thinkster ✔

A Better Way to Learn AngularJS

Congratulations on taking the plunge!

This AngularJS course is built with the intent of exposing you to the best available resources on each Angular topic. Our desire is to present these topics richly, and from a variety of vantage points, in order to afford you a more complete perspective on them.

This course is accompanied by AngularJS Tutorial: Learn to Build Modern Web Apps (http://www.thinkster.io/pick/GUIDJbpIie/angularjs-tutorial-learn-to-build-modern-web-apps).

The Course

The learning curve of AngularJS can be described as a hockey stick. Initially getting off the ground with apps featuring basic functionality is delightfully easy. However, when apps eventually grow large or complicated, structure without heed to Angular's inner workings will cause development to become awkward and cumbersome.

With AngularJS, the "Ready, Fire, Aim" learning methodology of duct taping together a handful of tutorials and a cursory glance through the documentation will lead to confusion and frustration. This curriculum is designed to properly guide you through each of the key Angular concepts thoroughly with a broad exposure to high quality content. With your eventual mastery of AngularJS, you will be able to fluently and efficiently construct large-scale applications.

Prerequisites

- Moderate knowledge of HTML, CSS, JavaScript, including the following concepts:
  - The POJO (plain old JavaScript object), including:
  - OOP, including encapsulation and inheritance
  - Object creation, prototypes
- Basic Model-View-Controller concepts
- The Document Object Model
- JavaScript functions, events, error handling

Resources

Since AngularJS is still in its infancy relative to other JavaScript frameworks, the number of encyclopaedic resources on it is still insufficient. Therefore, the curriculum will employ a healthy number of excellent blogs in order to offer a more meaty perspective on respective topics.

- Required Resources
  - AngularJS - O'Reilly Media (http://shop.oreilly.com/product/0636920028055.do) (available on Amazon (http://www.amazon.com/AngularJS-Brad-Green/dp/1449344852/ref=sr_1_1?ie=UTF8&qid=1372874049&sr=8-1&keywords=angularjs))
  - John Lindquist's egghead.io (http://www.egghead.io/)
  - egghead.io source code (https://github.com/msfrisbie/egghead-angularjs)
  - AngularJS docs (http://docs.angularjs.org/)
- Supplemental Resources
  - Jim Hoskins blog (http://jimhoskins.com/)
  - Ben Nadel blog (http://www.bennadel.com)
  - OneHungryMind (http://onehungrymind.com)
  - year of moo (http://www.yearofmoo.com)
  - Bruno Scopelliti blog (http://blog.brunoscopelliti.com/)

Part 1. Kicking the Tires

AngularJS is not a library.

thinkster

It doesn't abstract away HTML, CSS, or JavaScript. It doesn't require inheritance from proprietary types. It doesn't use one-way data binding. It doesn't treat testing as an afterthought. It doesn't require boilerplate code. And it's not that complicated.

AngularJS is a JavaScript framework that embraces extending HTML into a more expressive and readable format. It decreases emphasis on directly handling DOM manipulation from the application logic, allowing for easier testing. It employs efficient two-way data binding and sensible MVC implementation, reducing the server load of applications. It features directives, which are incredibly robust tools that are significant contributors to Angular's ubiquity.

AngularJS is self-described as 'HTML enhanced for web apps', and it is most certainly that.

Filling the Tank

We've found that the egghead.io videos are the best starting resource available, so every chapter will lead off with them. The transcribed screencasts and source code are provided along with the videos. We encourage you to follow along with them, as they make the video content much more readily digestible.

As good as the egghead videos are, they should serve only as an introductory resource. Excerpts from the O'Reilly AngularJS book and the angularjs.org documentation complement the videos as the broader and more thorough source, and should be treated as the main reference for the course.

Adjusting Your Mirrors

When descending upon an entirely new topic, it is important to frame the topic correctly before diving into the minutia.

thinkster

Read the following two entries in the AngularJS guide docs, they will give you a good idea of what you're about to get into. Don't worry about picking up on every aspect of the topics they glaze over, all of these will be covered thoroughly in subsequent lessons.

- [ ] **AngularJS Overview (http://docs.angularjs.org/guide/overview)**
- [ ] **Introduction to AngularJS (http://docs.angularjs.org/guide/introduction)**

Revving the Engine

- [ ] Before we get on with it, this post, Things I Wish I Were Told About Angular.js (http://ruoyusun.com/2013/05/25/things-i-wish-i-were-told-about-angular-js.html), goes over a handful of topics that might be helpful in building the appropriate mental models while consuming the Angular curriculum. Some, probably most, of the terms will bounce right off you until you have gone through that section of the course, but it should provide valuable context when approaching a new topic.

Off to the races!

Part 2: Taking It for a Spin

One of the awesome things about AngularJS is its usability right out of the box. Very little information about how the framework operates is needed to get up and running with your first application. With that in mind, go ahead and tear through the first five egghead videos:

- [ ] **AngularJS - Binding (/pick/51cfc59a067d471763000002/angularjs-binding)**

thinkster

- [ ] **AngularJS - Controllers (/pick /51d269941d7225e2e7000001/angularjs-controllers)**
- [ ] **AngularJS - The Dot (/pick /51d26ba41d72252b87000002/angularjs-the-dot)**
- [ ] **AngularJS - Sharing Data Between Controllers (/pick /51d26e1f1d7225cb4b000003/angularjs-sharing-data-between-controllers)**
- [ ] **AngularJS - Defining a Method on the Scope (/pick /51d26fd51d7225edb4000004/angularjs-defining-a-method-on-the-scope)**

Readings

We're going to crack the cover of AngularJS (http://shop.oreilly.com/product/0636920028055.do) for the first time.

- [ ] **Read all of Chapter 1. Introduction to Angular JS**
- [ ] **Read from the beginning of Chapter 2. Anatomy of an AngularJS Application to the section "Templates and Data Binding - Publishing Model Data with Scopes"**

AngularJS Documentation

The following set of guides will serve to reinforce many of the topics just covered, and explain some new ones in detail:

- [ ] **Bootstrap (http://docs.angularjs.org/guide/bootstrap)**
- [ ] **Expressions (http://docs.angularjs.org/guide /expression)**
- [ ] **Forms (http://docs.angularjs.org/guide/forms)**
- [ ] **About MVC in Angular (http://docs.angularjs.org/guide /dev_guide.mvc)**
- [ ] **Understanding the Model Component (http://docs.angularjs.org/guide /dev_guide.mvc.understanding_model)**

thinkster

- [ ] **Understanding the Controller Component (http://docs.angularjs.org/guide/dev_guide.mvc.understanding_controller)**
- [ ] **Understanding the View Component (http://docs.angularjs.org/guide/dev_guide.mvc.understanding_view)**
- [ ] **Data Binding In Angular (http://docs.angularjs.org/guide/dev_guide.templates.databinding)**

Part 3: Filters

Filters are a simple but powerful tool in Angular, used primarily to format expressions in bindings in views and templates.

These three egghead videos serve to cover the basics nicely:

- [ ] **AngularJS - Filters (/pick/51d271201d7225df42000005/angularjs-filters)**
- [ ] **AngularJS - ngFilter (/pick/51d27422f76892c290000003/angularjs-ngfilter)**
- [ ] **AngularJS - Built-In Filters (/pick/51d2765bf76892d959000004/angularjs-built-in-filters)**

Readings

- [ ] **In Chapter 2: "Anatomy of an AngularJS Application", read the section "Formatting Data with Filters"**

AngularJS Documentation

Finally, the Angular guides offer a bit more depth on filters:

- [ ] **Understanding Angular Filters (http://docs.angularjs.org/guide/dev_guide.templates.filters)**

thinkster

- [ ] **Creating Angular Filters (http://docs.angularjs.org/guide/dev_guide.templates.filters.creating_filters)**
- [ ] **Using Angular Filters (http://docs.angularjs.org/guide/dev_guide.templates.filters.using_filters)**

Part 4. Directives

Now you're really getting into the meat of what makes Angular special. Directives are certainly one of the most important facets of the framework, and as such, this is one of the biggest sections of the course. The egghead videos do a superb job covering many of the features and use cases of directives:

- [ ] **AngularJS - First Directive (/pick/51d27818f76892ed71000005/angularjs-first-directive)**
- [ ] **AngularJS - Directive Restrictions (/pick/51d27920f7689270f1000007/angularjs-directive-restrictions)**
- [ ] **AngularJS - Basic Behaviors (/pick/51d279f0f76892b250000009/angularjs-basic-behaviors)**
- [ ] **AngularJS - Useful Behaviors (/pick/51d27b41f76892a32100000b/angularjs-useful-behaviours)**
- [ ] **AngularJS - Directives Talking to Controllers (/pick/51d27dd5f76892198800000e/angularjs-directives-talking-to-controllers)**
- [ ] **AngularJS - Directive to Directive Communication (/pick/51d27f261e4b9c3c98000001/angularjs-directives-to-directive-communication)**
- [ ] **AngularJS - Transclusion Basics (/pick/51d284221e4b9ccf8500000b/angularjs-transclusion-basics)**
- [ ] **AngularJS - Components and Containers (/pick/51d28d421e4b9c8d79000019/angularjs-components-and-containers)**
- [ ] **AngularJS - Directive Communication (/pick/51d290a61e4b9cbf0200001d/angularjs-directive-communication)**

thinkster

Readings

There's a lot to digest in those videos, the text will help to clarify:

- [ ] **Chapter 6: Directives - Read from the beginning of the chapter up to and including: In the section "API Overview": Subsection, "Compile and Link Functions**
- [ ] **Also, in section "API Overview": Read subsection "Controllers"**

AngularJS Documentation

To wrap up, the Angular guides offer a bit more detail on the specifics of some aspects of directives:

- [ ] **HTML Compiler (http://docs.angularjs.org/guide/compiler)**
- [ ] **Directives (http://docs.angularjs.org/guide/directive)**

Part 5: Scope

Interestingly, scopes are dramatically underrepresented in mainstream Angular resources compared to how important they are. A solid understanding of scope mechanics is essential when scaling applications, as well as writing modular and testable code. Fortunately, the egghead videos do it justice:

- [ ] **AngularJS - Understanding Isolate Scope (/pick/51d280251e4b9cac38000003/angularjs-understanding-isolate-scope)**
- [ ] **AngularJS - Isolate Scope "@" (/pick/51d281531e4b9c222f000005/angularjs-isolate-scope)**
- [ ] **AngularJS - Isolate Scope "=" (/pick/51d2821d1e4b9c34d0000006/angularjs-isolate-scope)**

thinkster

- [ ] **AngularJS - Isolate Scope "&" (/pick /51d282f21e4b9c9200000008/angularjs-isolate-scope)**
- [ ] **AngularJS - Isolate Scope Review (/pick /51d283891e4b9c01eb00000a/angularjs-isolate-scope-review)**

Readings

- [ ] **Chapter 6: Directives - In the section "API Overview": Read subsection "Scopes"**

AngularJS Documentation

Finally, check out the Angular docs for an in-depth analysis:

- [ ] **Scopes (http://docs.angularjs.org/guide/scope)**

Part 6: Application Structure and Organization

At this point, it's important you step back to rethink and examine Angular as a whole. It's easy to mentally paint yourself into a corner when learning a new language or framework, and a great way to combat this is by exposing yourself to alternate ways of thinking, and viewing the bigger picture.

Watch these egghead videos to get started on thinking about alternative application structure:

- [ ] **An Alternative Approach to Controllers (/pick /51d285a81e4b9c6c9c00000f/angularjs-an-alternative-approach-to-controllers)**

thinkster

- [ ] **Thinking Differently About Organization (/pick /51d286851e4b9cd1f9000011/angularjs-thinking-differently-about-organization)**
- [ ] **Experimental "Controller as" Syntax (/pick /51d28fe81e4b9cdbff00001b/angularjs-experimental-controller-as-syntax)**

Readings

- [ ] **In Chapter 2. "Anatomy of an AngularJS Application": Read the section "Organizing Dependencies with Modules"**

AngularJS Documentation

- [ ] **Read the AngularJS page on Modules (http://docs.angularjs.org/guide/module)**

Part 7: The View and the DOM

This section is a bit of a hybrid of seemingly unrelated topics, DOM manipulation, $watch, and view services, but they are closely related, as they all live in close proximity around the application's views in implementation.

The egghead videos do a superb job of tying these topics together:

- [ ] **AngularJS - angular.element (/pick /51d289de1e4b9c7551000016/angularjs-angular-element)**
- [ ] **AngularJS - $index, $event, $log (/pick /51d28ea81e4b9ceb9200001a/angularjs-index-event-log)**

thinkster

Readings

These book sections will give additional depth on these subjects:

- [ ] **In Chapter 2. "Anatomy of an AngularJS Application": in section "Templates and Data Binding": Read "Observing Model Changes with $watch"**
- [ ] **In Chapter 2. "Anatomy of an AngularJS Application": in section "Templates and Data Binding": Read "Performance Considerations in $watch"**
- [ ] **In Chapter 2. "Anatomy of an AngularJS Application": Read section "Changing the DOM with Directives"**
- [ ] **In Chapter 6. "Directives": In the section "API Overview": Read subsection "Manipulating DOM Elements"**

Additionally, we've put together a section that goes into further detail on DOM manipulation:

- [ ] **Read jqLite, angular.element, and the DOM (/pick/51d3c7e96a4f20b0ed000001/jqlite-angular-element-and-the-dom)**

AngularJS Documentation

- [ ] **Working with CSS in Angular (http://docs.angularjs.org/guide/dev_guide.templates.css-styling)**

Part 8: Templates

Despite this being a short section, understanding Angular templates is critical to being able to build applications effectively.

This handful of egghead videos is an excellent primer on the subject:

- [ ] **templateUrl (/pick/51d2788df768926895000006/angularjs-templateurl)**

thinkster

- [ ] **$templateCache (/pick/51d279a8f7689260dd000008/angularjs-templatecache)**

Readings

The AngularJS book has a terrific and comprehensive section detailing templates:

- [ ] **In Chapter 6. "Directives": in the section "API Overview": Read subsection "The Directive Definition Object"**

AngularJS Documentation

The documentation guide also has a quick but quality piece on Angular templates:

- [ ] **Understanding Angular Templates (http://docs.angularjs.org/guide/dev_guide.templates)**

Part 9: Routing

Angular routing, while not unduly complicated, does introduce a large number of concepts all at once. It also will handle the lion's share (or close to it) of logic for many single page applications. It should then be no surprise that this is the largest section of the course.

The egghead videos appropriately devote a great deal of podium time to routing:

- [ ] **AngularJS - ng-view (/pick/51d27ac2f768926b0300000a/angularjs-ng-view)**
- [ ] **AngularJS - The config function (/pick/51d27b91f76892141100000c/angularjs-the-config-function)**

thinkster

- [ ] **AngularJS - $routeProvider api (/pick /51d27d10f76892aa3300000d/angularjs-routeprovider-api)**
- [ ] **AngularJS - $routeParams (/pick /51d27f741e4b9c0912000002/angularjs-routeparams)**
- [ ] **AngularJS - redirectTo (/pick /51d2808e1e4b9c7216000004/angularjs-redirectto)**
- [ ] **AngularJS - Promises (/pick /51d282301e4b9c48e2000007/angularjs-promises)**
- [ ] **AngularJS - Resolve (/pick/51d2836a1e4b9ce88a000009 /angularjs-resolve)**
- [ ] **AngularJS - Resolve conventions (/pick /51d284991e4b9c068300000c/angularjs-resolve-conventions)**
- [ ] **AngularJS - Resolve $routeChangeError (/pick /51d285b71e4b9c3d69000010/angularjs-resolve-routechangeerror)**
- [ ] **AngularJS - Directive for Route Handling (/pick /51d2871d1e4b9c69ff000012/angularjs-directive-for-route-handling)**
- [ ] **AngularJS - Route Life Cycle (/pick /51d288aa1e4b9c7ed2000015/angularjs-route-life-cycle)**

Readings

Some of the concepts introduced here are probably still fuzzy, and the text does an excellent job of clearing them up:

- [ ] **In Chapter 2. "Anatomy of an AngularJS Application": Read the section "Changing Views with Routes and $location"**
- [ ] **In Chapter 5. "Communicating with Servers": Read the section "The $q and the Promise"**
- [ ] **In Chapter 5. "Communicating with Servers": Read the section "Response Interception"**
- [ ] **In Chapter 7. "Other Concerns": Read the section "$location"**

thinkster

AngularJS Documentation

Finally, the Angular docs have a great section on the $location service:

- [ ] **Using $location (http://docs.angularjs.org/guide/dev_guide.services.$location)**

Part 10: Examples and Analysis

At this point, most of the core Angular topics have been covered, and it's appropriate to get into some examples.

egghead has a good, though fairly trivial, example, "Zippy":

- [ ] **Building Zippy (/pick/51d287891e4b9ce3a9000014/angularjs-building-zippy)**

Readings

Next, the O'Reilly text has a great, chapter-long example that is terrific to work through:

- [ ] **Read all of Chapter 4. "Analyzing an AngularJS App"**

AngularJS Documentation

Last, we recommend working through the AngularJS documentation tutorial. The format they present it in is a bit too hands-off to be exceedingly helpful, so we recommend playing around with it and modifying it to get the most out of the exercise.

- [ ] **AngularJS Tutorial (http://docs.angularjs.org/tutorial)**

thinkster

Part 11: Under the Hood

At this juncture, it's appropriate to dive into the niceties of AngularJS. In order to truly master the framework, hand-wavy arguments for how things work aren't sufficient anymore. You need to get into the nitty-gritty of what makes Angular tick.

These egghead videos offer an excellent primer for some increasingly advanced topics:

- [ ] **$scope vs. scope (/pick/51d271fff7689208c9000001/angularjs-scope-vs-scope)**
- [ ] **Providers (/pick/51d28a811e4b9cd3a2000017/angularjs-providers)**
- [ ] **Injectors (/pick/51d28ba41e4b9c412e000018/angularjs-injectors)**
- [ ] **ngmin (/pick/51e756d857740ed769000001/angularjs-ngmin)**

Readings

Following this, the book has a handful of sections detailing these and other new framework topics:

- [ ] **In Chapter 7. "Other Concerns": in the section "AngularJS Module Methods": Read the section "Communicating Between Scopes with $on, $emit, and $broadcast"**

Additionally,

- [ ] **Read this in-depth analysis of $digest (/pick/51dc70a1fc30e44f96000001/digest)**

AngularJS Documentation

thinkster

Finally, there are a healthy number of Angular documentation guide pages that really get down into dissecting Angular. These are some of the best resources on Angular's innards out there, make sure and fully take them in:

- [ ] **Conceptual Overview (http://docs.angularjs.org/guide/concepts)**
- [ ] **Dependency Injection (http://docs.angularjs.org/guide/di)**
- [ ] **Angular Services (http://docs.angularjs.org/guide/dev_guide.services)**
- [ ] **Creating Services (http://docs.angularjs.org/guide/dev_guide.services.creating_services)**
- [ ] **Injecting Services into Controllers (http://docs.angularjs.org/guide/dev_guide.services.injecting_controllers)**
- [ ] **Managing Service Dependencies (http://docs.angularjs.org/guide/dev_guide.services.managing_dependencies)**
- [ ] **Understanding Angular Services (http://docs.angularjs.org/guide/dev_guide.services.understanding_services)**

Part 12: Development Environment and Testing

Much of Angular's design is built around being highly testable. Central to this is the widespread utilization of dependency injection, which you read about in Chapter 11. Not only this, but with tools like Yeoman available, a robust test suite becomes realistic and manageable.

There is only one egghead video on testing, and it gives a simplistic overview of a unit test on a filter. (It's worth mentioning that the Testacular test runner is now called 'Karma'):

thinkster

- [ ] **AngularJS - Testing Overview (/pick/51d27283f768928d0b000002/angularjs-testing-overview)**

Readings

We'd like to get a bit more in depth than that, so next read all of Ch.3 in O'Reilly. This chapter focuses on Yeoman (http://yeoman.io), which is a set of productivity tools that make Angular a lot more digestible: Yo, a scaffolding tool, Grunt, the build and testing tool, and Bower, the package management tool.

- [ ] **Read Chapter 3: Developing in AngularJS**

Meet the Gang

A person new to testing might be a little overwhelmed with these concepts and how they play together in the world of testing. This reference should help out:

thinkster

- **Yeoman (http://yeoman.io/)** has the scaffolding tool `yo` that generates an application skeleton to start out with, complete with things like Bootstrap or a pre-configured testing setup. The scaffold of the application is different in many ways to the angular-seed scaffold, but it is important to note that they both use Karma and Jasmine in the same ways.
- **angular-seed (https://github.com/angular/angular-seed)** is a ready-to-eat AngularJS scaffold available on their github with directory structure and testing amenities pre-prepared. Testing this application is accomplished by running a standalone Karma test server.
- **Grunt (http://gruntjs.com/)** is the testing tool used in Yeoman, but it is used as a wrapper for Karma.
- **Karma (http://karma-runner.github.io/)** is the actual test runner that is used to test AngularJS. It can be used standalone from Grunt. Karma circumvents testing inconsistencies across browsers, which would happen with things like PhantomJS, by actually launching a browser and running the tests in it.
- **Jasmine (http://pivotal.github.io/jasmine/)** is the testing framework which is used for unit tests by default in Karma. Angular E2E tests with Karma don't and can't use the Jasmine adapter, although E2E tests use very similar syntax with the Angular Scenario Runner. This blog post (http://sravi-kiran.blogspot.com/2013/04/UnitTestingAngularJsControllerUsingJasmine.html) does a fine job of going through how to actually author some Jasmine tests, and gives some excellent examples.
- **Angular Scenario Runner (http://docs.angularjs.org/guide/dev_guide.e2e-testing)** is used as the E2E testing framework for AngularJS, and is syntactically similar to Jasmine.

AngularJS Documentation

Now that we have fleshed out how testing should generally go, let's take a look at the Angular docs on testing. These are going to give some more information on how to think about Angular testing. While they are a good resource to have, regrettably, some of them are not yet complete.

- [ ] **Unit Testing (http://docs.angularjs.org/guide/dev_guide.unit-testing)**

thinkster

- [ ] **Testing Angular Services (http://docs.angularjs.org/guide/dev_guide.services.testing_services)**
- [ ] **E2E Testing (http://docs.angularjs.org/guide/dev_guide.e2e-testing)**

More Readings and Examples

- [ ] Now, read through How to Test an AngularJS Directive (http://newtriks.com/2013/04/26/how-to-test-an-angularjs-directive/). This blog post goes through setting up Yeoman, generating a very simple sample application, writing tests for a directive, and running tests using Karma.

- [ ] Finally, go through one of the better Angular testing resources to date, Full-Spectrum Testing with AngularJS and Karma (http://www.yearofmoo.com/2013/01/full-spectrum-testing-with-angularjs-and-karma.html). This is an outstanding resource that has fantastic explanations and demonstrations of testing with Grunt and Karma. Included are examples for each of the major testing categories in Angular. It also adds the intermediate Midway test paradigm to the fray, which is a compelling convenience if you're trying to streamline authoring tests.

Part 13. $http and Server Interaction

$http

In the face of a slew of new concepts, it's important to not overthink the $http service.

$http can be thought of as a wrapper to make requests to external HTTP entities with the browser's XHR object or JSONP. Because of the nature of these requests, it employs callbacks afforded to it by the $q defer/promise API.

thinkster

More generally, $http is used predominantly for AJAX requests. Its API exists as you would expect, with the ability to make GET, POST, PUT, DELETE, HEAD, and JSONP requests. You are responsible for making these calls manually, and for constructing how to handle the objects they return, and the callbacks for various outcomes that might occur with an asynchronous request.

$resource

AngularJS offers another level abstractions above $http, the $resource service. This is a convenience offered to you when dealing with external resources that are RESTful. It wraps the $http service, which is pointed at a singular endpoint, and presents a REST API on the $resource to handle RESTful requests. The $resource get() method returns a Resource object, which can then be modified with CRUD (or custom defined) operations invoked upon it.

Authentication

Since services are singletons, and can be injected pretty much anywhere, they are perfect for use in authentication logic. There is a diverse spectrum of implementations on how exactly to construct the service, but they are all variations on basically the same theme.

services.js

```
1. myApp.factory('AuthenticationService', function() {
2.   var current_user;
3.   return {
4.     signIn: function() {
5.       // check password on server, get user data, unique token
6.     },
7.     signOut: function() {
8.       // clear current_user data, unset logged in status, etc.
9.     },
10.    isSignedIn: function() {
11.      // logic to check if current user has signed in
12.    },
13.    currentUser: function() {
14.      // return the current_user object, or handle if the user
15.    }
16.  };
17. });
```

thinkster

Since it is a service, you can inject this to your heart's desire, set watchers on its methods, use the user data in views, handle routing conditionals based on signin status, etc. Since all this data is still just freely living in the AngularJS framework, you still need to exercise the same security precautions when performing server transactions involving authentication.

- [ ] Read this blog entry, Deal with users authentication in an AngularJS web app (http://blog.brunoscopelliti.com/deal-with-users-authentication-in-an-angularjs-web-app). It offers excellent insight into service-based Angular authentication.

Readings

The text offers a wealth of information on interacting with servers:

- [ ] **In Chapter 2. "Anatomy of an AngularJS Application": in the section "Changing Views with Routes and $location": Read the subsection "Talking to Servers"**
- [ ] **Read all of Chapter 5. "Communicating with Servers"**
- [ ] **In Chapter 8. "Cheatsheet and Recipes": Read the section "Working with Servers and Login"**

AngularJS Documentation

Finally, check out the Angular API docs on $http and $resource:

- [ ] **The $http service (http://docs.angularjs.org/api/ng.$http)**
- [ ] **The $resource service (http://docs.angularjs.org/api/ngResource.$resource)**

thinkster