Backbone.js

Introduction Script

Me: “Hi, my name is Chad Maughan and I’m a Java Developer.”

You: (in unison) “Hi, Chad.”

Me: “It’s been 16 hours since my last line of Java code.”

Real Introduction

- About Me
  - Name: Chad Maughan
  - Position: Sr. Software Engineer
  - Portfolio: General Authority
  - (Core) Language: Java
  - Awesome: Yes
  - Backbone.js expert: No

Overview

- Jeremy Ashkenas
  - Dude who built CoffeeScript
- Feather-light
  - Under 4 KB
- Back-end agnostic
- Plays nice with other libraries
  - jQuery, RequireJS, handlebars.js
- So well architected, it's like getting a hug from your Grandma.

Overview

"Backbone supplies structure to JavaScript-heavy applications by providing ***models*** with key-value binding and custom events, ***collections*** with a rich API of enumerable functions, ***views*** with declarative event handling, and connects it all to your existing application over a RESTful JSON interface."

-Jeremy Ashkenas

Why?

Because:

- We're coding in Javascript
- We're ninjas
- We're allergic to callback spaghetti
- We don't like finding and updating concrete DOM elements
- jQuery alone isn't enough

Open Web Stack

- Backbone.js
  - MVC, REST, History, Pubsub
- Less
  - Enhanced CSS
- Handlebars.js
  - Javascript templating
- jQuery
  - DOM, AJAX, Events
- RequireJS
  - Structure, Modularity, Scalability, Compression

MV[RC](C*)

- Interwebs MVC definitions tend to be diluted
- For MVC purists, Backbone is not
  - Backbone View is really more of a controller
- ‘C’ originally stood for collections
- As of 0.5.0 release, Controller is now Router

Modules

- Events
- **Model**
- **Collections**
- **Router**
- History
- Sync
- **View**
- Utility

Events

- Bind a callback function to an object
  - Allows object to bind & trigger custom events
- Event specified by an arbitrary string
  - `model.bind('change', this.render)`
- Use colon convention to namespace events
  - change:selection
  - poll:start
- Supply context value for *this* (3rd parameter)
  - `model.bind('change', this.render, this)`

Events

You can also use Backbone for global events

```
// global event ‘bus’
Backbone.Events.trigger();
Backbone.Events.bind();
```

Events

```
var object = {};

_.extend(object, Backbone.Events);

object.bind("alert", function(msg) {
   alert("Triggered " + msg);
});

object.trigger("alert", "an event");
```

Model

- initialize (constructor)
  - Pass initial values of the attributes
- get/set

```
note.get("title");
note.set({title: "October 12", content: "Lorem Ipsum "});
```

- escape

```
var hacker = new Backbone.Model({
        name: "<script>alert('xss')</script>" });
alert(hacker.escape('name'));
```

Model

- defaults

```
var Meal = Backbone.Model.extend({
     defaults: {
          "appetizer": "caesar salad",
          "entree": "ravioli",
          "dessert": "cheesecake" } });
alert("Dessert will be " + (new Meal).get('dessert'));
```

- toJSON
  - Return a copy of the attributes for stringification
- validate

```
var Chapter = Backbone.Model.extend({
     validate: function(attrs) {
          if (attrs.end < attrs.start) {
               return "can't end before it starts";
          }
     }
});
```

Model

- fetch

```
setInterval(function() {
    channel.fetch();
}, 10000);
```

- url
  - Return relative URL where model's resource
  - /[collection.url]/[id] (if with collection)
  - /[urlRoot]/id (not part of collection)
- urlRoot
  - Specify if not part of collection

Model

```
var Sidebar = Backbone.Model.extend({
    promptColor: function() {
        var cssColor = prompt("Please enter a CSS color:");
        this.set({color: cssColor});
    }
});

window.sidebar = new Sidebar;
sidebar.bind('change:color', function(model, color) {
    $('#sidebar').css({background: color});
});

sidebar.set({color: 'white'});
sidebar.promptColor();
```

Collections

- Ordered sets of models
- Any event triggered on a model in a collection will also be triggered on the collection
- Bind "change" events to be notified when any model in the collection has been modified, listen for "add" and "remove" events, fetch the collection from the server
- Check out the underscore.js functions as well

Collections

- model
  - Specifies the class of model the collection holds
- Initialize

```
var tabs = new TabSet([tab1, tab2, tab3]);
```

- add

```
var ships = new Backbone.Collection;
ships.bind("add", function(ship) {
 alert("Ahoy " + ship.get("name") + "!");
});
ships.add([ {name: "Flying Dutchman"}, {name: "Black Pearl"} ]);
```

- remove
  - Remove a model or array of models
  - Fires ‘remove’ event

Collections

- get

```
var book = Library.get(110);
```

- comparator (woop!)

```
var Chapter = Backbone.Model;
var chapters = new Backbone.Collection;

chapters.comparator = function(chapter) {
    return chapter.get("page");
};
```

- sort
  - Force a collection to re-sort itself

Router

- Provides methods for routing client-side pages and connecting them to actions and events
- *:param* url parts
  - match a single URL component between slashes
- **splat* url parts
  - match any number of URL components.
- Uses hashtags or HTML5 History API
- Manually change URL
  - `router.navigate(fragment, [triggerRoute])`

Router

```
var Workspace = Backbone.Router.extend({
  routes: {
    "help": "help",              // #help
    "search/:query": "search”  // #search/kiwis
  },

  help: function() { ... },

  search: function(query, page) { ... }

});
```

History

- Handle *hashchange* events or *pushState*
- Created automatically (if you use Routers)
- HTML5 pushState support is opt-in
  - Older browsers use hash-based URL fragments
  - Start with options
    - `Backbone.history.start({pushState: true})`

Sync

- Called when Backbone attempts to read/save a model to the server
- Uses jQuery .ajax
  - Override to use WebSockets, XML transport, or Local Storage
- `.sync(method, model, [options])`
  - method – CRUD (create, read, update, delete)
  - model – model to be saved
  - Options – success and error callbacks
- Called directly on Model.save(), .fetch(), .delete()

Sync

- Maps CRUD to REST for you
  - **create → POST** /collection
  - **read → GET** /collection[/id]
  - **update → PUT** /collection/id
  - **delete → DELETE** /collection/id

View

- Allows for any templating library
  - Handlebars.js (open web stack)
- Helps you break up your interface into logical views
- Backed by models that are updated independently
- RequireJS text file dependency
  - `define(['text!../templates/item.html'] …`

View

- Model Changes
  - Without Backbone
    - find in JSON,
    - look up element in DOM,
    - update HTML
  - With Backbone
    - Bind *render* function to model's *change* event and it's done automagically for you

View

- initialize
  - attached directly to the view:
    - model, collection, el, id, className, and tagName
- el
  - DOM element of view
    - Created from view's properties
      - tagName (default), className, id
    - Attach to existing DOM element
      - `var BodyView = Backbone.View.extend({ el: 'body' });`

View

- $
  - If included, each view has a $ function that runs queries ***scoped within*** the view's element
- render
  - You override with code that renders the view template from model data and updates `this.el`
  - return this for chained calls

View

“Backbone is agnostic with respect to your preferred method of HTML templating. Whatever templating strategy you end up with, it's nice if you *never* have to put strings of HTML in your JavaScript.”

Backbone.js Documentation

“Amen, brother.”

Chad Maughan

Utility

- Single method noConflict()
- Returns Backbone object to it's original value
- Useful for embedding Backbone in existing website where you don't want to clobber an existing instance of Backbone.

```
var localBackbone = Backbone.noConflict();
var model = localBackbone.Model.extend(...);
```

More Than One Way To Do It

"It's common for folks just getting started to treat the examples listed on this page as some sort of gospel truth. In fact, Backbone.js is intended to be fairly agnostic about many common patterns in client-side code."

Backbone.js Documentation

Leeeeroy Jenkins

- Go git the code
  - git clone https://chadmaughan@github.com/chadmaughan/backbone-ninja.git
- Or download it the less cool way
  - https://github.com/chadmaughan/backbone-ninja

Leeeeroy Jenkins

- Firefox – just works
- Chrome
  - OSX
    - open –b com.google.chrome –args --allow-file-access-from-files
  - Windows
    - chrome.exe --allow-file-access-from-files
- IE – seriously?
- Python
  - python –m SimpleHTTPServer
- Node.js
  - http-server /path/project (after 'npm install http-server')