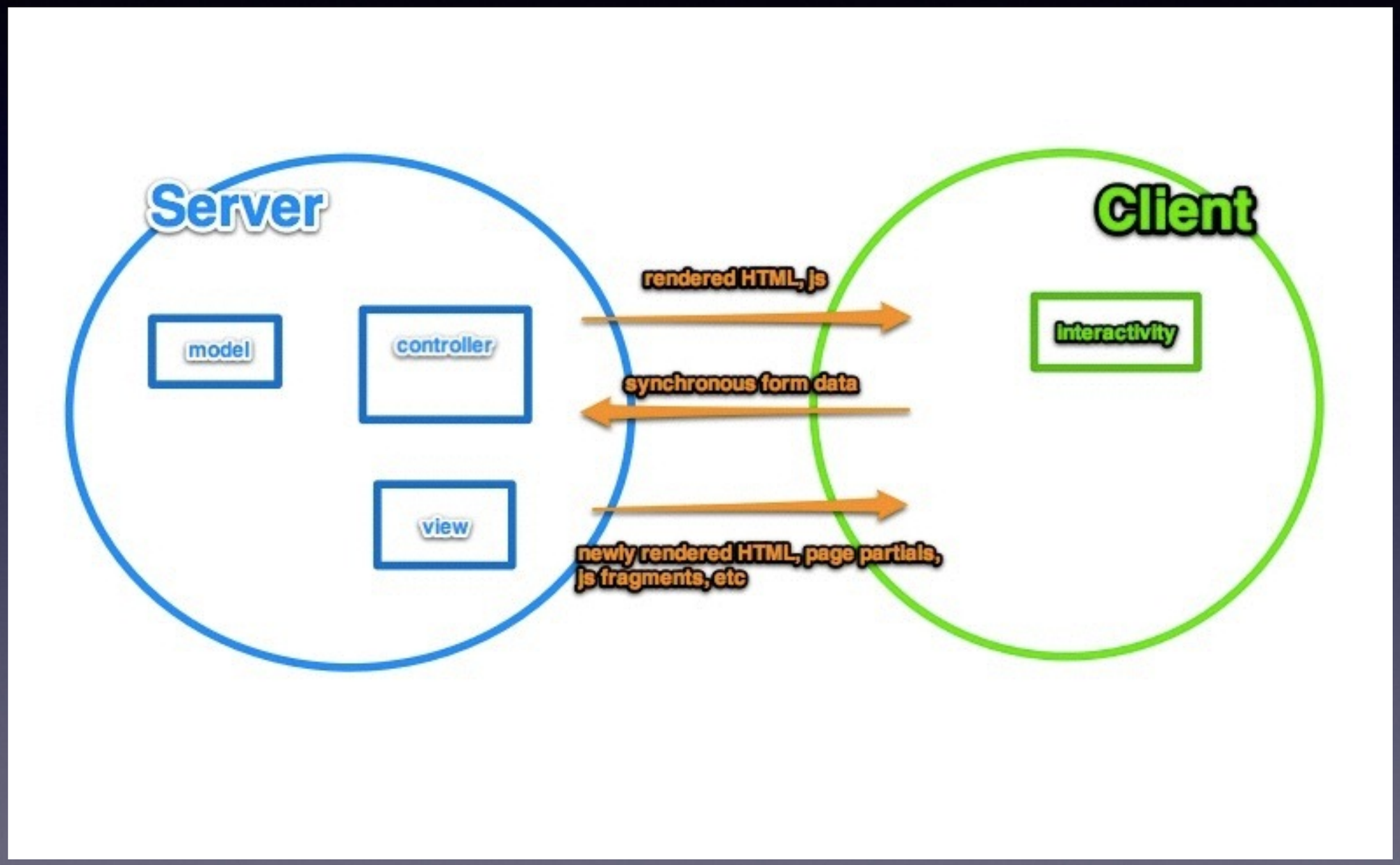
Welcome to CampNG!

Hi!

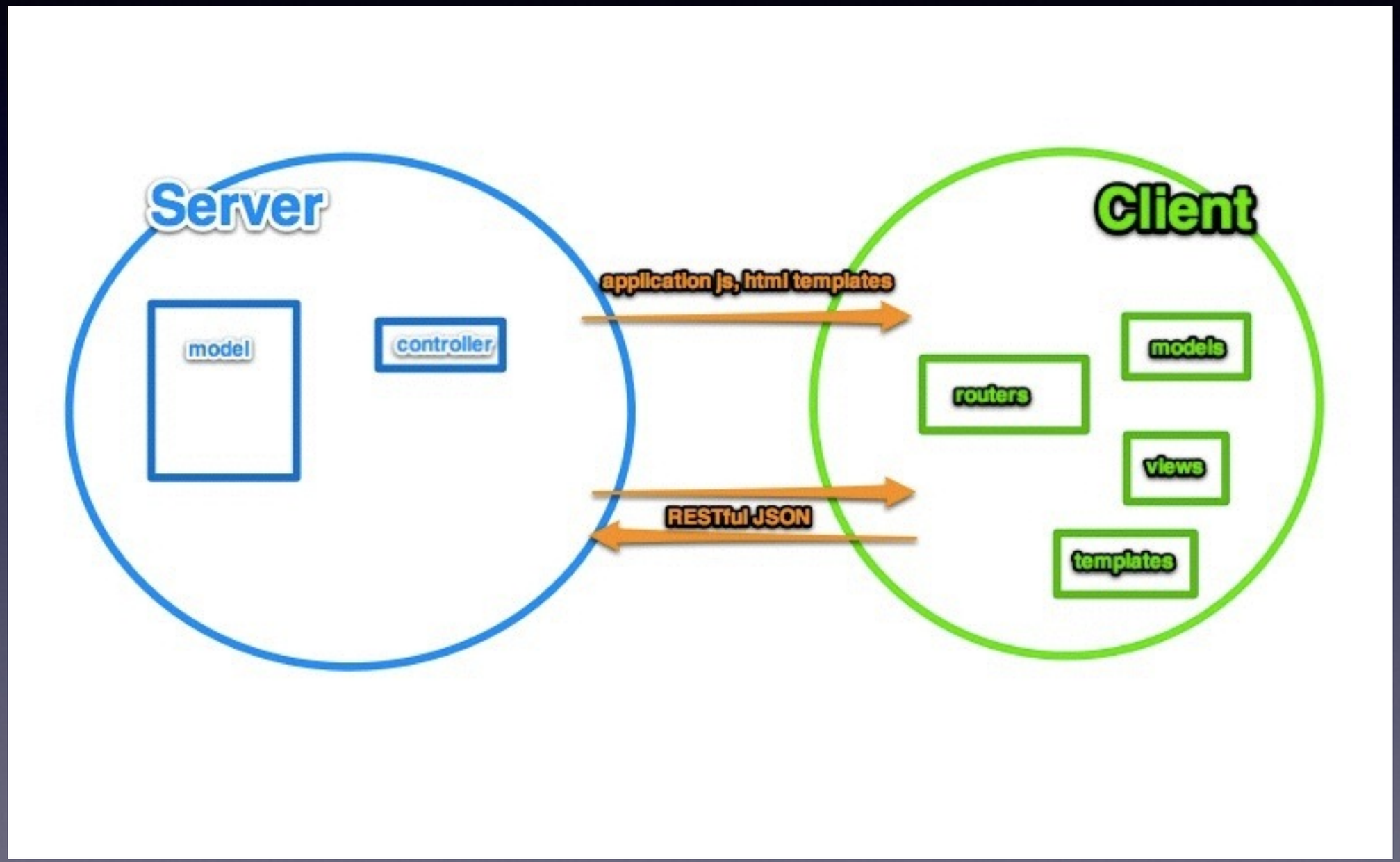
Chris Nelson

chris@gaslight.co

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

Dependency Injection

Testing

HTML compiler

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

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

You try it!

- Have your page say: "Hi, my name is _ and I am _ years old"
- Need 2 expressions and object(s) on scope
- Bonus point for single object on scope

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!

Let's edit students

- Click on a student
  - Link or edit button, your choice
- Add an input to edit the name

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
- Use http://avatars.io/email/:email_address
- Checkout avatars.io for more options

Built-in 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

So many!

But wait, there's more...

More about $scope

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

Beware of shadowing!

Let’s see why!

Modules

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

Modules

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

Show me the module!

Let's module!

- Put your app in it's own module
- Call controller on that module to make your controller

We got both kinds of injection

- By argument name
- Inline (array syntax)
- Let's see some code

Dependencies

- You can define your own!
- 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

Make Student Factory

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

Config

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

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()

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

Fruit routes

You can route too!

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

ui-router

Forms

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

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

$location.path(“/foos”)

Edit students

- Wire up the edit form
- use $location to return to show view after save

Form validation

- All inputs:
  - ng-required
  - ng-minlength
  - ng-maxlength
  - ng-pattern
- 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
- ng-pattern is your friend

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

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