ANGULARJS

by Google

Compiled by: Tanner Linsley and Brennan Davis

Introduction

What is AngularJS?

AngularJS is a structural framework for dynamic web apps. It lets you use **HTML** as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Out of the box, it eliminates much of the code you currently write through data binding and dependency injection. And it all happens in **JavaScript** within the browser, making it an ideal partner with any server technology.

Angular is what HTML would have been had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in what do I have to do to trick the browser into doing what I want.

The impedance mismatch between dynamic applications and static documents is often solved with:

- **library** - a collection of functions which are useful when writing web apps. Your code is in charge and it calls into the library when it sees fit. E.g., jQuery.
- **frameworks** - a particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific. E.g., knockout, ember, etc.

Angular takes another approach. It attempts to minimize the impedance mismatch between document centric HTML and what an application needs by creating new HTML constructs. Angular teaches the browser new syntax through a construct we call directives. Examples include:

- Data binding, as in {{ }}.
- **DOM** Control structures for repeating/hiding DOM fragments.
- Support for forms and form validation.
- Attaching code-behind to DOM elements.
- Grouping of HTML into reusable components

A complete client-side solution

Angular is not a single piece in the overall puzzle of building the client-side of a web application. It handles all of the DOM and AJAX glue code you once wrote by hand and puts it in a well-defined structure. This makes Angular opinionated about how a CRUD application should be built. But while it is opinionated, it also tries to make sure that its opinion is just a starting point you can easily change. Angular comes with the following out-of-the-box:

- Everything you need to build a **CRUD** app in a cohesive set: data-binding, basic templating directives, form validation, routing, deep-linking, reusable components, dependency injection.
- Testability story: unit-testing, end-to-end testing, mocks, test harnesses.
- Seed application with directory layout and test scripts as a starting point.

Angular Sweet Spot

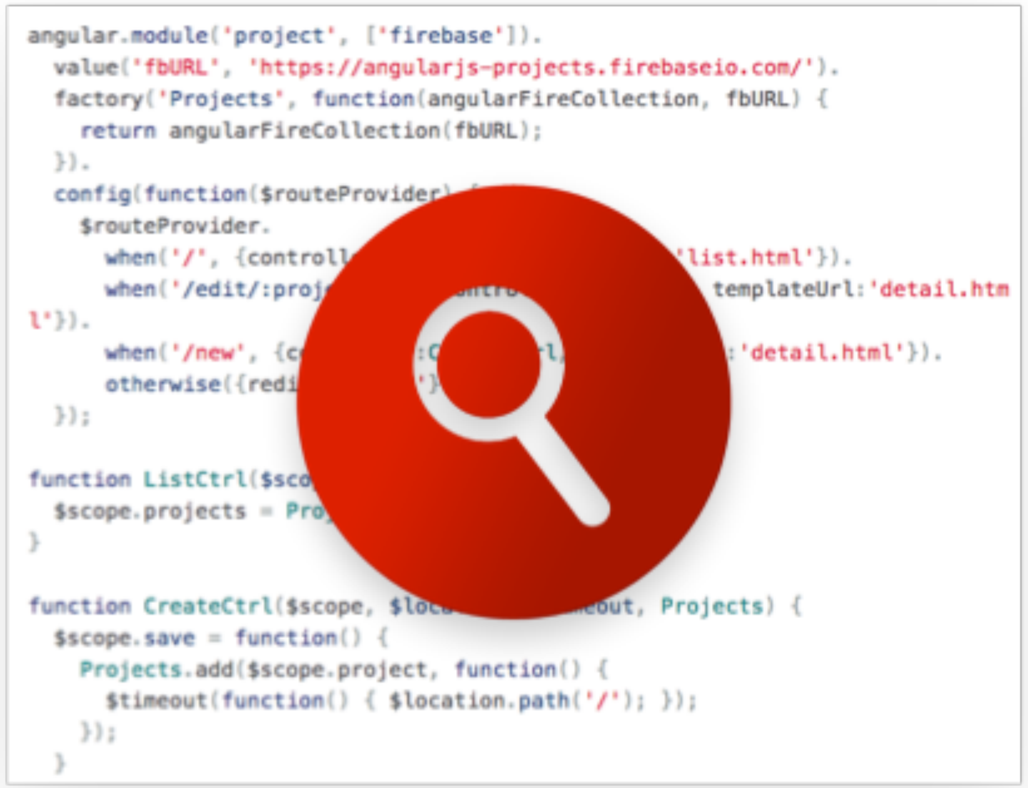
Angular simplifies application development by presenting a higher level of abstraction to the developer. Like any abstraction, it comes at a cost of flexibility. In other words not every app is a good fit for Angular. Angular was built for the CRUD application in mind. Luckily CRUD applications represent the majority of web applications. But to understand what Angular is good at, one also has to understand when an app is not a good fit for Angular.

Games and **GUI** editors are examples of applications with intensive and tricky DOM manipulation. These kinds of apps are different from CRUD apps, and as a result are probably not a good fit for Angular. In these cases it may be better to use a library with a lower level of abstraction, such as jQuery.

The Basics

The Basics

The best way to learn any new web technology is to jump right in. Provided here is a basic web application built using AngularJS.



```
angular.module('project', ['firebase']).
  value('fbURL', 'https://angularjs-projects.firebaseio.com/').
  factory('Projects', function(angularFireCollection, fbURL) {
    return angularFireCollection(fbURL);
  }).
  config(function($routeProvider) {
    $routeProvider.
      when('/', {controll[illegible]'list.html'}).
      when('/edit/:proj[illegible] templateUrl:'detail.htm
l'}).
      when('/new', {c[illegible]:'detail.html'}).
      otherwise({redi[illegible]
  });

function ListCtrl($sco[illegible]
  $scope.projects = Pro[illegible]
}

function CreateCtrl($scope, $loc[illegible]out, Projects) {
  $scope.save = function() {
    Projects.add($scope.project, function() {
      $timeout(function() { $location.path('/'); });
    });
  }
```

Add Some Control

Add Some Control

Data Binding

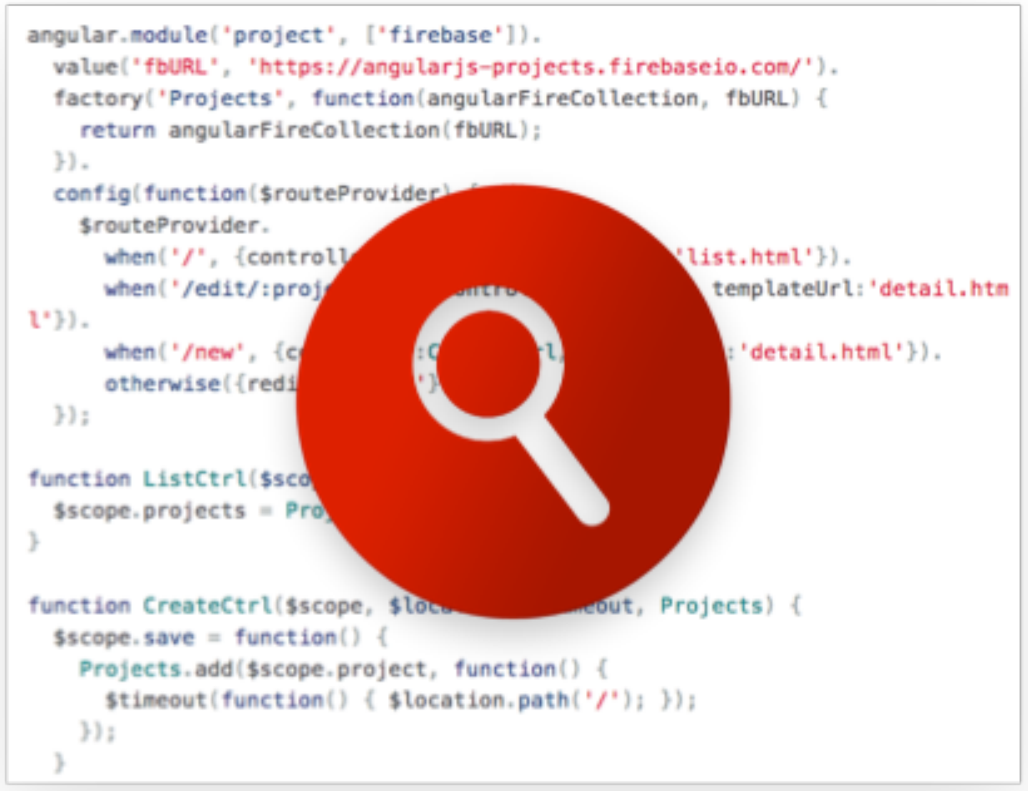
Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. This is awesome because it eliminates DOM manipulation from the list of things you have to worry about.

Controller

Controllers are the behavior behind the DOM elements. AngularJS lets you express the behavior in a clean readable form without the usual boilerplate of updating the DOM, registering callbacks or watching model changes.

Plain Javascript

Unlike other frameworks, there is no need to inherit from proprietary types; to wrap the model in accessors methods. Just plain old JavaScript here. This makes your code easy to test, maintain, reuse, and again free from boilerplate.



```
angular.module('project', ['firebase']).
  value('fbURL', 'https://angularjs-projects.firebaseio.com/').
  factory('Projects', function(angularFireCollection, fbURL) {
    return angularFireCollection(fbURL);
  }).
  config(function($routeProvider) [illegible]
    $routeProvider.
      when('/', {controll[illegible]'list.html'}).
      when('/edit/:proj[illegible] templateUrl:'detail.htm
l'}).
      when('/new', {c[illegible]:C[illegible]l,[illegible]:'detail.html'}).
      otherwise({redi[illegible]'}[illegible]
  });

function ListCtrl($sco[illegible]
  $scope.projects = Pro[illegible]
}

function CreateCtrl($scope, $loc[illegible]out, Projects) {
  $scope.save = function() {
    Projects.add($scope.project, function() {
      $timeout(function() { $location.path('/'); });
    });
  }
```

Wire up a Backend

Wire up a Backend

Deep Linking

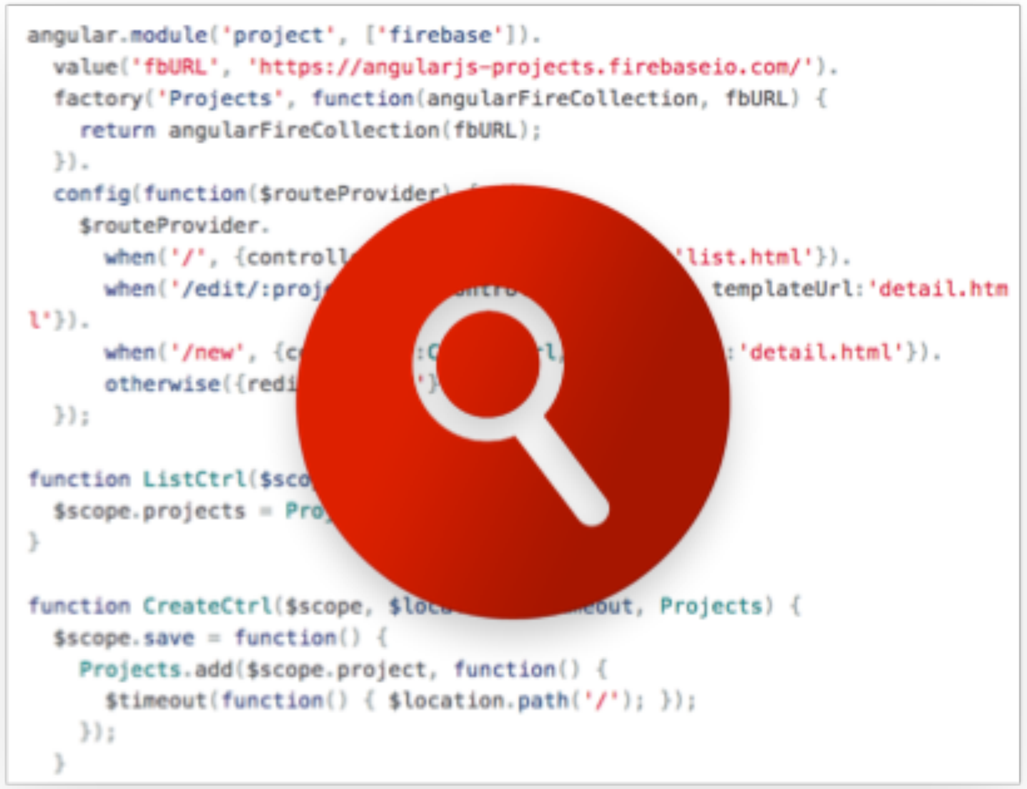
A deep link reflects where the user is in the app, this is useful so users can bookmark and email links to locations within apps. Round trip apps get this automatically, but AJAX apps by their nature do not. AngularJS combines the benefits of deep link with desktop app-like behavior.

Form Validation

Client-side form validation is an important part of great user experience. AngularJS lets you declare the validation rules of the form without having to write JavaScript code. Write less code, go have beer sooner.

Server Communication

AngularJS provides services on top of XHR, that dramatically simplify your code. We wrap XHR to give you exception management and promises. Promises further simplify your code by handling asynchronous return of data. This lets you assign properties synchronously when the return is actually asynchronous.



```
angular.module('project', ['firebase']).
  value('fbURL', 'https://angularjs-projects.firebaseio.com/').
  factory('Projects', function(angularFireCollection, fbURL) {
    return angularFireCollection(fbURL);
  }).
  config(function($routeProvider) [illegible]
    $routeProvider.
      when('/', {controll[illegible]'list.html'}).
      when('/edit/:proj[illegible] templateUrl:'detail.htm
l'}).
      when('/new', {c[illegible]:C[illegible]l,[illegible]:'detail.html'}).
      otherwise({redi[illegible]'}[illegible]
  });

function ListCtrl($sco[illegible]
  $scope.projects = Pro[illegible]
}

function CreateCtrl($scope, $loc[illegible]eout, Projects) {
  $scope.save = function() {
    Projects.add($scope.project, function() {
      $timeout(function() { $location.path('/'); });
    });
  }
```

Create Components

Create Components

Directives

Directives is a unique and powerful feature available only in Angular. Directives let you invent new HTML syntax, specific to your application.

Reusable Components

We use directives to create reusable components. A component allows you to hide complex DOM structure, CSS, and behavior. This lets you focus either on what the application does or how the application looks separately.

Localization

An important part of serious apps is localization. Angular's locale aware filters and stemming directives give you building blocks to make your application available in all locales.

```
angular.module('project', ['firebase']).
  value('fbURL', 'https://angularjs-projects.firebaseio.com/').
  factory('Projects', function(angularFireCollection, fbURL) {
    return angularFireCollection(fbURL);
  }).
  config(function($routeProvider[illegible]
    $routeProvider.
      when('/', {controll[illegible]'list.html'}).
      when('/edit/:proj[illegible] templateUrl:'detail.htm
l'}).
      when('/new', {c[illegible]:'detail.html'}).
      otherwise({redi[illegible]
  });

function ListCtrl($sco[illegible]
  $scope.projects = Pro[illegible]
}

function CreateCtrl($scope, $loc[illegible]out, Projects) {
  $scope.save = function() {
    Projects.add($scope.project, function() {
      $timeout(function() { $location.path('/'); });
    });
  }
```

Embed & Inject

Embed and Inject

Embeddable

AngularJS works great with other technologies. Add as much or as little of AngularJS to an existing page as you like. Many other frameworks require full commitment. This page has multiple AngularJS applications embedded in it. Because AngularJS has no global state multiple apps can run on a single page without the use of iframes. We encourage you to view-source and look around.

Injectable

The dependency injection in AngularJS allows you to declaratively describe how your application is wired. This means that your application needs nomain() method which is usually an unmaintainable mess. Dependency injection is also a core to AngularJS. This means that any component which does not fit your needs can easily be replaced.

Testable

AngularJS was designed from ground up to be testable. It encourages behavior-view separation, comes pre-bundled with mocks, and takes full advantage of dependency injection. It also comes with end-to-end scenario runner which eliminates test flakiness by understanding the inner workings of AngularJS.

Conclusion

So, Why AngularJS?

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

Alternatives

Other frameworks deal with HTML's shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for dynamic views.

Extensibility

AngularJS is a toolset for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs.

CRUD

In computer programming, create, read, update and delete (CRUD) (Sometimes called SCRUD with an "S" for Search) are the four basic functions of persistent storage.Sometimes CRUD is expanded with the words retrieve instead of read, modify instead of update, or destroy instead of delete. It is also sometimes used to describe user interface conventions that facilitate viewing, searching, and changing information; often using computer-based forms and reports. The term was likely first popularized by James Martin in his 1983 book Managing the Data-base Environment. The acronym may be extended to CRUDL to cover listing of large data sets which bring additional complexity such as pagination when the data sets are too large to hold easily in memory.

Related Glossary Terms

Drag related terms here

Index Find Term

Chapter 1 - Introduction

DOM

The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects inHTML, XHTML and XML documents. Objects in the DOM tree may be addressed and manipulated by using methods on the objects. The public interface of a DOM is specified in its application programming interface (API). The history of the Document Object Model is intertwined with the history of the "browser wars" of the late 1990s between Netscape Navigator and Microsoft Internet Explorer, as well as with that ofJavaScript and JScript, the first scripting languages to be widely implemented in the layout engines of web browsers.

Related Glossary Terms

HTML, JavaScript

Index Find Term

Chapter 1 - Introduction

Frameworks

A particular implementation of a web application, where your code fills in the details. The framework is in charge and it calls into your code when it needs something app specific. E.g., knockout, ember, etc.

Related Glossary Terms

Library

Index Find Term

Chapter 1 - Introduction

GUI

In computing, graphical user interface (GUI, sometimes pronounced 'gooey') is a type of user interface that allowsusers to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on thekeyboard.

The actions in GUI are usually performed through direct manipulation of the graphical elements. Besides in computers, GUIs can be found in hand-held devices such as MP3 players, portable media players, gaming devices, household appliances, office, and industry equipment. The term GUI is usually not applied to other low-resolution types of interfaceswith display resolutions, such as video games (where HUD is preferred), or not restricted to flat screens, like volumetric displays because the term is restricted to the scope of two-dimensional display screens able to describe generic information, in the tradition of the computer science research at the PARC (Palo Alto Research Center).

Related Glossary Terms

Drag related terms here

Index Find Term

Chapter 1 - Introduction

HTML

HyperText Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within the web page content. HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example<img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, further tags, comments and other types of text-based content.

Related Glossary Terms

DOM, JavaScript

Index Find Term

Chapter 1 - Introduction

JavaScript

JavaScript (JS) is an interpreted computer programming language. As part of web browsers, implementations allow client-side scripts tointeract with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It has also become common in server-side programming, game development and the creation of desktop applications.

JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

Related Glossary Terms

DOM, HTML

Index Find Term

Chapter 1 - Introduction

Library

A collection of functions which are useful when writing web apps. Your code is in charge and it calls into the library when it sees fit. E.g., jQuery.

Related Glossary Terms

Frameworks

Index Find Term

Chapter 1 - Introduction