**Preamble**

Let’s take a step back and define our goal clearly: how to develop most efficiently a core skillset in AngularJS.

To get to this point, we’ll focus on the big concepts that an AngularJS coder needs to understand, and subsequent posts will address these in detail.

1. Understand Angular’s approach to the Model-View-Controller (MVC) programming paradigm

2. Implement the basic nuts-and-bolts of Angular: views, models, scope, two-way databinding, and controllers. Our hands-on how-to guide.

3. Routing – how to use Angular’s page routing system to implement single-page-applications that are the new popular approach to modern Javascript/AJAX style development

4. Directives – how Angular allows us to create special active extensions to HTML

5. CSS integration and creating polished interfaces

Understanding these concepts will win you basic competency in Angular, and pave the way for intermediate and advanced work later.

**AngularJS: Introduction And Hello World Example**

What is AngularJS? AngularJS is the (relatively) new kid on the coding block. To quote from their website it is “a structural framework for dynamic web apps” which is especially well-suited for building one-page web apps, although it’s certainly not limited to that.

Developed in 2009 by Miško Hevery and Adam Abrons — both Google employees at that time — it is entirely JavaScript and entirely client-side, so anywhere JavaScript can run, AngularJS can run. It’s also small: compressed and minified it’s less than 29 kb. And it’s open source under the MIT license. You can even use the logo, available under the Creative Commons Attribution-ShareAlike 3.0 Unported License.

According to Wikipedia the goal of Angular “is to augment browser-based applications with model–view–controller (MVC) capability” and it does just that, providing a binding/MVC framework. That’s two-way binding, mind you. Delicious. With a structure as simple as {{ my data }}, you bind data to your page. The $scope service detects changes to the model and modifies HTML expressions in the view by way of controllers. Working in the other direction, changes to the view are reflected in the model. This removes the need for the vast majority of data-centric DOM manipulations many of us, myself included, take for granted when working with a library like jQuery.

Angular runs right out of the box with no library dependencies, although it can be extended with the many modules that are available, and of course you can build your own to fit your specific needs. Being pure JavaScript, it also has the benefit of being server-agnostic.

Being accustomed to a powerful library like jQuery, it’s easy to want to mix it in to do things Angular can already do. Recognizing this potential pitfall, the developers have this to say: “If you’re struggling to break the habit, consider removing jQuery from your app. Really. Angular has the $http service and powerful directives that make it almost always unnecessary.” One thing is for sure, if you stick to Angular, the jQuery loops and explicit back-and-forth with the server will be absent from your code, since Angular provides such a succinct and clean method of achieving the same things.

| **Concept** | **Description** |
| --- | --- |
| [Template](https://docs.angularjs.org/guide/concepts#template) | HTML with additional markup |
| [Directives](https://docs.angularjs.org/guide/concepts#directive) | extend HTML with custom attributes and elements |
| [Model](https://docs.angularjs.org/guide/concepts#model) | the data that is shown to the user and with which the user interacts |
| [Scope](https://docs.angularjs.org/guide/concepts#scope) | context where the model is stored so that controllers, directives and expressions can access it |
| [Expressions](https://docs.angularjs.org/guide/concepts#expression) | access variables and functions from the scope |
| [Compiler](https://docs.angularjs.org/guide/concepts#compiler) | parses the template and instantiates directives and expressions |
| [Filter](https://docs.angularjs.org/guide/concepts#filter) | formats the value of an expression for display to the user |
| [View](https://docs.angularjs.org/guide/concepts#view) | what the user sees (the DOM) |
| [Data Binding](https://docs.angularjs.org/guide/concepts#databinding) | sync data between the model and the view |
| [Controller](https://docs.angularjs.org/guide/concepts#controller) | the business logic behind views |
| [Dependency Injection](https://docs.angularjs.org/guide/concepts#di) | Creates and wires objects / functions |
| [Injector](https://docs.angularjs.org/guide/concepts#injector) | dependency injection container |
| [Module](https://docs.angularjs.org/guide/concepts#module) | configures the Injector |
| [Service](https://docs.angularjs.org/guide/concepts#service) | reusable business logic independent of views |

## Directives

Angular uses directives to plug its action into the page. Directives, all prefaced with ng-, are placed in html attributes.

Some common directives that come pre-built with Angular are:

**ng-app:** this is essentially the initial entry point of Angular to the page. It tells Angular where it gets to act. <html ng-app> is all it takes to define a page as an Angular application.

**ng-bind:** changes the text of an element to the value of an expression.  
<span ng:bind=”name”></span> will display the value of ‘name’ inside the span. Any changes to ‘name’ are reflected instantly in the DOM anywhere the variable is used.

**ng-controller:** specifies the JavaScript class for the given action. Controllers are typically kept in external .js files.

**ng-repeat:** creates the very clean loop structures in your page.

AngularJS, a JavaScript framework developed by a Googler and supported by Google has become quite a buzz word in past few months. More and more developers are using it and thus the community has grown significantly. Not only they love it, but they can’t stop praising it :)

The reason is very simple. AngularJS rocks \m/ !! :D

If you come from a jQuery background and try to learn Angular, you would be amazed by this framework. How much you can do with few lines of code in Angular is mind-boggling. We remember jQuery did the same when it was first released back in 2006. Developers used to write hundreds if not thousands of lines of code in pure JavaScript just to make the damn works with different browsers. jQuery changed that paradigm with selectors, custom events, animations etc.

But as web grew, so did the web technologies. Browsers are faster and faster day by day. It just dint make any sense to render a full HTML page on server-side as browser can do the same. With Ajax picking up the pace it makes more sense to render UI of webapp dynamically. A new wave of Single Page Applications (SPA) started which lead to development of many front-end frameworks like Backbone, Knockout, Ember, Angular.

Coming back to AngularJS, I wanted to start a series of articles which describe Angular in plain vanilla fashion. Any one with basic JavaScript, HTML background can start working in AngularJS without any hassle.

Let us dive into the world of AngularJS and see what it is.

## Introduction to AngularJS

AngularJS as it says is a Superheroic JavaScript MVW framework. It assists with running single-page applications. Its goal is to augment browser-based applications with model–view–controller (MVC) capability, in an effort to make both development and testing easier.

AngularJS takes declarative programming to whole new level. It adapts and extends traditional HTML to better serve dynamic content through two-way data-binding that allows for the automatic synchronization of models and views.

### MVW – Model View Whatever

There are many software architecture pattern which separates the representation of information from the user’s interaction with it. The central ideas behind these patterns are code reusability and separation of concerns. The most famous pattern that is used widely today is MVC or Model-View-Controller. Similar to MVC, there is another pattern called MVP or Model-View-Presenter. The MVP is based on MVC and the presenter assumes the functionality of the “middle-man” (played by the controller in MVC). In MVP, all presentation logic is pushed to the presenter. Eventually, the model becomes strictly a domain model. And then there are other patterns such as MVVM or Model-View-View-Model.

Angular doesn’t care actually what software architecture pattern you want to use in your app. And thus the pattern MVW or Model-View-Whatever. A basic concept of MVW is that all definitions are associated with a named Module. Modules can then be aggregated to form complete web applications. Modules can depend on one another, so that including a single Module in your WebApplication may bring along additional functionality on which that Module depends. Angular JS provides you with rich set of APIs to define these modules and linked them together with dependency injection. We will see this in great detail in next few articles.

## Hello World, AngularJS

Before we get into any theory, let us get our hands dirty with some actual Angular code. That way would learn a great deal of whats happening.

In order to write a hello world application in Angular, all we have to do is it include Angular JS JavaScript in our HTML document.

|  |
| --- |
| <script type="text/javascript"          src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script> |

And that’s pretty much it. Now define your HTML page like below.

|  |
| --- |
| <!DOCTYPE html>  <html ng-app>  <head>      <title>Hello World, AngularJS - ViralPatel.net</title>      <script type="text/javascript"          src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>    </head>  <body>        Write some text in textbox:      <input type="text" ng-model="sometext" />        <h1>Hello {{ sometext }}</h1>    </body>  </html> |

**Online Demo**  
  
If you want to change and play around with code use this JSFiddle: <http://jsfiddle.net/viralpatel/vFcZ7/>

Just write some text in above demo and see how the value after “Hello” changes.

Note that we didn’t write a single line of JavaScript and still this example works like a charm! Let us go through the demo step by step. There are some Angular related tags we put in our HTML document.

### ng-app

First thing that we notice is an attribute ng-app within <html> tag. This attribute tells the Angular to be active in this portion of the page. So in our case entire document. If you want to enable Angular only at specific location in your webpage then you can define ng-app attribute to any DIV or other tag.

In case you are building app that also works with IE7, add id=”ngapp”. For example:

|  |
| --- |
| <html ng-app id="ng-app"> |

If you choose to use the old style directive syntax ng: then include xml-namespace in html to make IE happy.

|  |
| --- |
| <html xmlns:ng="http://angularjs.org"> |

### ng-model and Two way data binding

We define attribute ng-model in textbox as ng-model=”sometext”. ng-model binds the state of textbox with model value. In our case we define a model **sometext**. This model value is bound with the value of textbox using ng-model attribute. Thus when the textbox value changes, Angular automatically changes the model **sometext** with this value. This is called **Two way data binding**. Similarly, if we change the model value than Angular will change the value of textbox. Two way data binding is the core of Angular’s magical spell. It just works. You’ll get to know about it more once we start adding complexity in our application.

AngularJS two-way data binding is its most notable feature and reduces the amount of code written by relieving the server backend from templating responsibilities.

### {{ sometext }}

Note how we wrap our model value in double curly braces. This tell Angular to bind the value of model**sometext** in place of {{ sometext }}.

Thus any change in **sometext** model value changes the text inside <h1> tag.

### ng-show / ng-hide

Now lets further modify our demo and add one more Angular attribute ng-show. In below code, we added attribute ng-show=”sometext” to <h1> tag.

|  |
| --- |
| <script type="text/javascript"          src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script> |

And that’s pretty much it. Now define your HTML page like below.

|  |
| --- |
| <!DOCTYPE html>  <html ng-app>  <head>      <title>Hello World, AngularJS - ViralPatel.net</title>      <script type="text/javascript"          src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>    </head>  <body>        Write some text in textbox:      <input type="text" ng-model="sometext" />        <h1 ng-show="sometext">Hello {{ sometext }}</h1>    </body>  </html> |

**Online Demo**

If you want to change and play around with code use this JSFiddle: <http://jsfiddle.net/viralpatel/ppgsS/>

In this demo, the text Hello does not appear unless we type anything in textbox. We added just one small attribute to <H1> tag and see how the functionality changed!.

ng-show attribute conditionally show an element, depending on the value of a boolean expression. Similar to ng-show you can also use ng-hide, which exactly does opposite of ng-show.

Note that until now we have not written any JavaScript code at all. Still our application became so dynamic.

## AngularJS Filters

AngularJS provides powerful mechanism to modify the data on the go using Filters. Filters typically transform the data to a new data type, formatting the data in the process. The general syntax for using filter is:

|  |
| --- |
| {{ expression | filter }} |

You can use more than filter on an expression by chaining them like:

|  |
| --- |
| {{ expression | filter1 | filter2 }} |

AngularJS by default provide few filters that we can use in our app. It is also possible to define your own custom filters. For now we will just check filters that Angular provide with framework.

### Filter uppercase and lowercase

As its name suggest, this filter convert the expression into uppercase letters. Lets check a quick demo. Lets modify few lines from our above example and use uppercase filter.

|  |
| --- |
| <!DOCTYPE html>  <html ng-app>  <head>      <title>Hello World, AngularJS - ViralPatel.net</title>      <script type="text/javascript"          src="//ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>    </head>  <body>        Write some text in textbox:      <input type="text" ng-model="sometext" />        <h1>Hello {{ sometext }}</h1>        <h4>Uppercase: {{ sometext | uppercase }}</h4>      <h4>Lowercase: {{ sometext | lowercase }}</h4>    </body>  </html> |

**Online Demo**

If you want to change and play around with code use this JSFiddle: <http://jsfiddle.net/viralpatel/nZ5sH/>

Notice when you type in the textbox, the value is converted to upper and lower case depending on the filter we used.

Similarly, Angular provides some more filters like:

#### date

Usage:

|  |
| --- |
| {{ date\_expression | date[:format] }} |

Formats date to a string based on the requested format. Read [Angular documentation](http://docs.angularjs.org/api/ng.filter:date) to know more about format.

#### number

Usage:

|  |
| --- |
| {{ number\_expression | number[:fractionSize] }} |

Formats a number as text. If the input is not a number an empty string is returned.

There are more filters like json, limitTo, filter, orderBy. We will go through them in next few articles as and when we use them. For now you can refer to Filter documentation for more details.

## That’s All Folks

We have just scratched the surface of a big ocean. AngularJS offers so much more that we can only touch different aspects once we actually start using it.

In this tutorial we went through an introduction of AngularJS and also learned how to set it up in any webapplication. Also we saw different angular attributes like ng-app, ng-model, ng-show etc and their usage. Then we checked filters in angular. We saw few basic filters like uppercase, lowercase.

 where we will see everything about **AngularJS Controllers** and **$scope**.**Update**: Next tutorial is published. Read [AngularJS Controller Tutorial](http://viralpatel.net/blogs/angularjs-controller-tutorial/).

I hope you like the series of tutorials on AngularJS.