container = new Ember.Container();
  ...
  var resolver = Resolver['default'].create();

  for (var i = fullNames.length; i > 0; i--) {
    var fullName = fullNames[i - 1];
    container.register(fullName,
      resolver.resolve(fullName));
  }
  return container;
}
export default isolatedContainer;
```



# TESTING ROUTES

```
test('route exists', function() {  
    ok(route);  
    ok(route instanceof Activities);  
});
```



# TESTING ROUTES MODEL()

```
test('#model', function() {  
  
  var store = {  
    find: function() {  
      return [{...}];  
    }  
  };  
  
  route.set('store', store);  
  
  deepEqual(route.model(), [{ ... }]);  
});
```



# USING FIXTURES

```
var Activity = DS.Model.extend({  
  display_id: DS.attr('string'),  
  ...  
});
```

```
Activity.FIXTURES = [  
  {...},  
  {...}  
];
```

```
export default Activity;
```



# USING FIXTURES

```
import Activities from 'appkit/routes/activities';
import Activity from 'appkit/models/activity';

test('#model', function() {

  var store = {
    find: function() {
      return Activity.FIXTURES;
    }
  };
  route.set('store', store);

  deepEqual(route.model(), Activity.FIXTURES);

});
```



# TESTING PROMISES

```
module("Unit - ActivitiesRoute", {
  setup: function() {

    var store = {
      find: function(type) {
        return new Em.RSVP.Promise(function(resolve) {
          resolve(Activity.FIXTURES);
        });
      }
    };
    route.set('store', store);
  }
});
```



# ASYNCTEST, EM.RUN()

```
asyncTest('#model', function(){
```

```
    Em.run(function(){
```

```
        route.model().then(function(result){
```

```
            equal(result, Activity.FIXTURES);
```

```
            start();
```

```
        });
```

```
    });
```

```
});
```



# TESTING MODELS

```
module('Unit - Activity Model', {
  setup: function() {
    App = startApp();
    store = App.__container__.lookup('store:main');
  },
  teardown: function() {
    Ember.run(App, 'destroy');
  }
});
```



# TESTING MODELS

```
asyncTest('display_id property returns correct value',
function() {

  Ember.run(function () {
    store.find('activity', 1).then(function(result){
      equal(result.get('display_id'), 'Activity2', 'display_id
was ' + result);

      start();
    });
  });
});
```



**GRACIAS Y ADIOS.**



