Ember.js

Russell Jones

Russell Jones

@codeofficer
EVERYWHERE

What is Ember

- JavaScript web application framework
- Based on model-view-controller (MVC)
- Designed for Single Page Applications
- Ember.js is omakase, trust the chef

History

- Sproutcore 1.0 (MobileMe, iCloud)
- Sproutcore 2.0?
- Application Framework <> Widget Library
- Amber.js (oops) > Ember.js

Core Concepts

- Ember Application
- Run Loop (backburner.js)
- Basic Object Model
- MVC* Pattern

Ember Application

```
window.App = Ember.Application.create();
```

- Creates a Namespace
- Bind Event Listeners
- Renders the Application's Template
- Starts the Application's Router

Ember Application

```
window.App = Ember.Application.create({
  rootElement: "body"
});

App.deferReadiness();

App.advanceReadiness();
```

```
<script type="text/x-handlebars" data-template-name=“application">
  <header></header>
  {{outlet}}
  <footer></footer>
</script>
```

```
<script type="text/x-handlebars" data-template-name="index">
  I’m all up in the DOM!
</script>
```

Run Loop (Backburner.js)

http://talks.erikbryn.com/backburner.js-and-the-ember-run-loop/

ember-run-loop-visualization

- sync - synchronizing bindings
- actions - app actions
- render - Ember's DOM-related changes
- afterRender
- destroy - object destruction

Basic Object

- Classes and Instances
- Computed Properties
- Bindings
- Observers

Classes and Instances

```
var Person = Ember.Object.extend({
  say: function(something) {
    alert(something);
  }
});

var AuthorityFigure = Person.extend({
  say: function(something) {
    var person = this.get('name');
    this._super(person + " says, " + something);
  }
});

var simon = AuthorityFigure.create({
  name: "Simon"
});


simon.say("Jump"); // alert "Simon says, Jump"
```

Computed Properties

```
var Person = Ember.Object.extend({
  firstName: null,
  lastName: null,

  fullName: function() {
    var firstName = this.get('firstName');
    var lastName = this.get('lastName');

   return firstName + ' ' + lastName;
  }.property('firstName', 'lastName')
});


var russ = Person.create({
  firstName: "Russ",
  lastName: "Jones"
});


russ.get('fullName') // "Russ Jones"
```

Computed Properties

```
var Controller = Ember.Object.extend({
  records: [
    Ember.Object.create({isProcessed: true}),
    Ember.Object.create({isProcessed: false})
  ],

  total: function() {
    return this.get('records.length');
  }.property('records.@each.isProcessed'),

  remaining: function() {
    var records = this.get('records');
    return records.filterProperty('isProcessed', false).get('length');
  }.property('records.@each.isProcessed')
});


Controller.create().get('total'); // 2
Controller.create().get('remaining'); // 1
```

Bindings

A binding creates a link between two properties such that when one changes, the other one is updated to the new value automatically.

```
App.PostController = Ember.ObjectController.extend({
  content:Ember.Object.create({
    title: "Ember vs Angular"
  })
});

App.CommentsController = Ember.ArrayController.extend({
  needs: [“post”],
  post: Ember.computed.alias("controllers.post")
});



// each of these are equal …
postController
commentsController.get('post')
commentsController.get('controllers.post')
```

Observers

Observers in Ember are currently synchronous

```
var Citizen = Ember.Object.extend({
  fullName: function() {
    var firstName = this.get('firstName');
    var lastName = this.get('lastName');

    return firstName + ' ' + lastName;
  }.property('firstName', 'lastName'),

  fullNameChanged: function() {
    // alert the NSA ...
  }.observes('fullName')
});
```

Prototype Extensions

By default, Ember.js will extend the prototypes
of these native JavaScript objects

- Array
- String
- Function

Array Prototype Extensions

Array is extended to implement Ember.Enumerable, Ember.MutableEnumerable, Ember.MutableArray and Ember.Array

```
[1,2,3].forEach(function(item) {
  console.log(item);
});

// returns
// 1
// 2
// 3
```

```
[1,2,3].get('lastObject');

// returns 3
```

```
[1,2,3,4,5].filter(function(item, index, self) {
  if (item < 4) { return true; }
})

// returns [1,2,3]
```

```
["goodbye", "cruel", "world"].map(function(item) {
  return item + "!";
});

// returns ["goodbye!", "cruel!", "world!"]
```

Ember.A([]);

Array Prototype Extensions

addEnumerableObserver
compact
contains
enumerableContentDidChange
enumerableContentWillChange
every
everyProperty
filter
filterProperty
find
findProperty
forEach
getEach
invoke
map
mapProperty
nextObject
reduce
reject
rejectProperty
removeEnumerableObserver
setEach
some
someProperty
toArray
uniq
without

String Prototype Extensions

String is extended to add convenience methods, such as camelize() and fmt()

```
'my string'.camelize();  // 'myString'
'my string'.capitalize() // 'My string'
'my string'.classify();  // 'MyString'
'Hello %@2, %@1'.fmt('John', 'Doe');  // "Hello Doe, John"
'my string'.underscore();  // 'my_string'
'my'.w();  // ['my', 'string']


Ember.STRINGS = {
  '_Hello World': 'Bonjour le monde',
  '_Hello %@ %@': 'Bonjour %@ %@'
};

'_Hello World'.loc();  // 'Bonjour le monde';
'_Hello %@ %@'.loc(['John', 'Smith']);  // "Bonjour John Smith"
```

Ember.String.camelize('my string');

Function Prototype Extensions

Function is extended with methods to annotate functions as computed properties or observers.

```
Ember.Object.create({
  valueObserver: function() {
    // Executes whenever the "value" property changes
  }.observes('value')
});

Ember.Object.create({
  valueObserver: function() {
    // Executes whenever the "value" property is about to change
  }.observesBefore('value')
});

var president = Ember.Object.create({
  firstName: "Barack",
  lastName: "Obama",

  fullName: function() {
    return this.get('firstName') + ' ' + this.get('lastName');
  }.property('firstName', 'lastName')
});

president.get('fullName'); // "Barack Obama"
```

fullName: Ember.computed(function() { /* … */ }).property('firstName', 'lastName')

Template

- http://handlebarsjs.com/
- http://emblemjs.com/ (HAML in JS)
- Bound to a Context
- Contain HTML and {{expressions}}

Template

```
// context
App.ApplicationController = Ember.Controller.extend({
  firstName: "Russ",
  lastName: "Jones"
});
```

```
// template
Hello, <strong>{{firstName}} {{lastName}}</strong>!
```

```
// output
Hello, <strong>Russ Jones</strong>!
```

Template

```
// conditionals
{{#if person}}
  Welcome back, <b>{{person.firstName}} {{person.lastName}}</b>!
{{else}}
  Please log in.
{{/if}}


// displaying a list of items
{{#each people}}
  Hello, {{name}}!
{{else}}
  Sorry, nobody is here.
{{/each}}
```

Template

```
// binding attributes
<div id="logo">
  <img {{bindAttr src=logoUrl}} alt="Logo">
</div>

// output
<div id="logo">
  <img src="http://www.example.com/images/logo.png" alt="Logo">
</div>


// binding classes
<input type="checkbox" {{bindAttr disabled=isAdministrator}}>

// output
<input type="checkbox" disabled>
```

Model

An (Ember Data) object that stores persistent state.
http://emberjs.com/guides/models/

```
App.Person = DS.Model.extend({
  firstName: DS.attr('string'),
  lastName: DS.attr('string'),

  fullName: function() {
    return this.get('firstName') + ' ' + this.get('lastName');
  }.property('firstName', 'lastName')
});


store.createRecord('person', {firstName: 'Russ', lastName: 'Jones'})

var russ = store.find('person', 1);

russ.get('fullName'); // Russ Jones
russ.get('isDirty'); // false

russ.set('lastName', 'Jeep Lover');
russ.get('isDirty'); // true
```

View

Encapsulates templates, combines templates with data and responds to user initiated events

```
App.PlaybackRoute = Ember.Route.extend({
  events: {
    turnItUp: function(level){
      //This won't be called since it's defined on App.PlaybackController
    }
  }
});

App.PlaybackController = Ember.ObjectController.extend({
  turnItUp: function(level){
  }
});

// lets assume that PlaybackController is our view’s context

App.ClickableView = Ember.View.extend({
  click: function(e) {
    this.get('controller').send('turnItUp', 11);
  }
});
```

Component

A completely isolated View that has no access to the surrounding context.
It's a great way to build reusable components for your apps.

```
App.PostSummaryComponent = Ember.Component.extend({
  actions: {
    toggleBody: function() {
      this.toggleProperty('isShowingBody');
    }
  }
});
```

```
<script type="text/x-handlebars" id="components/post-summary">
  <h3 {{action "toggleBody"}}>{{title}}</h3>
  {{#if isShowingBody}}
    <p>{{{body}}}</p>
  {{/if}}
</script>
```

```
{{#each}}
  {{post-summary title=title body=body}}
{{/each}}
```

Controller

- Decorates a Model
- ObjectController
- ArrayController

Controller

```
App.Router.map(function() {
  this.resource("post", { path: "/posts/:post_id" }, function() {
    this.resource("comments", { path: "/comments" });
  });
});

App.CommentsController = Ember.ArrayController.extend({
  needs: [“post”],
  post: Ember.computed.alias(”controllers.post”)
});

// and in the comments template ...

<h1>Comments for {{controllers.post.title}}</h1>

<h1>Comments for {{post.title}}</h1>

<ul>
  {{#each comment in controller}}
    <li>{{comment.text}}</li>
  {{/each}}
</ul>
```

Router

http://emberjs.com/guides/routing/

http://www.youtube.com/watch?v=gz7Jy2abm10

https://gist.github.com/machty/5723945

- Translates a URL into a series of nested templates, each backed by a model
- Represents Application State as a URL
- Routes may generate all the things
- Updates the controllers

Router

Location API ... hash / push state / x.x

```
/#/posts/new => ‘posts.new‘ route

App.Router.map(function() {
  this.resource('posts', function() {
    this.route('new');
  });
});

/posts/new

App.Router.reopen({
  location: 'history'
});

x.x

App.Router.reopen({
  location: 'none'
});
```

Router

```
App.Router.map(function() {
  this.resource('posts');
  this.resource('post', { path: '/posts/:post_id' });
});

#/
App.IndexRoute = Ember.Route.extend({
  redirect: function() {
    this.transitionTo('posts');
  }
});

#/posts
App.PostsRoute = Ember.Route.extend({
  model: function() {
    return App.Post.find();
  }
});

#/posts/1
App.PostRoute = Ember.Route.extend({
  setupController: function(controller, model) {
    controller.set('model', model);
  }
});
```

model
setupController
renderTemplate
redirect

Ember Testing

```
App.rootElement = '#arbitrary-element-that-doesnt-interfere-with-qunit';

App.setupForTesting();
  * Ember.testing = true;
  * deferReadiness()
  * set location api to 'none'

App.injectTestHelpers();
  * visit (url)
  * find(selector, context)
  * fillIn(input_selector, text)
  * click(selector)
  * keyDown(selector, type, keyCode)
  * wait()

module("Integration Tests", {
  setup: function() {
    App.reset();
  }
});

test("creating a post displays the new post", function() {
  visit("/posts/new")
  .fillIn(".post-title", "A new post")
  .fillIn(".post-author", "John Doe")
  .click("button.create")
  .then(function() {
    ok(find("h1:contains('A new post')").length, "The post's title should display");
    ok(find("a[rel=author]:contains('John Doe')").length, "A link to the author should display");
  });
});
```

Useful App Stuff

```
App = Ember.Application.create({
  LOG_STACKTRACE_ON_DEPRECATION : true,
  LOG_BINDINGS : true, // LOG OBJECT BINDINGS
  LOG_TRANSITIONS : true, // LOG ROUTER TRANSITIONS
  LOG_TRANSITIONS_INTERNAL : true, // SEE ALL THE THINGS
  LOG_VIEW_LOOKUPS : true, // LOG VIEW LOOKUPS
  LOG_ACTIVE_GENERATION : true, // LOG GENERATED CONTROLLER

  rootElement: '#app',

  ready: function(){
    console.log('App.ready()')
  },
  lookupStore: function() {
    return this.__container__.lookup('store:main');
  },
  lookupRouter: function() {
    return this.__container__.lookup('router:main');
  },
  lookupController: function(controllerName, options) {
    return this.__container__.lookup('controller:' + controllerName, options);
  },
  lookupContainer: function() {
    return this.__container__;
  }
});

App.deferReadiness();
```

Useful App Stuff

http://emberjs.com/guides/understanding-ember/debugging/

VIEW ALL REGISTERED ROUTES

```
Ember.keys(App.Router.router.recognizer.names)
```

VIEW ALL REGISTERED TEMPLATES

```
Ember.keys(Ember.TEMPLATES)
```

GET THE STATE HISTORY OF AN EMBER-DATA RECORD

```
record.stateManager.get('currentPath')
```

GET THE VIEW OBJECT FOR A GENERATED EMBER div BY ITS DIV ID

```
Ember.View.views['ember605']
```

LOG STATE TRANSITIONS

```
record.set("stateManager.enableLogging", true)
```

Useful App Stuff

VIEW AN INSTANCE OF SOMETHING FROM THE CONTAINER

```
App.__container__.lookup("controller:posts")
App.__container__.lookup("route:application")
```

VIEW EMBER-DATA'S IDENTITY MAP

```
// all records in memory
App.__container__.lookup('store:main').recordCache

// attributes
App.__container__.lookup('store:main').recordCache[2].get('data.attributes')

// loaded associations
App.__container__.lookup('store:main').recordCache[2].get('comments')
```

SEE ALL OBSERVERS FOR A OBJECT, KEY

```
Ember.observersFor(comments, keyName);
```

DEALING WITH DEPRECATIONS

```
Ember.ENV.RAISE_ON_DEPRECATION = true
Ember.LOG_STACKTRACE_ON_DEPRECATION = true
```

Useful App Stuff

```
HANDLEBARS

{{debugger}}
{{log record}}

IMPLEMENT A Ember.onerror HOOK TO LOG ALL ERRORS IN PRODUCTION

Ember.onerror = function(error) {
  Em.$.ajax('/error-notification', 'POST', {
    stack: error.stack,
    otherInformation: 'exception message'
  });
}
```

Ember Chrome Extension

- DEMO TIME!!!!
- https://github.com/tildeio/ember-extension

Active Model Serializers

https://github.com/rails-api/active_model_serializers

```ruby
class PostSerializer < ActiveModel::Serializer
  embed :ids, include: true

  attributes :id, :title, :body
  has_many :comments
end
```

```json
{
  "post": {
    "id": 1,
    "title": "New post",
    "body": "A body!",
    "comment_ids": [ 1, 2 ]
  },
  "comments": [
    { "id": 1, "body": "what a dumb post", "tag_ids": [ 1, 2 ] },
    { "id": 2, "body": "i liked it", "tag_ids": [ 1, 3 ] },
  ],
  "tags": [
    { "id": 1, "name": "short" },
    { "id": 2, "name": "whiny" },
    { "id": 3, "name": "happy" }
  ]
}
```

JSON API

http://jsonapi.org/format/

"JSON API" is a JSON-based read/write hypermedia-type designed to support a smart client who wishes to build a data-store of information.

Ember Data Alternatives

- **Ember-Model** (https://github.com/ebryn/ember-model)
- **Ember-Resource** (https://github.com/zendesk/ember-resource)
- **Ember-RESTless** (https://github.com/endlessinc/ember-restless)
- **Emu** (https://github.com/charlieridley/emu)
- **Ember-REST** (https://github.com/cerebris/ember-rest)
- **Ember.js Persistence Foundation** (http://epf.io)

Ember in the Wild

- discourse.org
- zendesk.com
- livingsocial.com
- groupon.com
- squareup.com
- yapp.us

Resources

- http://emberjs.com
- http://discuss.emberjs.com (forum, built with ember)
- http://emberwatch.com (resource aggregator)
- https://peepcode.com/products/emberjs (official screencast 12$)
- http://eviltrout.com (blog)
- http://ember-doc.com/ (searchable docs)
- http://emberaddons.com (Ember add-ons)
- http://www.cerebris.com/blog/ (blog)
- https://github.com/dgeb/ember_data_example (cannonical Ember Data / rails example)
- https://github.com/rails-api/active_model_serializers (amazing work being done here)
- http://jsonapi.org (and here)