# A Blind Date with Backbone.js

Rails, Javascript, and Modern Web Apps

Dennis Kuczynski
9mmedia

# Javascript?

# Or Javascript?

# Javascript Has the Power!

- This presentation uses Javascript


- "Javascript: The Good Parts"
  - Douglas Crockford
- "Learning Javascript Design Patterns"
  - Addy Osmani
- https://github.com/rmurphey/js-assessment
  - Rebecca Murphey

# BACKBONE.JS

"Backbone.js gives structure to web 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 API over a RESTful JSON interface."

http://backbonejs.org/

# BACKBONE.JS

"When working on a web application that involves a lot of JavaScript, one of the first things you learn is to stop tying your data to the DOM. It's all too easy to create JavaScript applications that end up as tangled piles of jQuery selectors and callbacks, all trying frantically to keep data in sync between the HTML UI, your JavaScript logic, and the database on your server. For rich client-side applications, a more structured approach is often helpful.."

http://backbonejs.org/

# BACKBONE.JS

"With Backbone, you represent your data as Models, which can be created, validated, destroyed, and saved to the server. Whenever a UI action causes an attribute of a model to change, the model triggers a "change" event; all the Views that display the model's state can be notified of the change, so that they are able to respond accordingly, re-rendering themselves with the new information. In a finished Backbone app, you don't have to write the glue code that looks into the DOM to find an element with a specific id, and update the HTML manually — when the model changes, the views simply update themselves."

http://backbonejs.org/

# BACKBONE.JS

- Organizes your Javascript
- Gets out of your way
- Lightweight

(6.3KB, Packed and gzipped)

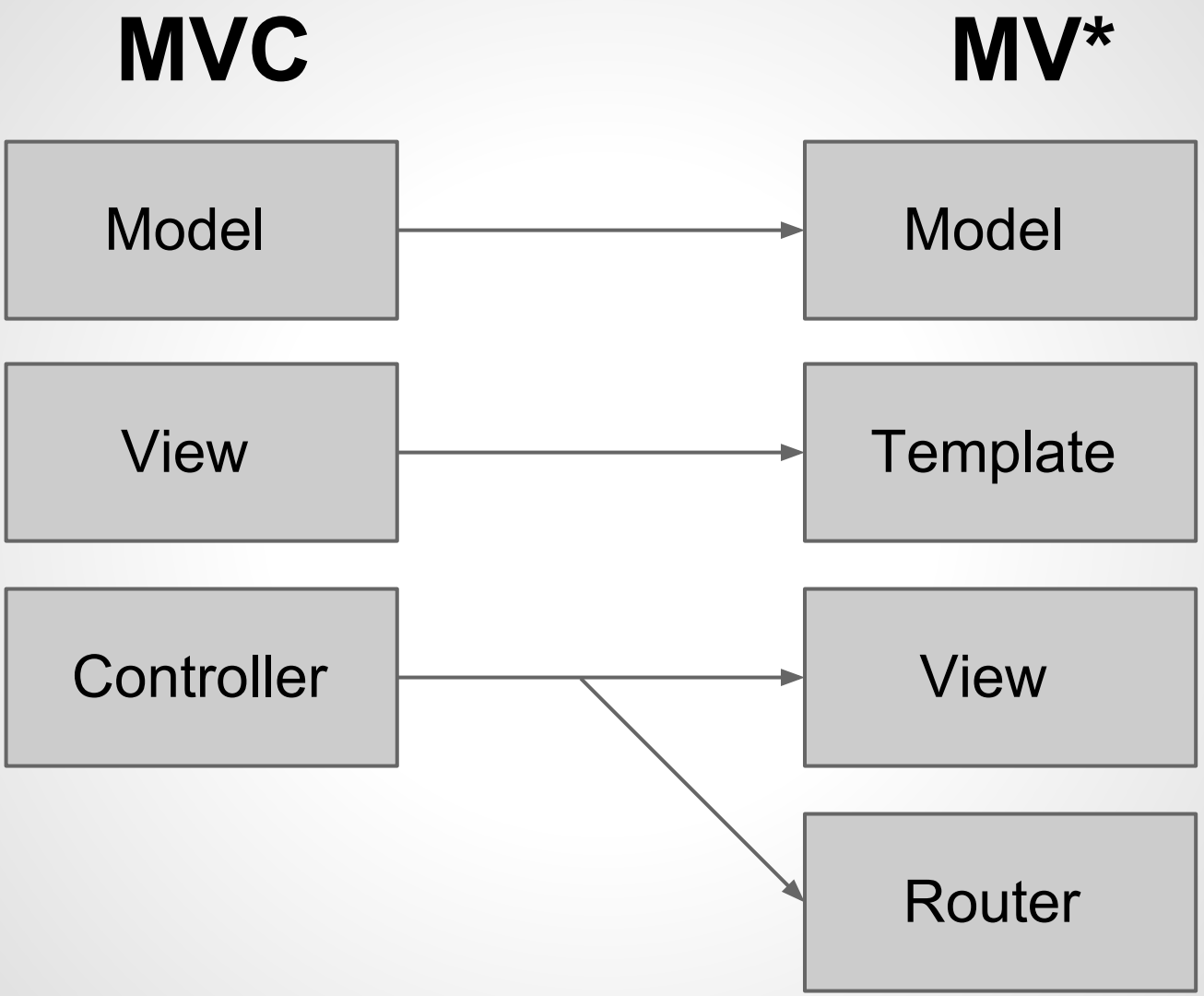
(Annotated Source Code)
http://backbonejs.org/docs/backbone.html

# It's another JS MVC Framework

- (or and MV* Framework...)

- http://angularjs.org/
- http://emberjs.com/
- http://knockoutjs.com/
- http://batmanjs.org/
- ...

- http://addyosmani.github.com/todomvc/


MVC
MV*
Model
View
Controller
Model
Template
View
Router

# Models

```
class Todo < ActiveRecord::Base

  attr_accessible :title, :completed

  def toggle
    completed = true
    save!
  end

end
```

```
app.Todo = Backbone.Model.extend({

  defaults: {
    title: '',
    completed: false
  },

  // Toggle the `completed` state
  toggle: function() {
    this.save({
      completed: !this.get('completed')
    });
  }

});
```

# Collections

```
todos = Todo.all

completed = Todo.where('completed = true')
```

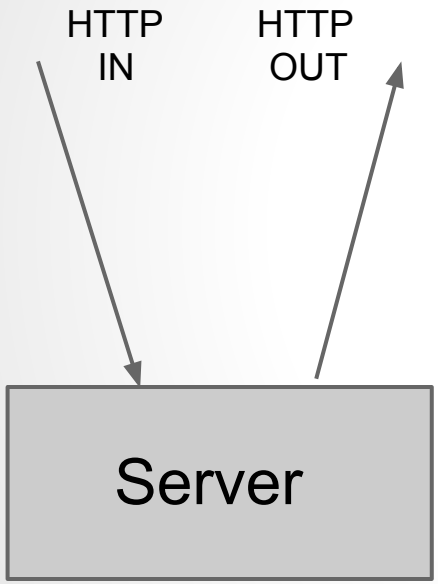
```
var TodoList = Backbone.Collection.extend({
  model: app.Todo,

  completed: function() {
    return this.filter(function( todo ) {
      return todo.get('completed');
    });
  }
});
```

# Template

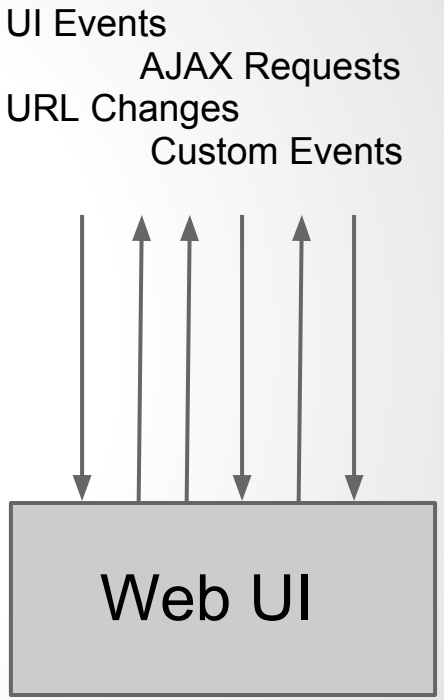
```
<div>
  <h1>Todo</h1>
  <p>Name: <%= @todo.name %></p>
  <p>Completed:
    <%= @todo.completed %>
  </p>
</div>
```

```
<div>
  <h1>Todo</h1>
  <p>Name: <%= todo.get('name') %></p>
  <p>Completed:
    <%= todo.get('completed') %>
  </p>
</div>
```

# Rails




# Backbone




A typical server responds to HTTP requests, while a web UI responds to events. A Backbone Router responds to URL Changes. A Backbone View responds to UI events.

# Router

```
var Workspace = Backbone.Router.extend({

  routes:{
    'show: 'showTodo'
  },

  initialize: function(options) {
    this.todolist_id = options.todolist_id;
  },

  showTodo: function( param ) {
    var model = new Todo({todolist_id: this.todolist_id});
    model.url = '/models';
    var view = new TodoView({model:  model.fetch({wait: true})});
    $('#container').html(view.render().el);
  }

});
```

# View

```
app.TodoView = Backbone.View.extend({
  tagName:  'li',
  template: _.template( $('#item-template').html() ),
  events: {
    'click .toggle':    'render',
  },

  initialize: function() {
    this.listenTo(this.model, 'change', this.render);
    this.listenTo(this.model, 'destroy', this.remove);
  },

  render: function() {
    this.$el.html( this.template( this.model.toJSON() ) );
    this.$el.toggleClass( 'completed', this.model.get('completed') );
    this.$input = this.$('.edit');
    return this;
  }
});
```

# But I Use Rails?

- Asset Pipeline
- CoffeeScript
- Unobtrusive Javascript
- Javascript Request Handlers

# Demos

TalkThatTalk

https://github.com/denniskuczynski/talk_that_talk

http://talkthattalk.herokuapp.com

- Basic Rails (with Turbolinks)
- Rails with Javascript Handlers
- Rails with Backbone

# Basic Rails

Pros

- Simple to develop
- Simple to understand
- Snappy with Turbolinks

Cons

- Need more dynamic behavior

This gets an application 70% of the way there.

# Rails with Javascript Handlers

Pros

- Can reuse partials for client and server
  (Big win!)

Cons

- JS templates can quickly get complicated if not just rendering partials
- JS code spread over all the view templates
- Need to take care to write modular and testable JS

This gets an application 90% of the way there.

# Rails with Backbone

Pros

- Organized Javascript
- Easy to Test
- Single page functionality can be broken out of main app

Cons

- Harder to develop
- Memory Leaks and ZOMBIES!!!
- Overkill?  Might be better off with simpler design

This gets an application 99% of the way there.


http://lostechies.com/derickbailey/2011/09/15/zombies-run-managing-page-transitions-in-backbone-apps/

# Too Much Boilerplate

- Backbone Extensions
- http://marionettejs.com/

# Do we even need Rails?

- Sinatra
  http://www.sinatrarb.com/
- Node / Express
  http://nodejs.org/
  http://expressjs.com/
- Yeoman
  http://yeoman.io/
- Rails-API
  https://github.com/rails-api/rails-api

# I Want To Learn More Backbone

http://addyosmani.github.com/backbone-fundamentals/