Agenda

- Web apps they are achangin'
- Learnin some Coffeescript
- Jumping into Backbone

Who are we?

- Chris Nelson @superchris
- Doug Alcorn @dougalcorn
- Gaslight Software

Who are you?

Why are we doing this?

How this will work

- We hate lecturing
- We dig pairing
- Learn by doing

In the beginning

- All the code in all your views
  - Giant CGI scripts
  - Perl's HTML:Mason
  - ASP/JSP
  - PHP
- Too much stuff

MVC to the Rescue

- Organized our code
- Separation of Concerns
- Easier to test

AJAX

Keep doing it on the server

- RJS
- Rails Helpers

POST http://localhost:3000/products/1/reviews 200 OK 118ms prototyp...237114650 (line 1234)

Headers Post Response

```
try {
Element.insert("reviews", { bottom: "<li>\n  <p style=\"font-weight: bold;\">Bender says</p>\n  <p>Bite
 my shiny metal TV stand.</p>\n</li>\n" });
Element.update("reviews_count", "3 Reviews");
$("new_review").reset();
Element.update("notice", "Thanks for your review!");
} catch (e) { alert('RJS error:\n\n' + e.toString()); alert('Element.insert(\"reviews\", { bottom: \"
<li>\\n  <p style=\\\"font-weight: bold;\\\">Bender says</p>\\n  <p>Bite my shiny metal TV stand.</p
>\\n</li>\\n\" });\nElement.update(\"reviews_count\", \"3 Reviews\");\n$(\"new_review\").reset();\nElement
.update(\"notice\", \"Thanks for your review!\");'); throw e }
```

http://localhost:3000

RJS error:

TypeError: Result of expression '$("new_reviewXXX")' [null] is not an object.

OK

http://localhost:3000

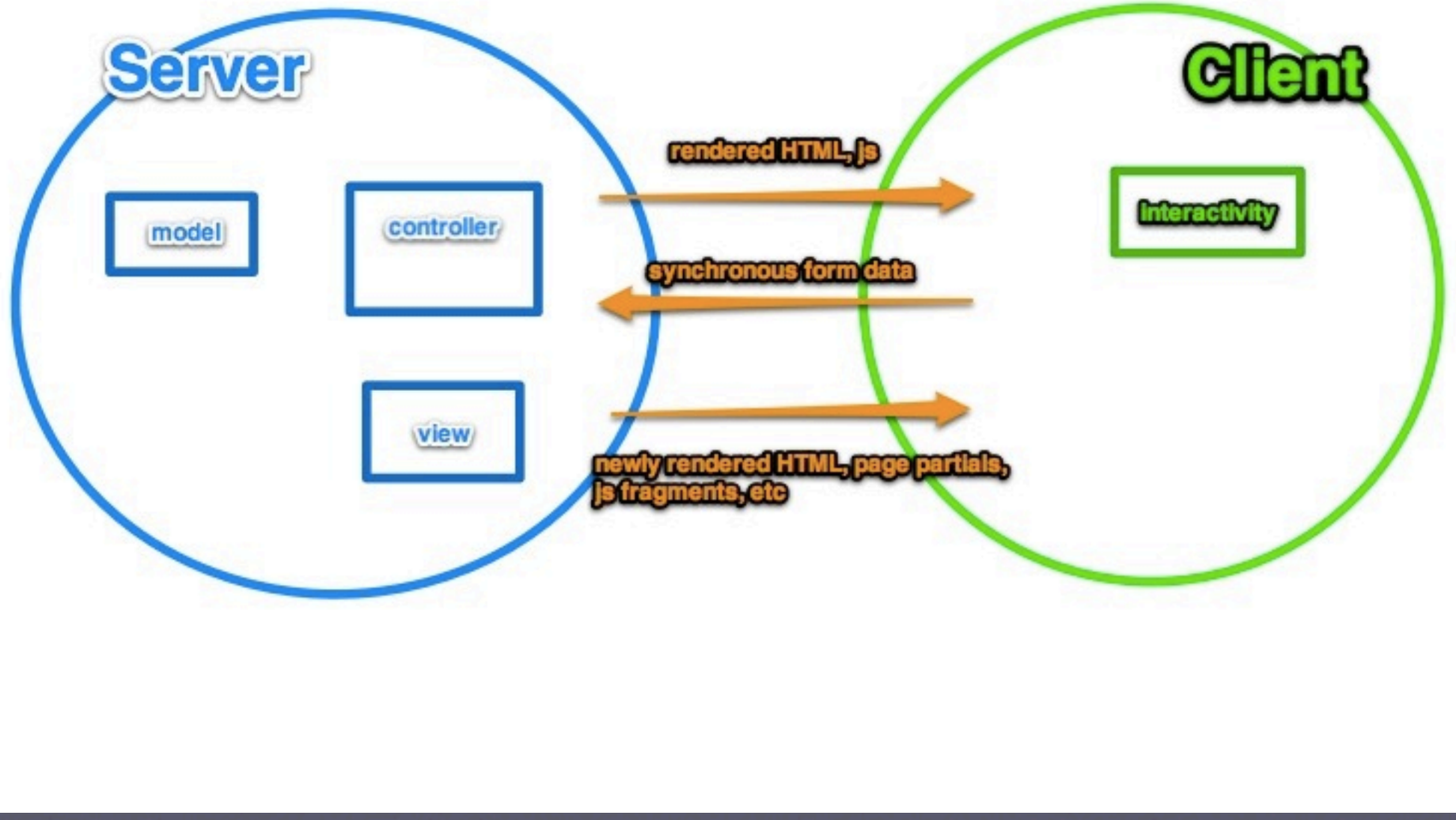
```
Element.insert("reviews", { bottom: "<li>\n  <p style=
\"font-weight: bold;\">Leila says</p>\n  <p>Wow!</
p>\n</li>\n" });
Element.update("reviews_count", "4 Reviews");
$("new_reviewXXX").reset();
Element.update("notice", "Thanks for your review!");
```

OK

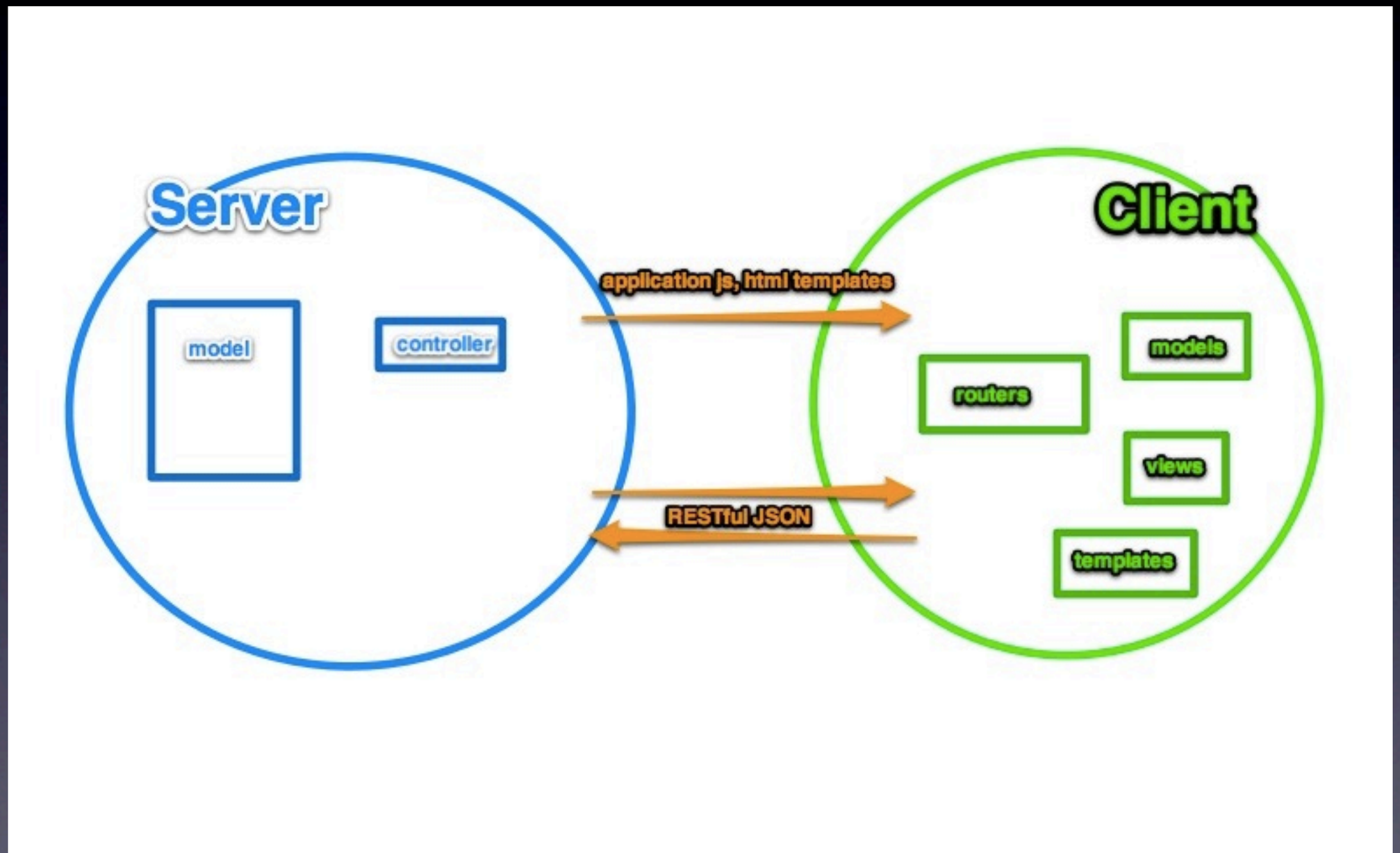
Trying to hide from javascript

- Know the language
- TDD
- Some better syntax wouldn't hurt ;)

The old


Server
model
controller
view
rendered HTML, js
synchronous form data
newly rendered HTML, page partials,
js fragments, etc
Client
interactivity

The new


Server
model
controller
application js, html templates
RESTful JSON
Client
routers
models
views
templates

Why Backbone?

Lightweight and Flexible

REST + JSON

I can refactor into it

Why Coffeescript?

The codez make me happy

What makes it different

- It's its own thing
- Limits itself to what JS does well
- compiles to clean, readable JS
- self-hosting

Installing

- (sudo?) npm -g install coffee-script
- coffee command
- coffee -c

Jasmine

- BDD framework for javascript
- Thanks, Pivotal!
- very rspec-like
- even more so in coffeescript

Jasmine

- describe “the thing your testing”, ...
- it “should be awesome”, ...
- expect(foo).toSomething

Coffeescript Koans

- cskoans.heroku.com
- git://github.com/gaslight/cskoans-online.git

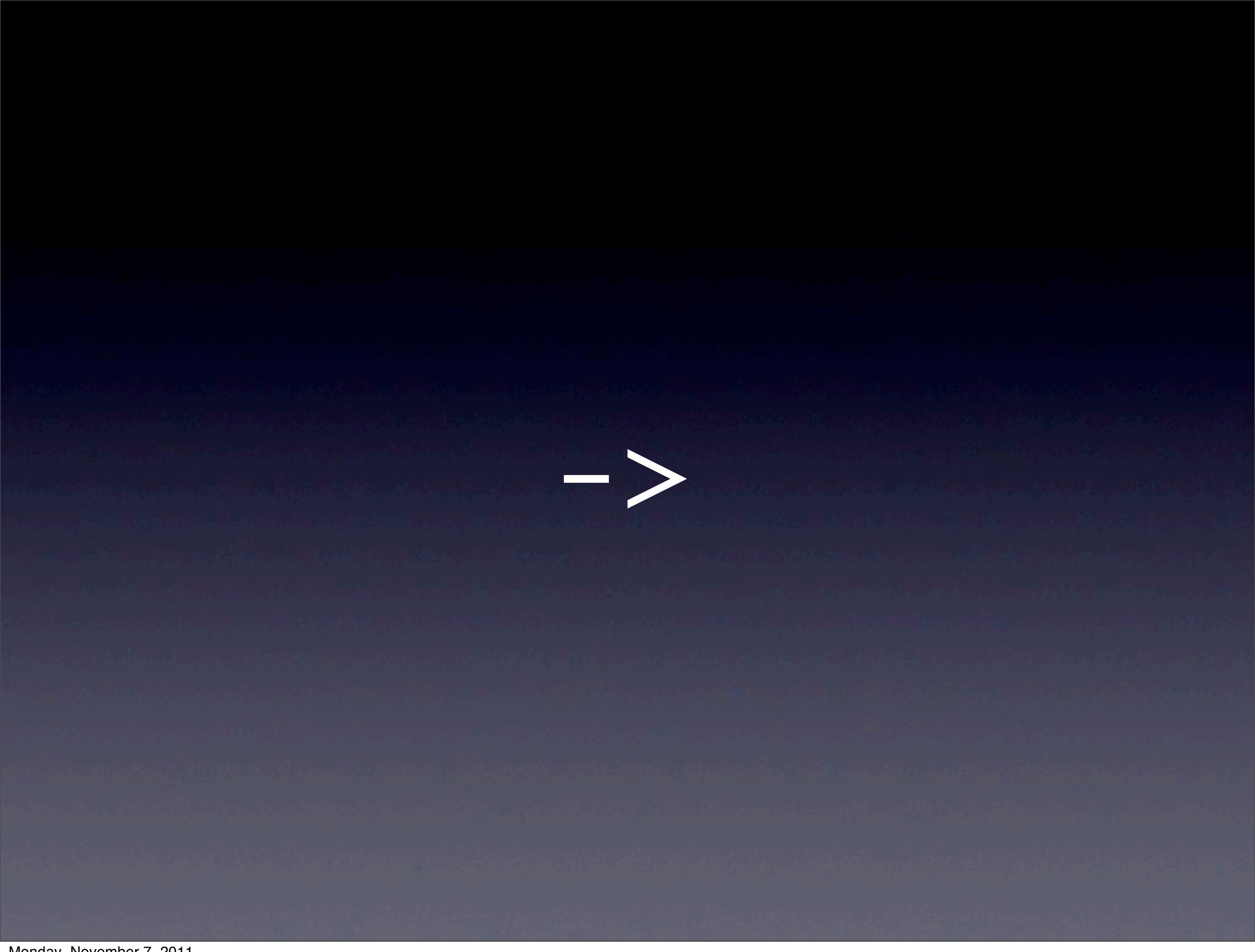
AboutExpects

var return function {) ;

significant whitespace

implicit return

function


->
Monday, November 7, 2011

function(foo, bar)

(foo, bar) ->

AboutFunctions

this.

AboutObjects

“#{string interpolation}”

foo?

foo.bar?.baz

```
var self = this;
```

=>

It's got Class and it's super()

The Cookbook

- Our humble app
- Tracks recipes and menus

rails new cookbook

Meet the asset pipeline

- build
- package
- dependencies

Under the covers

- Sprockets
  - packages, manages deps
- Tilt
  - builds

Where to put stuff

- app/assets
- lib/assets
- vendor/assets
- and in gems...

NPM

- rubygems for javascript
  - packaging, dependencies, installation
- package.json = .gemspec
- puts stuff in ./node_modules

npm_assets gem

- require_npm
  - adds to package.json
- node_modules added to asset paths
- rake npm_assets:install
- .gitignore node_modules

Adding backbone

- add bfec_generator gem to Gemfile
- bundle install
- rails g bfec
- edit superchris out of your Gemfile
- bundle install
- rake npm_assets:install

application.js.coffee

Give a hoot, don't pollute

the global namespace

Lab #0

- create a very simple spec in spec/javascripts
- Hit localhost:3000/jasmine to very things are copacetic

Backbone Views

- extend Backbone.View
- manages a DOM element
- render
  - up to you to implement

View events

- binds dom events to view functions
- specified in events property of view
- “click .save_button”:“save”

jasmine fixtures,

jasmine jQuery

```
$ git remote add bfec_class \
git://github.com/gaslight/bfec_class.git
```

```
$ git fetch bfec_class
```

RecipeView

- render
- @$
- debugging a spec
- git cherry-pick STEP_10

Put in on the page

Lab #1: MenuView

STEP_12

As a restauranteur, I would like to see a menu with a title and a description

Backbone.Model

- represent the domain
  - just like rails
- have basic persistence methods
  - just like rails
- but those methods are asynch
  - not like rails at all!

Events

- object.bind “event”, -> do something
- @trigger “anything i want”
- built-in:
  - change, error, reset, add, destroy

REST baked in

- fetch, save, destroy
- url
  - collection.url + id
  - urlRoot
  - url()
- triggers change or invokes success handler upon completion

Backbone.sync

- method
  - create, read, update, delete
- model
- options
- override sync to change per Class/object

toJSON

- result passed to JSON.stringify
- do any marshal specific code here

callbacks

- {success: -> ..., error: -> ...}
- usually success and error
- passed into async model functions

Attributes

- get “foo”
- set foo: “bar”, baz: “bing”
- triggers “change” event
- id, cid
- defaults

Recipe model/spec

STEP_13

Lab #2

Create a Menu model and spec
git cherry-pick STEP_14

Change RecipeView to use Recipe

Lab #3

- Change MenuViewSpec and MenuView to use a Menu
- Change the $ -> in application.js.coffee to do the right thing too

Instant back end

STEP_16, STEP_17

- add inherited_resources gem
- bundle install
- rails generate resource recipe title:string description:text --no-fixture --no-helper --no-assets
- rake db:migrate
- RecipesController < InheritedResources::Base
- respond_to :json, :html

jasmine ajax

STEP_18

- require it
- and the matcher

fetchin' Recipes

STEP_19

Lab #4 Fetch Menu

- write a spec for fetching menus
- change the page to fetch a menu from rails

eco

- embedded coffeescript
- uses <%, <%= (escaped) or <%- (unescaped)
- uses : and end in place of indentation

eco gem

- *.jst.eco files go in assets/templates
- call them as JST[“template_file_name”]
- pass the scope object as the arg

Change RecipeView to use a template

STEP_22

- add eco to Gemfile
- create app/assets/templates
- add require_tree ../templates to application.js.coffee
- create recipe_view_template.jst.eco
- Modify recipe_view to use it

Other templating options

- Jade
- Handlebars/Mustache
- CoffeeKup
- Writing your own Tilt::Template is not hard

Lab #5

- Use a template for MenuView

Routers

- extend Backbone.Router
- handles navigation within a Backbone application
- coordinates between multiple views

RecipesRouter

- STEP_23: failing router spec
- STEP_24: RecipesRouter
- STEP_25: extract ready
- STEP_26: add route

Lab: MenusRouter

- with a spec
- and put in on the page

Collections

- extend Backbone.Collection
- model: TheModelClass
- url property
- fetch
  - triggers reset
  - or pass success:

More about collections

- autosorted (call sort() if you really need to)
  - implement comparator(model)
    - returns the thing to sort by (not 0, -1, 1)!
- the rebroadcast their models events
- also add, remove, reset
- free goodies from underscore

Recipe collection

- STEP_27: spec
- STEP_28: RecipesCollection

Lab #6

- create a Menu collection with spec

Lab #7

- cherry-pick MenuListView spec STEP_31
- make it pass
- add a method and route to load it in MenuRouter

Pro Tip

- A view and its element should have the same life cycle
- When they don't bad things happen

Let's make it purty

- Free styling goodness from twitter bootstrap STEP_34
- Move our view to rails STEP_35
- Do it for menus!

Lab MenuEditView

- create a view which renders a menu in a form
- code the spec yourself
- make it pass
- Bonus:
  - add an edit link on MenuListView

Saving a Recipe

Lab: Saving menus

The problem

- MenuListView does not update when we edit a menu
- How do we tell the list view it needs to rerender itself?

Model/View communication

- Models trigger events, views listen to them
- No direct calling of views from models

Lab: Fix it!

- Hints:
  - Remember that model events propagate to their collection
  - but only for models that are actually in that collection ;)

What about some validation?

- Add validates_presence_of :title
- See what happens

Handling errors

- bind to “error”
- pass a function in as error:
- functions receive model, xhrResponse

Recipe error handing

Lab: Menu error handling

Problem: List view changes on errors

What's going on?

- list view listens to change event
- model fires change events when set is called

Solution

- We only want to be notified after success
- If only there was a “persisted” event
- Let’s make one!

override save in Recipe

Lab

- Do the same on Menu
- Bonus: add a superclass

Lab: new menus

- Add a new menu link
- In the router, you'll need to:
  - create a new Menu and give it to EditMenuView
  - add it to the collection after save
  - Be sure the list view is listening for adds!

Lab: destroying menus

Add Ingredients

```
$ rails g resource Ingredient description:string \
recipe:belongs_to --migration
```

Check it

- Create some ingredients tied to a recipe in the console
- hit /recipes/<id>/ingredients.json in the browser

Subviews

- Views within a view
- An element within a larger view is related to a nested model
- Eliminates need to figure out which model a DOM event goes with

Subviews

- have their el rendered by the parent view
- themselves need to be render()ed
- usually happens in parents render

Lab: displaying ingredients

Lab: click to edit ingredients

Lab: adding ingredients

Lab: removing ingredients

Lab: adding recipes to menu