



Welcome to CampNG!

Hi!

Chris Nelson

chris@gaslight.co



Agenda

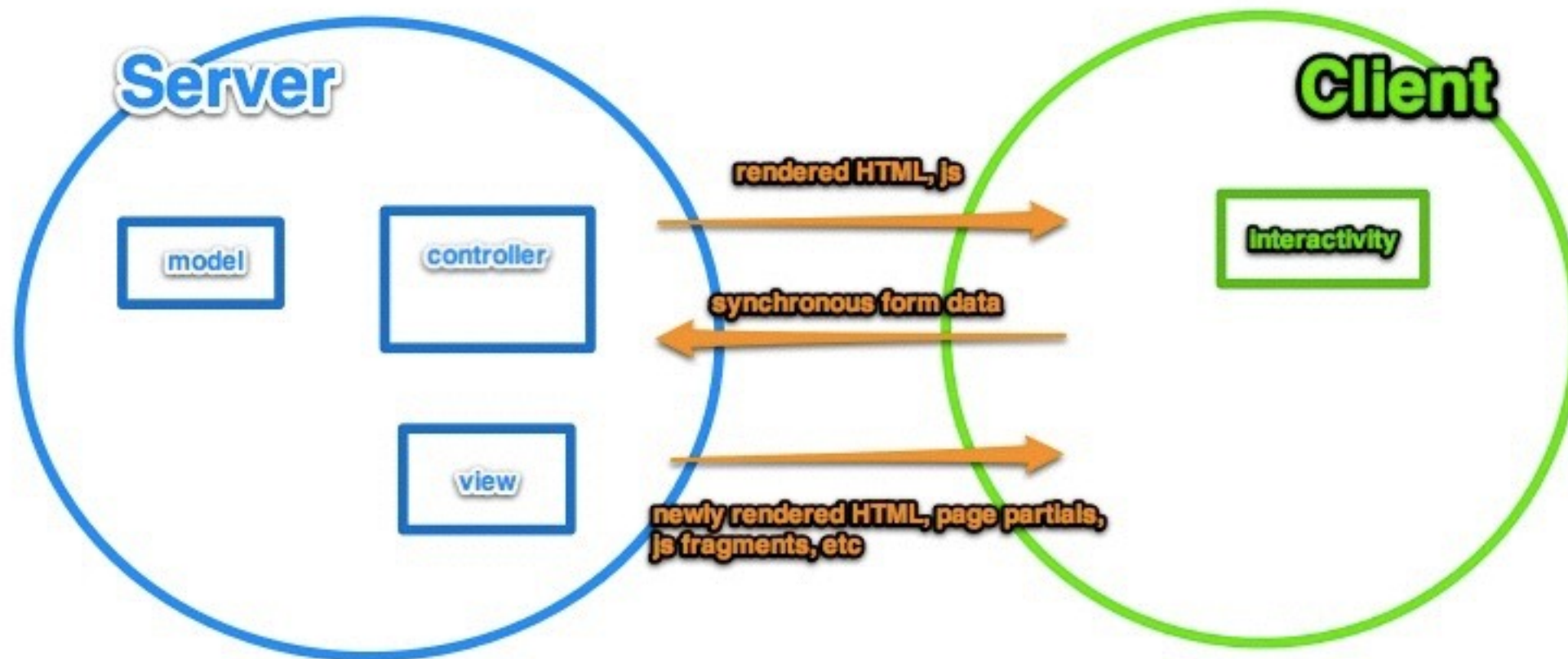
- Day 1 - Angular Basics
- Day 2 - Advanced Angular
- Day 3 - AA Cont. & Special Topics

Who are you?

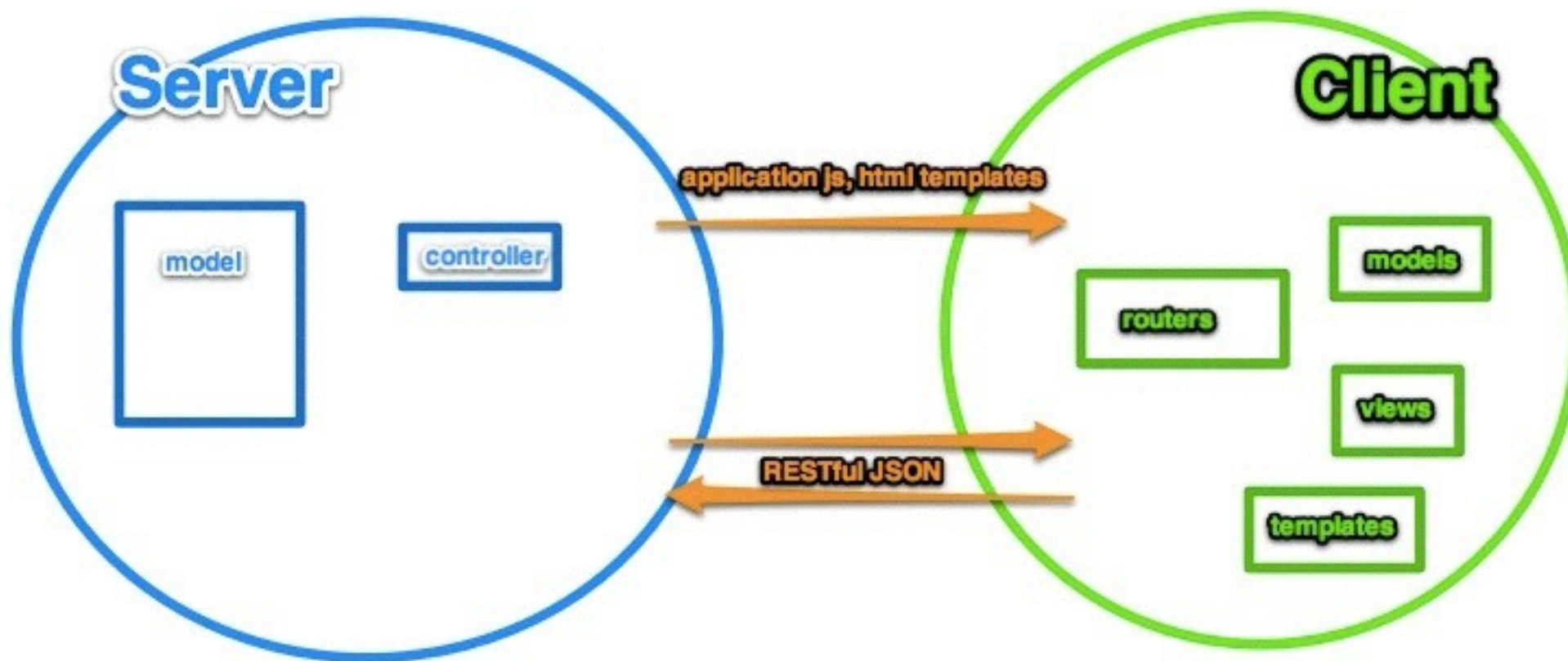
- Name
- Background with web development
- Background with Angular
- What do you hope to get out of this?

The elephant in the
room

The old



The new



Angular Zen

Simplicity

Decouple all the things

Binding

Testing

HTML

Directives

- Basic building block of angular
- Custom attributes `<div thing="wut">`
- Custom element `<thing></thing>`
- CSS class `<div class="thing">`

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

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

ES6 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

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

Some notable directives

- input, textarea, select
- ng-app
- ng-blur
- ng-change
- ng-checked
- ng-class
- ng-click
- ng-cloak
- ng-controller
- ng-disabled
- ng-show, ng-hide
- ng-model
- ng-repeat

But wait, there's more...

\$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 Dependencies

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

Meet Fruit Factory

Services with ES6 Classes

The ways of injection

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

Make Student Factory

- Pull your students out of your controller into a factory
- Inject the factory into your controller

ng-router

- Broken out from angular core in 1.2
- Client side routing for SPAs
- Need to add script and depend on ngRoute module

Routing

- `$routeProvider`
 - Used in a config block to define routes
- `$routeParams`
 - Can be injected in controller to access params
- `$location`
 - manually trigger route change with `$location.path()`

Config blocks

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

Defining a route

- `$routeProvider.when("route", options)`
- everything after “#”
- can have params “/things/:id/:name”
 - these become properties of `$routeParams`
- `$routeProvider.otherwise({redirectTo: “/route”});`

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

\$routeParams

- injectable into controller
- properties for each param

ng-view

Fruit routes

You can route too!

- Make students clickable
- Display their details when you click 'em
- Move image display into the details

ng-route limitations

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!

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

Your turn

- Make name capitalization mandatory
- Make email valid with a regex
- ng-pattern is your friend
 - surrounded by //
- Use a ng-messages-include to make things re-usable

selects in angular

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

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

Talking to the server

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

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

Fin

Creating directives

The ~~easy~~ deprecated way

- `module.directive("fooBar", function() {...})`
- Return a function(scope, element, attrs)
- element is jQuery wrapped if you jQuery
 - handy for wrapping plugins

version directive

The ~~hard~~ recommended way

- return an object
 - specifies options to specify how directive works
- there are many

The highlights

- restrict:
 - E for element
 - A for attribute
 - C for class
 - M for comment (no one uses that)

link

- a function(scope, element, attrs)
- Earlier directive form is a simplified syntax

Other things

- `template`
- `templateUrl`
- `replace`
- must have a single root element

Directive scope

- true = new scope
- {} = “isolate” scope
 - Maps attributes on directive to scope
 - @ maps attribute
 - = bind to parent scope
 - & pass in an expression

Directive controllers

- Specified with controller attribute
- Can be name or function

\$scope.\$watch

- Two args
 - An expression to watch
 - A function to execute on change
 - receives newval, oldval as params

upperCase directive

Directive with
controller

Transclusion

- `transclude: true`
- Allows directive to wrap arbitrary angular template content
- Use `ng-transclude` to specify where the original body content goes