AngularJS is a client side JavaScript MVC framework to develop a dynamic web application.

Requirement for this class in knowledge of JavaScript JS but if you do not know we will learn in few weeks class the basic of JS and also such as HTML, CSS, AJAX, etc.

Note: Serious students needed for this class who do the homework on time. This class requires lot of time to practice the examples and homework on daily basis, not in the class only.

We need the following tools to setup a development environment for AngularJS:

1. AngularJS Library
2. Editor/IDE
3. Browser
4. Web server

Link: [**angularjs.org**](https://angularjs.org/)

The following editors are recommended:

* [Sublime Text](http://www.sublimetext.com/)
* [Aptana Studio 3](http://www.aptana.com/)
* [Ultra Edit](http://www.ultraedit.com/)
* [Eclipse](https://eclipse.org/)
* [Visual Studio](https://www.visualstudio.com/)
* Basic Introduction to AngularJS
	+ What does AngularJS do for us?
	+ Controls AngularJS?
	+ How can we get AngularJS?
* Create a simple AngularJS web application step by step and understand the basic building blocks of AngularJS. Using angular-seed. Two-way data binding and Directives (Directives are markers (attributes) on a DOM element that tell AngularJS to attach a specific behavior to that DOM element or even transform the DOM element and its children).
* The following table lists all the important concepts in AngularJS:

Template ---> HTML with additional markup

Directives ----> Extends the HTML with custom attributes and elements

Model -----> The data shown to the user in the view and with which the user interacts

Scope ----> A context where the model is stored so that controllers, directives and expressions can access it

Expressions ----> Executes JavaScript code inside brackets {{ }}.

* + Compiler ==> Parses the template and instantiates directives and expressions Expressions are lightweight code snippets
	+ Expression capabilities
	+ Limitations
	+ The border between expressions and $eval

Filter ===> Formats the value of an expression for display to the user

View ===> what the user sees (the DOM) Angular's take on the View: a little bit different

* + Tying a View to a Controller
	+ Tying a View to a model

Data Binding ===> Sync data between the model and the view

* + Controller Maintains the application data and business logic - Where Controllers fit in, and what they do, from Angular’s perspective
	+ Managing Scope
	+ Setting up Behavior
	+ Building a basic controller
	+ A more advanced controller
	+ Module a container for different parts of an app including controllers, services, filters, directives which configure the Injector How to create a model
	+ Explicit models
	+ Implicit models

Service Reusable business logic, independent of views

Dependency Injection Creates and wires objects and functions

Injection Dependency injection container

* Fit together. How much of the page is an Angular application?
	+ How it fits together
		- How much of the page is an Angular application?
	+ Model, View, Controller from the AngularJS Perspective
* Single page application
	+ What do we mean by Single Page Application?
	+ How to Create Angular Modules
	+ Define Angular's Routing Service
		- Routing Basics
		- Accessing URL Data
		- How to use the $location Service
	+ Creating a Skeleton Single Page Application
* Filters
	+ Standard filters
	+ Writing your own filter
	+ Tying filters together
* Scopes
	+ What are scopes?
	+ What do scopes provide?
	+ Scope lifecycle
	+ Scopes as glue between controller and view
	+ Scope hierarchies
* Scope and eventsAngular Forms
	+ Angular forms vs HTML forms
	+ Angular form controls
	+ Form events
	+ The form controller
	+ Form validation
		- CSS classes for form data
* Ajax, Data, and Angular
	+ High level interactions with servers
	+ Low-level server interactions with $http
	+ The deferred/promises API
	+ Making RESTful Service calls with $resource
* Directives
	+ Teaching HTML new tricks
	+ Binding text and attributes
	+ Directive processing lifecycle
		- DOM Processing
		- Compilation
		- Linking
	+ A basic directive
	+ Directives and scopes
	+ Creating reusable directives
	+ Turning directives into components
* Testing in Angular
	+ Unit testing
		- Working with Dependency Injection
		- Other unit testing issues
	+ User End-to-end testing
		- Angular’s E2E testing framework
		- Commands and expectations
		- Controlling what happens before and after the test
		- Running a scenario