**Learning Angular JS**

**Angular Keys**

Angular JS is an **open source web-app framework** maintained by Google.  
It supports MVC architecture for single page client side apps.  
**ng-app** attribute in HTML is used to bootstrap AngularJS application.

**Module:**

* It is a **container** for different parts of app like controllers, services, filters, directives, etc.
* Roles of modules –

1. To associate an AngularJS application with a region of an HTML document
2. To act as a gateway to key AngularJS framework features
3. To help organize the code and components in an AngularJS application

* **angular.module** method returns a Module object which provides access to different features like animation, config, controllers, providers, factory, etc.
* The **angular.module** is a global place for creating, registering and retrieving Angular modules
* Methods of Module returns a Module object allowing chaining of methods i.e., *fluent API*
* Categories of methods defined by Module object –

1. That define components
2. That make it easier to create those components
3. That help manage AngularJS life cycle

* All components/building blocks of Module accepts *factory* functions as arguments

**Controller:**

* It is a big component or building block of an AngularJS application
* Acts as a conduit (bridge) b/w model and views
* It should be used only for business logic for testability
* Defined using **Module.controller** method that takes *String* (controller name) and *factory* function as arguments
* *factory* function is used to prepare controller for use
* Controller’s dependencies are declared using *factory* function
* ng-controller attribute is used to bind it with HTML

**Directive:**

* Markers forHTMLDOM element
* Can be applied as element (restrict E), attribute (restrict A), comment (restrict M) or CSS (restrict C)
* It tells angular JS compiler to perform a specified behavior on HTML DOM element or transform DOM element
* It is the extended HTML attribute/element
* Custom directives are defined using **Module.directive** method
* compile() and link() functions are used when HTML DOM is to be manipulated
* **compile()** function is called once for each occurrence of a directive; any one-time configuration can be done using it
* **compile()** function returns a **link()** function
* **link()** function binds element data with $scope and is called every time element data is modified
* **ng-bind** used for one-way data binding
* **ng-model** used for two-way data binding
* **ng-repeat** used to repeat element
* **ng-switch** and **ng-include** used for displaying partial views based on some conditions
* **ng-cloak** used for preventing display of inline expressions in case of slow data loading
* **Reference:** <http://www.ng-newsletter.com/posts/directives.html>

***factory* functions:**

* They create an object that AngularJS uses to perform the actual work

**Filters:**

* Used in views, controllers, services to format data
* Custom filters can be created using **Module.filter**
* Filters are applied to html as *{{expression | filterName}}*
* AngularJS built in filters are –

1. ***currency*** Format a number to a currency format
2. ***filter*** Select a subset of items from an array
3. ***lowercase*** Format a string to lower case
4. ***orderBy*** Orders an array by an expression
5. ***uppercase*** Format a string to upper case

**Services:**

* Objects that provide a functionality to use throughout an app
* Services are javascript functions and are responsible to do a specific tasks only
* Consumed by controllers, filters as per requirement
* Injected using DI (Dependency Injection) mechanism of AngularJS
* Angular services are –
  1. ***Lazily instantiated*** Angular only instantiates a service when an application component depends on it
  2. ***Singletons*** Only one instance of the object will be created by AngularJS and shared between the parts of the application that need the service only one instance of the object will be created by AngularJS and shared between the parts of the application that need the service
* Can be created using any of three methods –

1. service
2. factory
3. provider

**Values:**

* Created using **Module.value** method
* Returns fixed values and objects
* It simplifies unit testing and allows to use advanced features like decoration
* Injected as a DI
* Cannot be declared as dependency in **Module.config** method

**Module Life Cycle:**

* **run()** method is invoked after all modules are loaded and their dependencies are resolved
* **config()** method is invoked after the module on which is called is loaded and its dependencies are resolved
* **Ex: example.main** and **example.services** are two modules declared with **example.main** having **example.services** added as dependency  
  callback functions (**run()** and **config()**) are declared in both modules  
  Below is the sequence in which callback functions are invoked:
  1. The **config()** callback on the **example.services** module
  2. The **config()** callback on the **example.main** module
  3. The **run()** callback on the **example.services** module
  4. The **run()** callback on the **example.main** module