Welcome to CampNG!

Hi!

Chris Nelson

chris@gaslight.co

Agenda

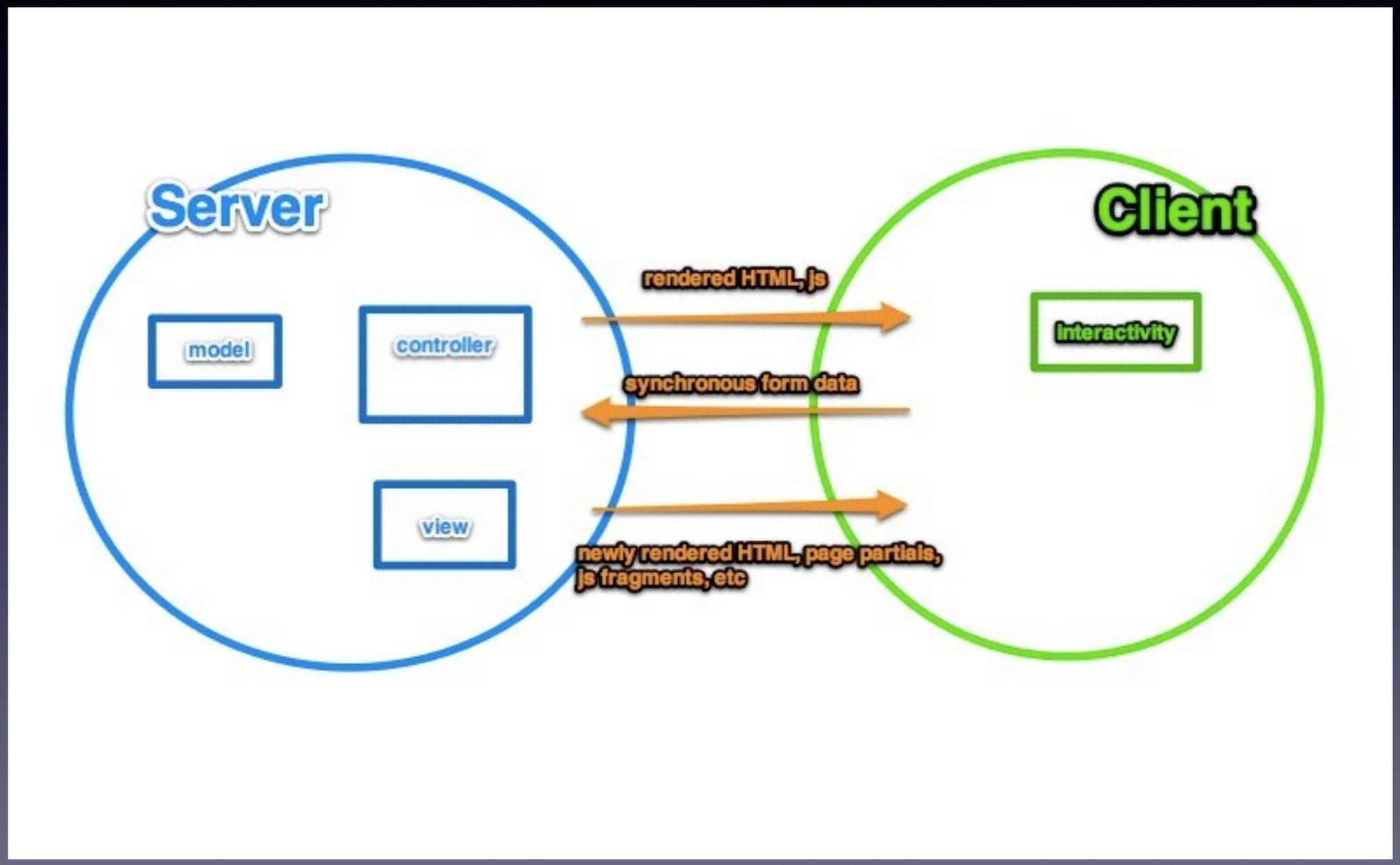
- Day 1 (AM) - Meet Angular
- Day 1 (PM) - Angular basics
- Day 2 (AM) - Angular basics continued
- Day 2 (PM) - Webpack setup and take-home project

Who are you?

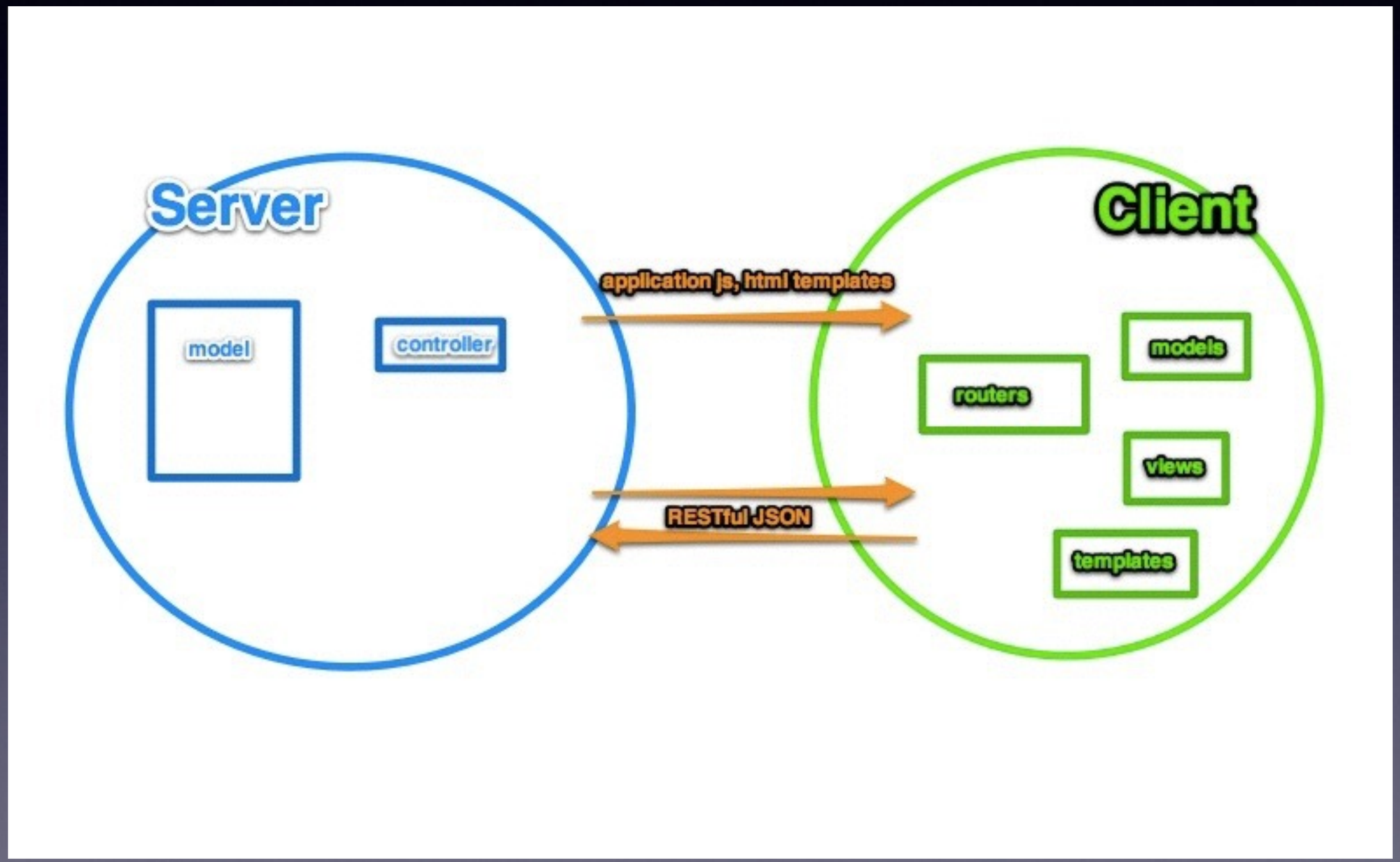
- Name
- Background with web development
- Background with Angular
- What do you hope to get out of this?

The elephant in the room

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

Angular Zen

Simplicity

Decouple all the things

Binding

Testing

HTML

Directives

- Basic building block of angular
- Custom attributes
- Custom element <thing></thing>
- CSS class

Angular expressions

- Insert data into your views
- variables
- numeric and string expressions
- filters (we'll get to these later)

Expressions cont.

- are bound
- can.be.nested
- “” for null

{{ stuff }}

Our first angular app

gaslight.github.io/campng

Welcome to JSBin!

You can make one, too!

- Start from Lab 0
- Make your page an angular app
- Add a very simple angular expression

Angular Modules

- An app or package
- Contain:
  - controllers
  - services
  - filters
  - directives
  - configuration

Modules

- create:
  - angular.module(“mod”, [deps])
- reference:
  - angular.module(“mod”)

ng-app=“module”

Controllers

- Add models and behavior to a section of your page
- 2 flavours
- just plain ole functions

ng-controller

Adds models

~~$scope~~ or this

Models = POJSOs

Behavior = functions

Should be thin(ish)

~~Manipulates the DOM~~

Let’s see a controller

ES2015 Classes as controllers

You try it!

- Have your page say: "Hi, my name is _ and I am _ years old"
- use a single object that has a name and age property

ng-repeat

ng-repeat=”item in items”

ng-repeat=”(key, value) in object”

Show me some show me some repeating!

track by expression

You try!

- Instead of just your name and age, display a list of students with names and ages
- students should be objects with a name and age property

ng-model

- Establish a two-way binding between model and view
- Changes from either side are immediately reflected

Show me some model

ng-click

- Binds to a function on scope
- on links or buttons
- Can pass things on scope as params

Show me the clicking!

ng-model-options

- updateOn - have other events, eg blur trigger update
- debounce - wait the specified time before triggering an update
- allowInvalid

ng-model-options example

Let's edit students

- Add selectedStudent property
- Add inputs for name and age
- Click on a student
  - Link or edit button, your choice
  - set selected student to the one clicked

selects in angular

- select decorates HTML select tag
- ng-options for building optons

ng-options syntaxes

- label for value in array
  - value is what gets bound to ng-model
- selected as label for value in array
  - selected is what is bound to ng-model

select example

Edit Students More

- Give students an Education Level select
- And a graduated checkbox

Showing and hiding

- ng-show
- ng-hide

Show/hide example

ng-src

- src="{{imageSrc}}"
  - browser fetches before angular loads
- ng-src="{{imageSrc}}"
  - does the right thing

Add a student gravatar image

- Add an email property
- Display an image in the list
  - only for students that have an email!
- Use http://avatars.io/email/:email_address
- Checkout avatars.io for more options

$scope

- Predates “controller as” syntax
- Can be nested
  - nested scopes inherit prototypically
- All inherit from $rootScope
- angular.element(“foo”).scope()

Beware of shadowing!

Let’s see why!

Dependencies

- Best way to share between things (controllers, directives, etc) in your app
- Singletons

Types of Dependency Providers

- Factory
  - function where return value is injected
- Service
  - returns a function called with new
- Provider
  - low level interface
- value
- constant

The ways of injection

- By argument name
- Inline (array syntax)
- thing.$inject

ngStrictDi

- directive on ng-app element
- disables injection by argument name

Make a Student Provider

- Pull your students out of your controller into a service or factory
- Inject it into your controller

Client side routing

- Handle links without page reloading
- Common features:
  - Route definition
  - view placeholder
  - programmatic transition
  - hooks to prevent transitions

So many to choose from…

- ngRouter
  - old, limited functionality
- ui-router
  - Not angular core, widely adopted
- New router
  - Not quite ready yet(?)

ui-router

- Part of AngularUI
  - separate project from angular
- Rich functionality (nested views, etc)
- Very similar API to ngRouter

Routing

- $stateProvider
  - Used in a config block to define routes
- $stateParams
  - Can be injected in controller to access params
- $state
  - manually trigger route change with $state.go()

Config blocks

- Some dependencies are configurable
- module.config is a function that can be injected with <depname>Provider

Defining a route

- $stateProvider.state("stateName", options)
  - options.url
  - can have params "/things/:id/:name"
  - these become properties of $stateParams

Route options

- controller
  - Invoked when the route is triggered
- template
  - String of markup
- templateUrl
  - External url to load template
  - script of type “text/ng-template” with an id

$stateParams

- injectable into controller
- properties for each param

ui-view

- Placeholder for what changes
- name
- autoscroll
- onload

Fruit routes

You can route too!

- Make students clickable
- Display their details when you click ‘em

Forms

- Don’t submit by default
- bind their values using ng-model
- form and inputs are available on scope by name

$location service

- Programmatically navigate to a route
- $location.path(“/foos”)

Form state

- <form name=”foo”>
  - $scope.foo is the form NOT the model
- <input name=”bar”>
  - $scope.foo.bar

State properties

- $dirty
- $pristine
- $invalid
- $error
- Also available as CSS classes

Fruit form

Form validation

- All inputs:
  - required, ng-required
  - pattern, ng-pattern
    - ng-minlength
    - ng-maxlength
- Number:
  - min, max

Moar validation

- magic properties
  - $valid
  - $invalid
  - $error
    - keys for each validation
- on form and inputs

Let’s see some!

Your turn

- Make name capitalization mandatory
- Make email valid with a regex
- ng-pattern is your friend
  - surrounded by //

ng-messages

- ng-messages - displays errors for a given object
- ng-message - displays a specific error
- Need angular-messages.js
- Need to depend on ngMessages

ng-messages-include

- Let's you have template containing reusable error messages
- Can be overridden with ng-message

ngMessages example

End of Day 1

Talking to the server

- $http
  - get(url)
  - post(url, data)
  - put(url, data)
  - delete(url)
- returns a promise
  - then(successFunc, errorFunc)

response

- then handlers receive this as argument
- data (in JSON)
- status
- headers
- statusText

Let’s see some $httping

Use your knowledge

- Add a select to choose a students favorite angular repo
- Use $http to build the list of options

Filters

- Transform an expression
- Are called like “foo | bar”
  - bar is a filter which transforms foo
- Can take parameters “foo | bar:baz”
  - bar is the filter, baz is the param

Built-in filters

- date
- currency
- json
- lowercase, uppercase
- orderBy
- number
- limitTo

The filter filter

- transforms a collection by filtering
- param is what to filter by
- strings that match or objects where any property matches
- eg stuff | filter:search

Show me the filter!

Filter students

- Add a text box
- Filter students based on what's entered

NG6 starter

Webpack

- module bundler
- manages dependencies
- rich ecosystem loaders and plugins

Let’s git started

- http://github.com/gaslight/ng6-barebones-starter
- Copy clone URL
- git clone <url>

Next steps

- cd into checked out directory
- git checkout start
- npm install
- npm install -g gulp
- gulp
- http://localhost:3000

Let’s explore

- index.html
- gulp tasks
- app.js

Crash ES2015 course

- classes
- arrow functions
- let
- template strings
- destructuring

Babel.js

exploringjs.com

ES2015 Modules

- != angular modules :(
- Precise control of namespacing
- Module = file
- Loaders are a separate spec :(

Export

- Multiple named exports
- One default export
- allows for renaming

Import

- Named
- Default
- Allows for renaming

Example time

Let’s angular

Let's ui-router

Lab: Add about

- Add an about template
- Have it load in /about

candidates module

Lab: List candidates

- Create a ListController class in a module
  - Give it a hardcoded list of candidates
  - Export it
  - Import it in candidates.js
  - Add it to the route
  - Change list.html to iterate over candidates

REST API

- node server.js
- Runs on port 8000
- Uses CORS

Use $http

Lab: CandidateService

- Doing $http right in the controller is gross!
- Lets refactor it into a CandidateService
- Create a ES2015 module that exports a class
- import it in module.js and call service
- Define a getCandidates method

Nested states

- named with a "." e.g. candidates.show
- path is relative to parent
- parent view must have ui-view

Nested routing!

Lab: show candidates

- Add a route
- Create a controller
  - it will need CandidateService and $stateParams
- Add a getCandidate method to service

Take-home

- Create candidates
- Edit candidates

Stretch yourself!

- Can you get the list to update when you create a candidate?
- What are some other interesting fields you could add to candidates?
- What about validation?