CampNG Advanced

Intros

The plan

- Jasmine, protractor
- ngResource
- Directives, directives, directives
- UI Router

The unplanned

-
-

Let’s git started

- http://github.com/gaslight/campng-advanced
- Copy clone URL
- git clone <url>

Lineman

- thin wrapper around grunt
- templates to get started fast
- somewhat like yeoman

lineman commands

- lineman run
  - builds project and runs dev server
  - watches files to rebuild
- lineman spec
  - runs testem and waits for browsers to connect
  - lineman run needs to be running
  - watches tests and re-runs
- lineman spec-ci
  - runs tests in headless browser

Installing lineman

- npm install -g lineman

Next steps

- git checkout start
- npm install
- lineman run
- http://localhost:8000

Jasmine

- Go to unit test framework for Angular
- Rspec/BDD style

Jasmine basics

- describe(“a suite”, function() { … })
- it(“a suite”, function() { … })
- expect(something).toXXX(something)

Expectation matchers

- toBe
- toEqual
- toBeDefined
- toBeTruthy
- toBeFalsey
- toMatch
- toContain
- toBeLessThan
- toBeGreaterThan
- not.toXXX

Setup and teardown

- beforeEach
- afterEach
- `this` is the same object as in tests

Let’s see

angular-mocks.js

- Extra goodness for testing angular
  - module - sets the module in a test case
  - inject - lets angular give you deps in your test case

$controller

- Service that creates controllers
- Usually don’t need to access explicitly
- Handy for tests

Controller spec example

ngResource

- $resource(url, [paramDefaults], [actions], [options])
- Gives you a class that interacts with REST back end

$resource operations

- Resource.query
- Resource.get
- instance.$save
- instance.$delete
- instance.$remove

Magic stubs

- Resource.get and Resource.query return right away
- Populated later
- Change detection and bindings make it work
- Contrast with promises

Examples

- Candidate resource
- server.js code

$httpBackend

- From angular-mocks.js
- expectGET, expectPOST, etc
- whenGET, whenPOST, etc

Resource spec example

Lab 1

- git checkout lab1_start
- Create a jasmine spec for new Candidate
- in candidate_resource_spec
- On $save it should POST to /candidates
- Use $httpBackend.expectPOST

spies

- spyOn(object, “function”)
- expectations
  - toHaveBeenCalled(With)
  - mostRecentCall.args

spy example

Lab 2

- Create jasmine spec for NewCandidateController
- Create a new Candidate in controller
- save should call $save on candidate
- spies can help

protractor

- end to end testing for angular
- wrapper around web-driver

Installing protractor

- npm install -g protractor
- webdriver-manager update

protractor API

- browser
  - navigate using browser.get
- element
  - interact with elements on the page
  - click, sendKeys

protractor API

- locators - for finding elements
- by
  - id
  - css
  - binding
  - model

Protractor example

- lab3_start
- spec-e2e/candidates_spec.js

Running protractor

- webdriver-manager start
- protractor config/spec-e2e.js

Lab 3

- Protractor spec for creating a Candidate
- Need template, controller, and route
- Should end up back on show view

We have a problem

- Create a new candidate works
- But the list doesn’t update :(

Events

- Pub sub is built in
- Use methods on any $scope
- $rootScope can be injected anywhere

$broadcast/$emit

- $broadcast goes to this scope and down
- $emit goes to this scope and up
- both take an event name an args
- Easiest to do $rootScope.$broadcast

$on

- Subscribes to an event
- Pass name, event handler
- Event handler receives event, any args

Event example

Lab 4

- checkout lab4_start
- Update the side list when the menu updates
- Use events
- Make the spec pass
- Need to restart lineman to see it fail :(

Creating filters

- module.filter(“name”, function(…) {…}
- returns a function which takes param(s) and returns filtered value

checkout lab5_start

hello filter and spec

$sce

- $sce.trustAsHtml
- $sce.getTrustedHtml

Lab 5

- Make a markdown filter
- Write a jasmine spec for it
- Use markdown.toHTML
- Use $sce to make it trusted

Services, factories, and providers

Creating directives

- module.directive(“name”, function…
- function can get dependencies injected
- returns a directive definition object

Directive naming

- strip any data- or x- prefix
- convert :, -, or _ separated to camelCase
- resulting name is use to find directive

Directive definition obj

- template or templateUrl
- restrict
  - E for element
  - A for attribute (default)
  - C for class
  - M for comment
  - any combination thereof

Hello directives

Your turn

- Make a directive that lets create a <hello-myname> element
- Have it print out a greeting

Directive scope

- Turns out inheriting scope is a bad idea
  - Leads to coupling
- Isolate scope to the rescue
- Scope property with a mapping object

Mapping scope

- The “interface” to your directive
- Uses attributes to set in properties on directive scope
- Prefixes specify how to map attribute to scope property
- Can be abbreviated to “@“, “=“, “&” if scope property and attribute are the same

@

- foo: “@bar”
- The foo property of scope holds the string value of the bar attribute

=

- foo: “=bar”
- creates a scope property foo
- bound to parent scope property specified by bar attribute

&

- evaluates an expression in parent scope
- foo: “&bar”
- foo property becomes a function
- bar is the expr to evaluate

Hello scopes

templateUrl

$compile

- Compiles string of html into angular
- Returns a function take takes a scope and returns an element
- Useful for testing directives and building very dynamic things

ng-bind-html

- Sets the innerHTML of it's element
- Uses $sanitize unless it receives trusted HTML
- Use in place of {{}} if you want to output HTML

Directive spec

- checkout lab6_start
- Let’s see our hello_directive_spec

Lab 6

- Make a markdown editor directive
- Directive should:
    - Pass in the markdown via scope mapping
    - Edit markdown in a textarea
    - Display the rendered markdown below
- With a jasmine spec
- Add it to newCandidate.html

link function

- link: function(scope, element, attrs, …)
- Called after the resulting element is in the DOM
- Do DOM manipulation or event listening here
- The right place for any jQuery calls
- But you need to $(element)

raty

- A jquery plugin to do star ratings
- http://wbotelhos.com/raty
- Let's see a demo!

Lab 7

- Make a raty directive
- Use a link function
- Be sure to {path: “/img”}
- Just get as far as plugin appearing

$scope.$watch

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

Lab 7a

- Add a rating scope mapping
- watch the rating mapping
- call $().raty(“score”, newValue)
- Use an input to check your work

scope.$apply()

- Triggers a $digest() cycle
    - This is where dirty checking happens
        - Normally not needed
        - Except to coordinate with external APIs
- Takes a function as arg

Lab 8

- Pass a click function into raty
- Use scope.$apply to update the rating property and have angular know about it
- Raty directive should work!

Directive controllers

- Specified with controller attribute
- Can be name or function

Directive with controller

require

- Allows directives to collaborate
- Specifies a directive this directive needs
  - “otherDirective” should be located on this element
  - “^parentDirective” search for directive on parent elements
  - “?maybeDirective” pass null if maybeDirective isn’t found instead of throwing error
- required directives have their controller passed to requiring directive

Require example

ngModelController

- Use require: “ngModel” to get handed an ngModelController
- Exposes form component API
- Manages model and view values
- Allows for building custom form controls that first class “angular form citizens”

ngModelController

- $render
  - set this to function which gets passed the view value to render.
- $setViewValue
  - Call this to update the model controllers view value

Lab 9

- Turn raty into a “real” angular form control
- Require ngModel and get passed an ngModelController
- User $render to update raty
- Use $setViewValue to update ng-model

$formatters

- pipeline (array) of functions
- Converts model values to view value
- Each function gets passed model value, returns view value

$parsers

- pipeline (array) of functions
- Converts view value to model value
- Each function gets passed view value, returns model value

parse/format example

Lab 10

- Let’s make raty convert from percentage
- Model value is 0 to 100
- Stars are 0 to 5

Transclusion

- The inclusion of one thing in something else
- transclude: true
- Allows directive to wrap arbitrary angular template content
- Use ng-transclude to specify where the original body content goes

Transclusion example

Lab 11

- Build a dollar input group directive
- Wrap a normal input with twitter bootstrap input group formatting
- Use it to enter a candidates expected salary

Lab 12

- Add message property to scope on save and create
- Create a directive to display it
- Allow directive to take dismiss handler
- Remember “&”?

validation directives

- require ngModel
- $setValidity(validationErrorKey, isValid)
- called during $parsers function

Lab 13

- Write an ssn directive
- Should validate ssn with regex and add an ssn error in case of failure
- Use it on edit and display appropriate error message

UI Router

- A replacement for ngRouter
- Allows for nested and multiple views

Switching

- ngRouter => ui.router
- $routeParams => $stateParams
- ng-view => ui-view

$stateProvider

- very close to $routeProvider API
- when => state
- state(stateName, options)
- url goes in options

Let’s see!

ui-sref

- directive to build links
- pass a state name
  - ui-sref=“state”
- with params
  - ui-sref=“state({id: 1})”

Lab 14

- Convert the links to ui-sref

Nested routes

- Name with “parentState.childState”
- url is relative to parent state
- Add a ui-view to parent template

Example me!

Lab 15

- Candidates now have comments
- List comments on the show candidate view
- Click on a comment title to view details
- Use nested routes make it happen!

$stateChangeStart

- called when a route change is about to happen

- listener function gets event, toState, toParams, fromState, fromParams as args

- event.preventDefault() will cancel

module.run()

- pass in a function which can be injected
- Useful place to listen for $stateChangeStart event

Lab 16

- Use this knowledge to implement authentication

Angular2

All about the web components

It’s totally different!

- Web components
- ES6/Typescript
- Routing
- Injection system
- bindings

It’s not done yet!

- Some of what I will show you is already obsolete!
- Even more of it will become obsolete soon!
- Yay! Let’s get started!

Angular2

- git clone:
- https://github.com/gaslight/angular2_labs
- inside that dir
  - git clone:
  - https://github.com/angular/quickstart.git

npm install -g http-server

SystemJS

App component

ES6 Modules

Typescript

bootstrap

Woot! We can haz angular2

You try!

- git checkout start
- Make your app element <name>-app
- Where <name> is your name
- Have it say something!

Components

- The class is the “controller”
- It is in scope in the template
- Expressions are still {{}}

Let’s see!

You try!

Nested components

Super weird for loop syntax

You try!

attribute bindings

Make a sub subcomponent

Event bindings

Add a button!