**Angular JS :**

Angular JS :

Overview,

MVC architecture,

directives,

expression,

controllers,

tables,

filters,

modules,

forms,

includes,

views,

scopes,

services

dependency injection

custom directives

Internationalization.

**Introduction to NodeJS and Struts:**

Overview, architecture, configuration, actions, interceptors, result types, validations, localization, exception handling, annotations.

**AngularJS is a structural framework** for dynamic web apps.

With AngularJS, designers can use HTML as the template language and it allows for the extension of HTML's syntax to convey the application's components effortlessly. Angular makes much of the code you would otherwise have to write completely redundant.

**AngularJS** is a very powerful JavaScript Framework. It is used in Single Page Application (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions. AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

Why to Learn AngularJS?

AngularJS is an open-source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google. Its latest version is 1.2.21.

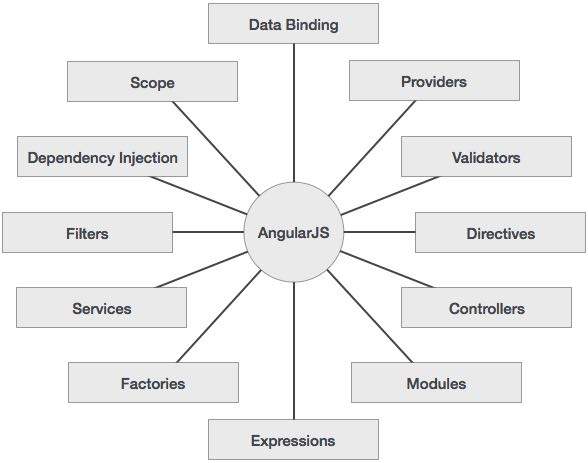
The general features of AngularJS are as follows −

* AngularJS is a efficient framework that can create Rich Internet Applications (RIA).
* AngularJS provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.
* Applications written in AngularJS are cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

Overall, AngularJS is a framework to build large scale, high-performance, and easyto-maintain web applications.

The core features of AngularJS are as follows –

* **Data-binding** − It is the automatic synchronization of data between model and view components.
* **Scope** − These are objects that refer to the model. They act as a glue between controller and view.
* **Controller** − These are JavaScript functions bound to a particular scope.
* **Services** − AngularJS comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
* **Filters** − These select a subset of items from an array and returns a new array.
* **Directives** − Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.
* **Templates** − These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using *partials*.
* **Routing** − It is concept of switching views.
* **Model View Whatever** − MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
* **Deep Linking** − Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
* **Dependency Injection** − AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.



## Advantages of AngularJS

The advantages of AngularJS are −

* It provides the capability to create Single Page Application in a very clean and maintainable way.
* It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.
* AngularJS code is unit testable.
* AngularJS uses dependency injection and make use of separation of concerns.
* AngularJS provides reusable components.
* With AngularJS, the developers can achieve more functionality with short code.
* In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

On the top of everything, AngularJS applications can run on all major browsers and smart phones, including Android and iOS based phones/tablets.

## Disadvantages of AngularJS

Though AngularJS comes with a lot of merits, here are some points of concern −

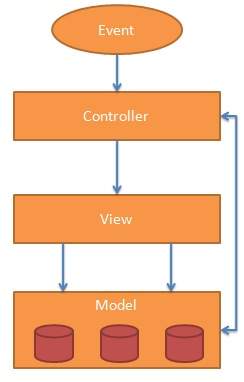
* **Not Secure** − Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
* **Not degradable** − If the user of your application disables JavaScript, then nothing would be visible, except the basic page.

# AngularJS - MVC Architecture

**M**odel **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts −

* **Model** − It is the lowest level of the pattern responsible for maintaining data.
* **View** − It is responsible for displaying all or a portion of the data to the user.
* **Controller** − It is a software Code that controls the interactions between the Model and View.

MVC is popular because it isolates the application logic from the user interface layer and supports separation of concerns. The controller receives all requests for the application and then works with the model to prepare any data needed by the view. The view then uses the data prepared by the controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.



## The Model

The model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself.

## The View

A presentation of data in a particular format, triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.

## The Controller

The controller responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

## AngularJS Directives

The AngularJS framework can be divided into three major parts −

* **ng-app** − This directive defines and links an AngularJS application to HTML.
* **ng-model** − This directive binds the values of AngularJS application data to HTML input controls.
* **ng-bind** − This directive binds the AngularJS application data to HTML tags.

## Creating AngularJS Application

### Step 1: Load framework

Being a pure JavaScript framework, it can be added using <Script> tag.

<script

src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">

</script>

### Step 2: Define AngularJS application using ng-app directive

<div ng-app = "">

...

</div>

### Step 3: Define a model name using ng-model directive

<p>Enter your Name: <input type = "text" ng-model = "name"></p>

### Step 4: Bind the value of above model defined using ng-bind directive

<p>Hello <span ng-bind = "name"></span>!</p>

* The ng-app directive indicates the start of AngularJS application.
* The ng-model directive creates a model variable named name, which can be used with the HTML page and within the div having ng-app directive.
* The ng-bind then uses the name model to be displayed in the HTML <span> tag whenever user enters input in the text box.
* Closing</div> tag indicates the end of AngularJS application.

# AngularJS - Directives

AngularJS directives are used to extend HTML. They are special attributes starting with **ng**-prefix. Let us discuss the following directives −

* **ng-app** − This directive starts an AngularJS Application.
* **ng-init** − This directive initializes application data.
* **ng-model** − This directive defines the model that is variable to be used in AngularJS.
* **ng-repeat** − This directive repeats HTML elements for each item in a collection.

# AngularJS - Expressions

Expressions are used to bind application data to HTML. Expressions are written inside double curly braces such as in {{ expression}}. Expressions behave similar to ngbind directives. AngularJS expressions are pure JavaScript expressions and output the data where they are used.

## Using numbers

<p>Expense on Books : {{cost \* quantity}} Rs</p>

## Using Strings

<p>Hello {{student.firstname + " " + student.lastname}}!</p>

## Using Object

<p>Roll No: {{student.rollno}}</p>

## Using Array

<p>Marks(Math): {{marks[3]}}</p>

# AngularJS - Controllers

AngularJS controllers **control the data** of AngularJS applications.

AngularJS controllers are regular **JavaScript Objects**.

AngularJS applications are controlled by controllers.

The **ng-controller** directive defines the application controller.

A controller is a **JavaScript Object**, created by a standard JavaScript **object constructor**.

AngularJS application mainly relies on controllers to control the flow of data in the application. A controller is defined using *ng-controller* directive. A controller is a JavaScript object that contains attributes/properties, and functions. Each controller accepts $scope as a parameter, which refers to the application/module that the controller needs to handle.

<div ng-app = "" ng-controller = "studentController">

...

</div>

# AngularJS - Tables

Table data is generally repeatable. The ng-repeat directive can be used to draw table easily. The following example shows the use of ng-repeat directive to draw a table –

<table>

<tr>

<th>Name</th>

<th>Marks</th>

</tr>

<tr ng-repeat = "subject in student.subjects">

<td>{{ subject.name }}</td>

<td>{{ subject.marks }}</td>

</tr>

</table>

# AngularJS - Filters

Filters are used to modify the data. They can be clubbed in expression or directives using pipe (|) character. The following list shows the commonly used filters.

* currency Format a number to a currency format.
* date Format a date to a specified format.
* filter Select a subset of items from an array.
* json Format an object to a JSON string.
* limitTo Limits an array/string, into a specified number of elements/characters.
* lowercase Format a string to lower case.
* number Format a number to a string.
* orderBy Orders an array by an expression.
* uppercase Format a string to upper case.

<p>The name is {{ firstName | lowercase }}</p>

<p>The name is {{ lastName | uppercase }}</p>

<li ng-repeat="x in names | orderBy:'country'">

<h1>Price: {{ price | currency }}</h1>

Filter filter

<li ng-repeat="x in names | filter : 'i'">

## Filter an Array Based on User Input

By setting the ng-model directive on an input field, we can use the value of the input field as an expression in a filter.

<li ng-repeat="x in names | filter:test">

Type a letter in the input field, and the list will shrink/grow depending on the match:

# AngularJS - Modules

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Controllers always belong to a module.

AngularJS supports modular approach. Modules are used to separate logic such as services, controllers, application etc. from the code and maintain the code clean. We define modules in separate js files and name them as per the module.js file. In the following example, we are going to create two modules −

* **Application Module** − used to initialize an application with controller(s).
* **Controller Module** − used to define the controller.

## Application Module

Here is a file named *mainApp.js* that contains the following code −

var mainApp = angular.module("mainApp", []);

Here, we declare an application **mainApp** module using angular.module function and pass an empty array to it. This array generally contains dependent modules.

## Controller Module

mainApp.controller("studentController", function($scope) {

$scope.student = {

firstName: "Mahesh",

lastName: "Parashar",

fees:500,

subjects:[

{name:'Physics',marks:70},

{name:'Chemistry',marks:80},

{name:'Math',marks:65},

{name:'English',marks:75},

{name:'Hindi',marks:67}

],

fullName: function() {

var studentObject;

studentObject = $scope.student;

return studentObject.firstName + " " + studentObject.lastName;

}

};

});

Here, we declare a controller **studentController** module using mainApp.controller function

## Use Modules

<div ng-app = "mainApp" ng-controller = "studentController">

...

<script src = "mainApp.js"></script>

<script src = "studentController.js"></script>

</div>

Here, we use application module using ng-app directive, and controller using ngcontroller directive. We import the mainApp.js and studentController.js in the main HTML page.

## Modules and Controllers in Files

It is common in AngularJS applications to put the module and the controllers in JavaScript files.

In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller:

<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  
<body>  
<div ng-app="**myApp**" ng-controller="**myCtrl**">  
{{ firstName + " " + lastName }}  
</div>  
<script src="**myApp.js**"></script>  
<script src="**myCtrl.js**"></script>  
  
</body>  
</html>

# AngularJS - Forms

Forms in AngularJS provides data-binding and validation of input controls.

AngularJS enriches form filling and validation. We can use ng-click event to handle the click button and use $dirty and $invalid flags to do the validation in a seamless way. Use novalidate with a form declaration to disable any browser-specific validation. The form controls make heavy use of AngularJS events. Let us have a look at the events first.

## Events

AngularJS provides multiple events associated with the HTML controls. For example, ng-click directive is generally associated with a button. AngularJS supports the following events −

* ng-click
* ng-dbl-click
* ng-mousedown
* ng-mouseup
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-keydown
* ng-keyup
* ng-keypress
* ng-change

input controls are the HTML input elements:

* input elements
* select elements
* button elements
* textarea elements

## Data-Binding

Input controls provides data-binding by using the ng-model directive.

<input type="text" ng-model="firstname">

<form>  
  First Name: <input type="text" ng-model="firstname">  
</form>  
  
<h1>You entered: {{firstname}}</h1>

## Checkbox

A checkbox has the value true or false. Apply the ng-model directive to a checkbox, and use its value in your application

<form>  
  Check to show a header:  
  <input type="checkbox" ng-model="myVar">  
</form>  
  
<h1 ng-show="myVar">My Header</h1>

## Radiobuttons

Bind radio buttons to your application with the ng-model directive.

Radio buttons with the same ng-model can have different values, but only the selected one will be used.

### Example

Display some text, based on the value of the selected radio button:

<form>  
  Pick a topic:  
  <input type="radio" ng-model="myVar" value="dogs">Dogs  
  <input type="radio" ng-model="myVar" value="tuts">Tutorials  
  <input type="radio" ng-model="myVar" value="cars">Cars  
</form>

## Selectbox

Bind select boxes to your application with the ng-model directive.

The property defined in the ng-model attribute will have the value of the selected option in the selectbox.

### Example

Display some text, based on the value of the selected option:

<form>  
  Select a topic:  
  <select ng-model="myVar">  
    <option value="">  
    <option value="dogs">Dogs  
    <option value="tuts">Tutorials  
    <option value="cars">Cars  
  </select>  
</form>

## AngularJS Validation

## <h1>{{myForm.myInput.$valid}}</h1>

# AngularJS - Includes

<div ng-app = "" ng-controller = "studentController">

<div ng-include = "'main.htm'"></div>

<div ng-include = "'subjects.htm'"></div>

</div>