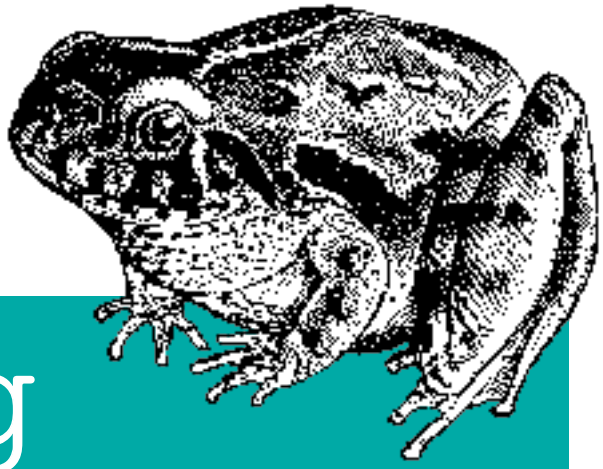
O'REILLY®



Learning AngularJS

A GUIDE TO ANGULARJS DEVELOPMENT

Ken Williamson

O'REILLY®

Learning AngularJS

With AngularJS, you can quickly build client-side applications that run well on any desktop or mobile platform, using REST web services for backend processes. You may have heard that the learning curve for this JavaScript MVC framework is too steep, but that's not the case. This practical guide provides a hands-on approach to learning AngularJS that will have you building high-quality applications and websites in no time.

Along with a conceptual understanding of the framework, you'll also gain direct experience with AngularJS by building a sample application throughout the book. If you're familiar with JavaScript, web development, and software design concepts and patterns, this book is the perfect way to get started.

- Understand how AngularJS differs from other MVC frameworks
- Learn about AngularJS controllers, views, and models by diving into the book's sample project
- Connect your working application to public REST services
- Build the application's security layer with non-REST AngularJS services
- Explore the basics of building and testing AngularJS directives
- Use AngularJS as part of the MEAN stack (MongoDB, ExpressJS, AngularJS, and Node.js)
- Discover how search engine optimization as it relates to AngularJS applications and sites

Ken Williamson, a software engineer and architect with over 20 years of experience, has designed and written mobile, desktop, and server software for some of the largest companies in the world. He's the founder of several open source projects, including Ulbora CMS.

"This book is a must-have for any aspiring Angular developer. Ken explains the framework in a clear, concise manner all the way from *hello, world* to end-to-end testing. Essential for any team."

—Sam Reaves
Web Developer at Nomi

JAVASCRIPT

US $34.99 **CAN $40.99**
ISBN: 978-1-491-91675-9

Twitter: @oreillymedia
facebook.com/oreilly

Learning AngularJS

Ken Williamson

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

Learning AngularJS
by Ken Williamson

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Meg Foley
Production Editor: Nicole Shelby
Copyeditor: Rachel Head
Proofreader: Rachel Monaghan
Indexer: WordCo. Indexing Services
Interior Designer: David Futato
Cover Designer: Ellie Volckhausen
Illustrator: Rebecca Demarest

March 2015: First Edition

Revision History for the First Edition
2015-03-10: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491916759* for release details.

978-1-491-91675-9

[LSI]

I would like to thank my son Chris for all his help and for being a sounding board.
Thanks, Chris.

Table of Contents

Preface

The world of software development has changed drastically over the last few decades. Many software methodologies and concepts that were considered "cutting edge" 20 or so years ago are now common practice in the field of software development, and have been for years. One example is the World Wide Web and the use of web browsers to deliver software to users. In 1993, the concept of delivering software over the Internet that could then run in a web browser on any machine running on any operating system was considered bleeding edge. But as any computer user knows, that practice has been commonplace for years now.

When JavaScript client-side web application frameworks like AngularJS, Backbone.js, and Ember.js first appeared, they were considered too cutting edge for most serious software projects. As they matured, however, software architects and developers saw great potential in these frameworks. Applications built with JavaScript client-side frameworks exist and run entirely on the user's hardware, much like conventional thick-client applications. Applications written using these frameworks are much faster than conventional web applications and provide a much better user experience.

Over the last couple of years, JavaScript client-side frameworks have made great strides in functionality and reliability, and they are now heavily used to build mobile HTML5 applications. But mobile applications are only the starting point. These frameworks now have the potential to radically change the way we build modern web application software. Of all the JavaScript frameworks available, AngularJS, backed by Google, is the one that shines the brightest.

AngularJS has many advantages over other JavaScript client-side frameworks. AngularJS uses the MVC design pattern and embraces that pattern completely. The model, view, and controller are all clearly defined in AngularJS and serve to greatly simplify the development process. With AngularJS, developers can build applications that have a clear separation between their functional layers.

One of the greatest advantages of AngularJS over other JavaScript client-side frameworks is the unique way in which it lets developers interact with RESTful web

services. AngularJS's `resource` object lets developers interact with REST services like standard objects. The complexity of REST services can be greatly simplified using this approach: with only a few lines of code, you can create an AngularJS service that interacts with multiple backend REST services. Those services can then be used throughout your application, reducing the total number of lines of code.

In fact, one of the biggest advantages of AngularJS over other client-side frameworks is its concept of services. AngularJS services help to greatly simplify an application by compartmentalizing client-side logic into single units of code. Those single units, called services, can then be used repeatedly throughout an application. AngularJS services prove especially powerful when you're building large enterprise applications with many lines of code and much complexity. Complex logic can be written only once inside an AngularJS service and then used wherever needed. That alone makes AngularJS the best choice for your next JavaScript project.

Thanks to this use of services and its all-inclusive design, AngularJS helps developers write less code, thereby greatly reducing application complexity. The simplicity of AngularJS makes it easy to learn and easy to use. Any time spent learning AngularJS is time well spent. Any time spent developing AngularJS applications is time spent turning a cutting-edge technology into a commonplace technology. In this book I strive to help you do both, encouraging design concepts and practices that will help you build better AngularJS applications.

Why I Wrote This Book

I constantly see development teams avoid using AngularJS because of its perceived steep learning curve. Those same teams often choose other JavaScript frameworks because they initially seem easier to learn. But AngularJS is not hard to learn at all. It is actually much easier to learn than other JavaScript frameworks, if the learning process is approached correctly. Like many others, I struggled to learn AngularJS in the beginning. This book was written to help developers avoid the early struggles associated with learning AngularJS and get started building AngularJS applications and websites very quickly.

What This Book Covers

This book covers everything you need to know to build fully functional AngularJS applications. The book starts off with the basics of AngularJS. You will learn about AngularJS components in early chapters. As chapters progress, you will get hands-on experience building working AngularJS projects.

Near the end of the book, you will write the AngularJS part of a working MEAN stack blog application and deploy the application to the cloud. MEAN stands for Mon-

goDB, ExpressJS, AngularJS, and Node.js. Many industry experts believe the MEAN stack will be a dominant web development platform in coming years.

After reading this book, you will have the knowledge to start building high-quality AngularJS applications and websites. You will also gain a clear understanding of the design concepts associated with AngularJS applications, and of security as it relates to AngularJS applications.

Who Should Read This Book

This book is intended for anyone who has an interest in learning to develop AngularJS applications or websites quickly. It will also be helpful to anyone interested in learning how AngularJS is used in a MEAN stack application. The reader will gain not only a conceptual understanding of AngularJS, but hands-on experience as well. Anyone reading this book should have some knowledge of JavaScript, software design concepts, and software design patterns. A prior knowledge of web development will also be helpful.

The Chapters in This Book

This book starts off with the very basics of AngularJS and assumes the reader has no prior knowledge of the framework. The chapters are arranged as follows:

- *Chapter 1, Introduction to AngularJS*, starts off with a basic introduction to AngularJS. It helps the reader understand how AngularJS differs from other frameworks.
- *Chapter 2, The IDE and AngularJS Projects*, covers setting up a development environment and building AngularJS projects. It also covers how to set up a test environment for AngularJS.
- *Chapter 3, MVC and AngularJS*, compares AngularJS to traditional web MVC frameworks. It shows why AngularJS is a better framework for building modern web applications and websites.
- *Chapter 4, AngularJS Controllers*, is a discussion of AngularJS controllers. The reader will build part of a working application and implement controller testing near the end of the chapter.
- *Chapter 5, AngularJS Views and Bootstrap*, covers AngularJS views built with Twitter Bootstrap. The reader will continue working on a functional application and implement testing.
- *Chapter 6, AngularJS and REST Services*, covers AngularJS services as they relate to REST services. The reader will continue working on the application and connect it to public REST services written for this book.

- *Chapter 7, AngularJS Models*, covers AngularJS models and how they relate to controllers and views. The reader will continue working on the application started earlier.
- *Chapter 8, Services and Business Logic*, covers non-REST AngularJS services. In this chapter the reader will build part of the security layer used later in the book.
- *Chapter 9, AngularJS Directives*, covers the basics of building and testing AngularJS directives.
- *Chapter 10, AngularJS Security*, shows the reader how to use the security layer introduced in Chapter 8 to secure the AngularJS application started earlier.
- *Chapter 11, MEAN Cloud and Mobile*, shows the reader how to use the AngularJS application developed in previous chapters in a MEAN stack application and in a mobile application.
- *Chapter 12, AngularJS and SEO*, covers search engine optimization as it relates to AngularJS applications and websites.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a general note.

This element signifies a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/KenWilliamson*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Learning AngularJS* by Ken Williamson (O'Reilly). Copyright 2015 Ken Williamson, 978-1-491-91675-9."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/learning-angularjs*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

CHAPTER 1

Introduction to AngularJS

Google's AngularJS is an all-inclusive JavaScript model-view-controller (MVC) framework that makes it very easy to quickly build applications that run well on any desktop or mobile platform. In a very short period of time, AngularJS has moved from being an unknown open source offering to one of the best known and most widely used JavaScript client-side frameworks offered. AngularJS 1.3 and greater combined with jQuery and Twitter Bootstrap give you everything you need to rapidly build HTML5 JavaScript application frontends that use REST web services for the backend processes. This book will show you how to use all three frontend components to harness the power of REST services on the backend and quickly build powerful mobile and desktop applications.

JavaScript Client-Side Frameworks

JavaScript client-side applications run on the user's device or PC, and therefore shift the workload to the user's hardware and away from the server. Until fairly recently, server-side web MVC frameworks like Struts, Spring MVC, and ASP.NET were the frameworks of choice for most web-based software development projects. JavaScript client-side frameworks, however, are sustainable models that offer many advantages over conventional web frameworks, such as simplicity, rapid development, speed of operation, testability, and the ability to package the entire application and deploy it to all mobile devices and the Web with relative ease. You can build your application one time and deploy and run it anywhere, on any platform, with no modifications. That's powerful.

AngularJS makes that process even faster and easier. It helps you build frontend applications in days rather than months and has complete support for unit testing to help reduce quality assurance (QA) time. AngularJS has a rich set of user documentation and great community support to help answer questions during your develop-

ment process. Models and views in AngularJS are much simpler than what you find in most JavaScript client-side frameworks. Controllers, often missing in other JavaScript client-side frameworks, are key functional components in AngularJS.

Figure 1-1 shows a diagram of an AngularJS application and all related MVC components. Once the AngularJS application is launched, the model, view, controller, and all HTML documents are loaded on the user's mobile or desktop device and run entirely on the user's hardware. As you can see, calls are made to the backend REST services, where all business logic and business processes are located. The backend REST services can be located on a private web server or in the cloud (which is most often the case). Cloud REST services can scale from a handful of users to millions of users with relative ease.

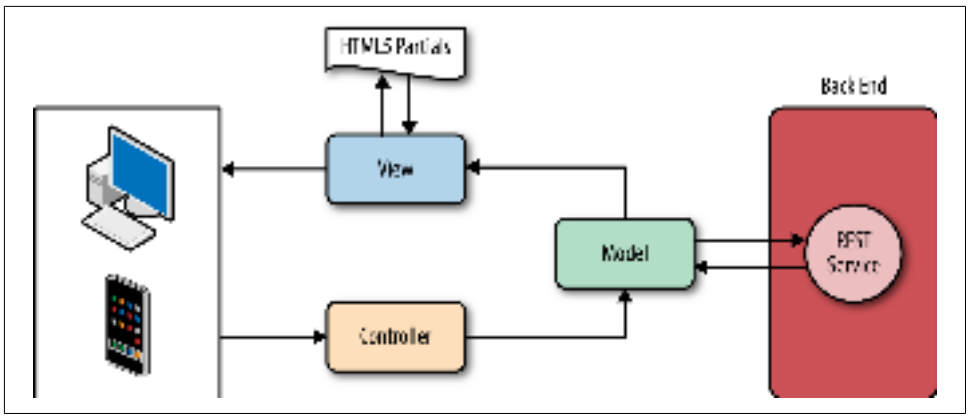


Figure 1-1. Diagram of an AngularJS MVC application

Single-Page Applications

AngularJS is most often used to build applications that conform to the single-page application (SPA) concept. SPAs are applications that have one entry point HTML page; all the application content is dynamically added to and removed from that one page. You can see the entry point of our SPA in the *index.html* code that follows. The tag `<div ng-view></div>` is where all dynamic content is inserted into *index.html*:

```
<!-- chapter1/index.html -->
<!DOCTYPE html>

<html lang="en" ng-app="helloWorldApp">

<head>

<title>AngularJS Hello World</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<script src="js/libs/angular.min.js"></script>
<script src="js/libs/angular-route.min.js"></script>
<script src="js/libs/angular-resource.min.js"></script>
<script src="js/libs/angular-cookies.min.js"></script>


<script src="js/app.js"></script>
<script src="js/controllers.js"></script>
<script src="js/services.js"></script>


</head>

<body>
<div ng-view></div>
</body>

</html>
```

As the user clicks on links in the application, existing content attached to the tag is removed and new dynamic content is then attached to the same tag. Rather than the user waiting for a new page to load, new content is dynamically displayed in a fraction of the time that it would take to load a new HTML web page.

You can download the source for the Chapter 1 "Hello, World" application from GitHub.

Bootstrapping the Application

Bootstrapping AngularJS is the process of loading AngularJS when an application first starts. Loading the AngularJS libraries in a page will start the bootstrap process. The *index.html* file is analyzed, and the parser looks for the `ng-app` tag. The line `<html lang="en" ng-app="helloWorldApp"></html>` shows how `ng-app` is defined. The following code shows the JavaScript that is fired by that line in the *index.html* file. As you can see, *app.js* is where the AngularJS application *helloWorldApp* is defined as an AngularJS module, and this is the entry point into the application. The variable `helloWorldApp` in this file could be named anything. I will, however, call it `helloWorldApp` for the sake of uniformity:

```
/* chapter1/app.js excerpt */
'use strict';
/* App Module */

var helloWorldApp = angular.module('helloWorldApp', [
```

```
  'ngRoute',
  'helloWorldControllers'
]);
```

Dependency Injection

Dependency injection (DI) is a design pattern where dependencies are defined in an application as part of the configuration. Dependency injection helps you avoid having to manually create application dependencies. AngularJS uses dependency injection to load module dependencies when an application first starts. The *app.js* code in the previous section shows how AngularJS dependencies are defined.

As you can see, two dependencies are defined as needed by the *helloWorldApp* application at startup. The dependencies are defined in an array in the module definition. The first dependency is the AngularJS `ngRoute` module, which provides routing to the application. The second dependency is our controller module, `helloWorldControllers`. We will cover controllers in depth later, but for now just understand that controllers are needed by our applications at startup time.

Dependency injection is not a new concept. It was introduced over 10 years ago and has been used consistently in various application frameworks; DI was at the core of the popular Spring framework written in Java. One of its main advantages is that it reduces the need for boilerplate code, writing of which would normally be a time-consuming process for a development team.

Dependency injection also helps to make an application more testable. That is one of the main advantages of using AngularJS to build JavaScript applications. AngularJS applications are much easier to test than applications written with most JavaScript frameworks. In fact, there is a test framework that has been specifically written to make testing AngularJS applications easy. We will talk more about testing at the end of this chapter.

AngularJS Routes

AngularJS routes are defined through the `$routeProvider` API. Routes are dependent on the `ngRoute` module, and that's why it is a requirement when the application starts. The following code from *app.js* shows how we define routes in an AngularJS application. Two routes are defined—the first is / and the second is */show*:

```
/* chapter1/app.js excerpt */
helloWorldApp.config(['$routeProvider', '$locationProvider',
function($routeProvider, $locationProvider){
 $routeProvider.
   when('/', {
     templateUrl: 'partials/main.html',
     controller: 'MainCtrl' }).
```

```
    when('/show', {
      templateUrl: 'partials/show.html',
      controller: 'ShowCtrl'
});
```

The two defined routes map directly to URLs defined in the application. If a user clicks on a link in the application specified as *www.someDomainName/show*, the */show* route will be followed and the content associated with that URL will be displayed. If the user clicks on a link specified as *www.someDomainName/*, the / route will be followed and that content will be displayed.

HTML5 Mode

The complete *app.js* file is shown next. The last line in *app.js* (`$locationProvider.html5Mode(false).hashPrefix('!');`) uses the `locationProvider` service. This line of code turns off the HTML5 mode and turns on the hashbang mode of AngularJS. If you were to turn on HTML5 mode instead by passing `true`, the application would use the HTML5 History API. HTML5 mode also gives the application pretty URLs like */someAppName/blogPost/5* instead of the standard AngularJS URLs like */someAppName/#!/blogPost/5* that use the *#!*, known as the hashbang.

```
/* chapter1/app.js complete file */

'use strict';
/* App Module */

var helloWorldApp = angular.module('helloWorldApp', [
  'ngRoute',
  'helloWorldControllers'
]);

helloWorldApp.config(['$routeProvider', '$locationProvider',
  function($routeProvider, $locationProvider) {
    $routeProvider.
      when('/', {
         templateUrl: 'partials/main.html',
         controller: 'MainCtrl'
      }).when('/show', {
         templateUrl: 'partials/show.html',
         controller: 'ShowCtrl'
});
$locationProvider.html5Mode(false).hashPrefix('!');
}]);
```

HTML5 mode can provide pretty URLs, but it does require configuration changes on the web server in most cases. The changes are different for each individual web server, and can differ for different server installations as well. HTML5 mode also handles URL changes in a different way, by using the HTML History API for navigation.

Using HTML5 mode is just a configuration change in AngularJS, and we won't cover the needed server changes in this book as our focus is on AngularJS. The AngularJS site has documentation on the changes needed for all modern web servers when HTML5 mode is enabled. Using this mode has some benefits, but we will stick with hashbang mode in our chapter exercises.

Hashbang mode is used to support conventional search engines that don't have the ability to execute JavaScript on Ajax sites like those built with AngularJS. When a conventional search engine searches a site built with AngularJS that uses hashbangs, the search engine replaces the *#!* with *?_escaped_fragment_=*. Conventional search engines expect the server to have HTML snapshots at the location where *_escaped_fragment_=* is configured to point. HTML snapshots are merely copies of the HTML rendered version of the website or application.

Modern Search Engines

Fortunately, modern search engines have the ability to execute JavaScript, as announced by Google in a news release on May 23, 2014. Hashbang mode also allows AngularJS applications to store Ajax requested pages in the browser's history. That process often simplifies browser bookmarks.

AngularJS Templates

AngularJS partials, also called *templates*, are code sections that contain HTML code that are bound to the `<div ng-view></div></div>` tag shown in the *index.html* file earlier in this chapter. If you look back at the complete *app.js* file, you can see that different `templateUrl` values are defined for each route.

The *main.html* and *show.html* files listed next show the two defined partials (templates). The templates contain just HTML code, with nothing special at this time. Later, we will use AngularJS's built-in template language to display dynamic data in our templates:

```
<!-- chapter1/main.html -->

<div>Hello World</div>

<!-- chapter1/show.html -->

<div>Show The World</div>
```

As the user clicks on the different links, the value assigned to `<div ng-view>` is replaced with the content of the associated template files. The value of *controller* defined for each route references the controller component (of the MVC pattern) that is defined for each particular route.