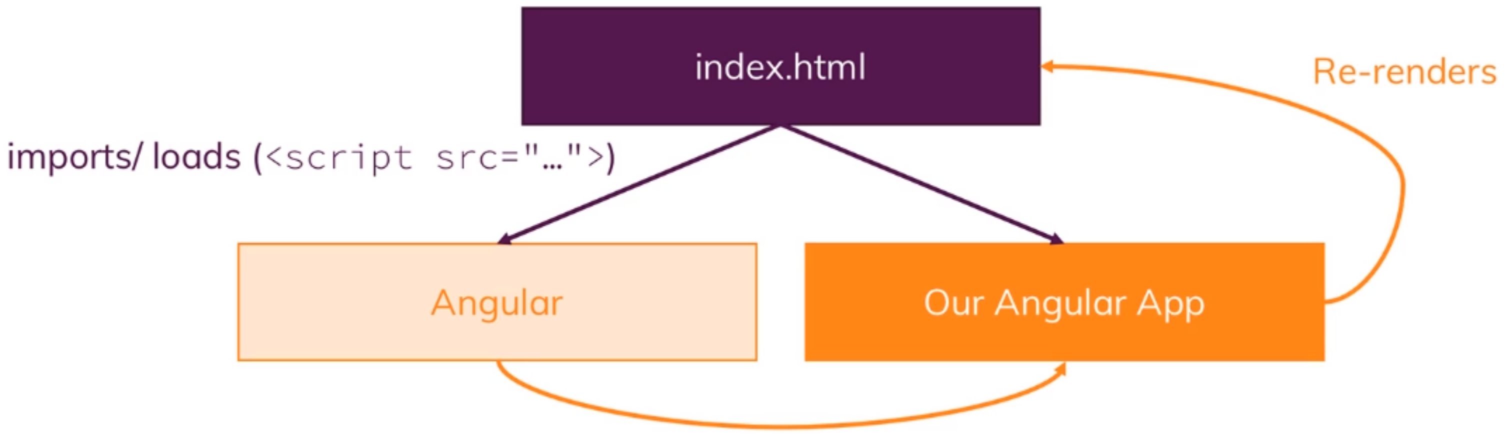
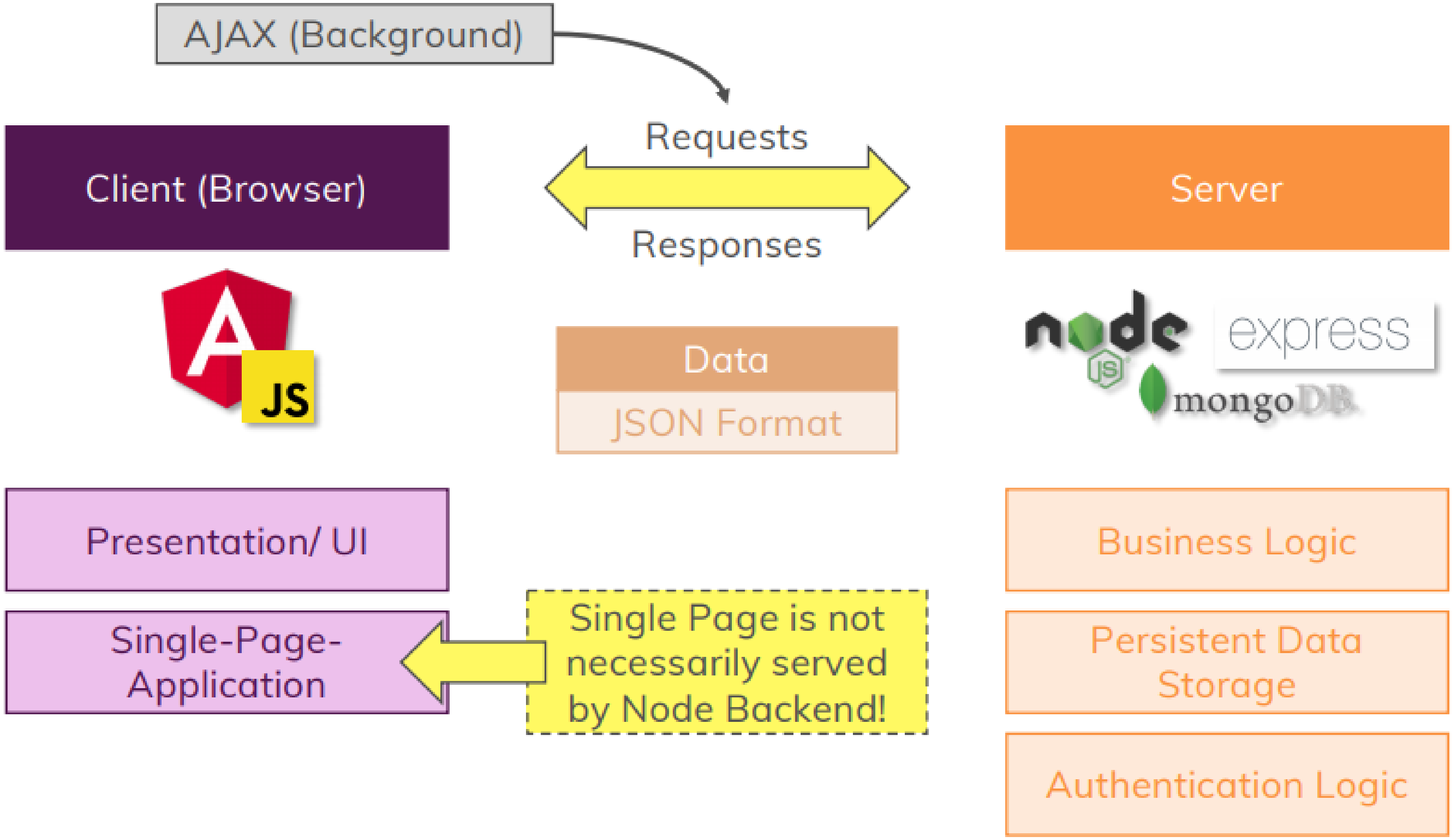
We Build a Single Page Application



This allows for instant re-rendering, instant user feedback and makes building highly engaging UIs possible


AJAX (Background)
Requests
Responses
Client (Browser)
Server
Data
JSON Format
Presentation/ UI
Single-Page-Application
Single Page is not necessarily served by Node Backend!
Business Logic
Persistent Data Storage
Authentication Logic

Angular

Angular is a JavaScript Framework

Angular is used to build client-side applications using HTML

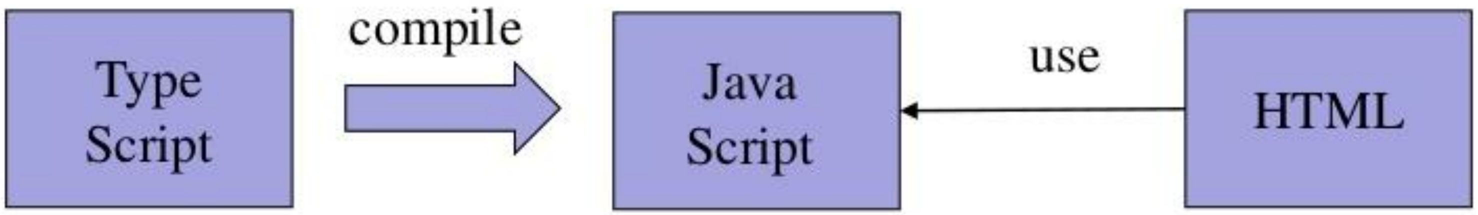
Angular bootstraps JavaScript with HTML tags.

Angular is used to make reach UI application.

Angular enhances UI experience for User.

Angular code is written in TypeScript language

- TypeScript is compiled into JavaScript
- JavaScript is used in HTML pages



Angular enhances HTML

Angular has set of directives to display dynamic contents at HTML page. Angular extends HTML node capabilities for a web application.

Angular provides data binding and dependency injection that reduces line of code.

Angular extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

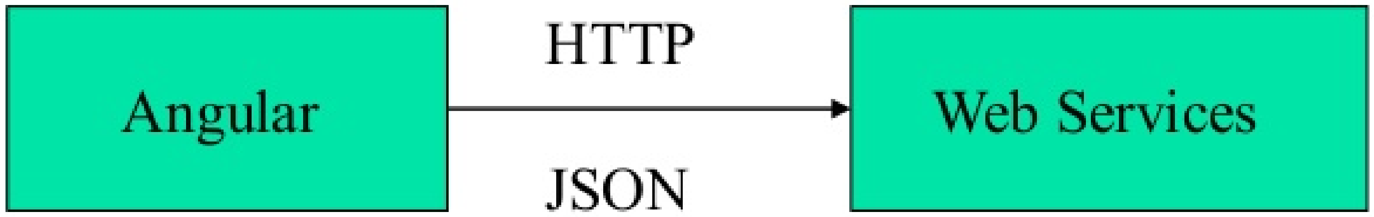
Angular follows MVC Architecture

Angular – REST web services

Angular communicates with RESTFul web services in modern applications

RESTful Web Services are accessed by HTTP call

RESTful Web Services exchange JSON data



Angular Application

An application is a Module

Module contains components

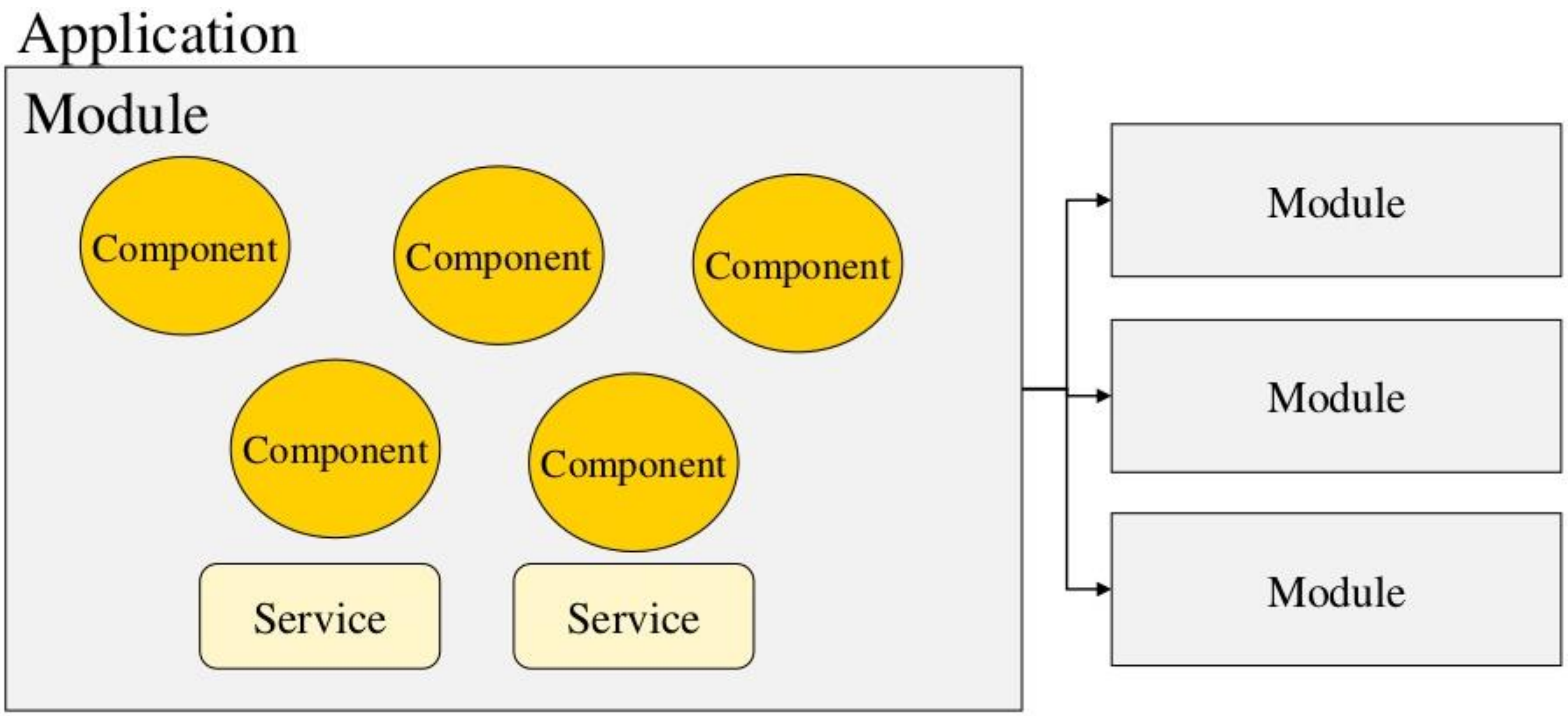
Component uses Services

Services contains data and reusable business methods

Basic building block of Angular is Component

It is said that Angular follows Component/Service architecture. Internally it follows MVC Architecture

Angular Application



- An Application is a Module
- Modules are reusable
- One Module’s components and services can be used by another module

AngularJS

There are several benefits as listed below:

1) Dependency Injection

- In software engineering, dependency injection refers to the passing of objects between the application and the client.

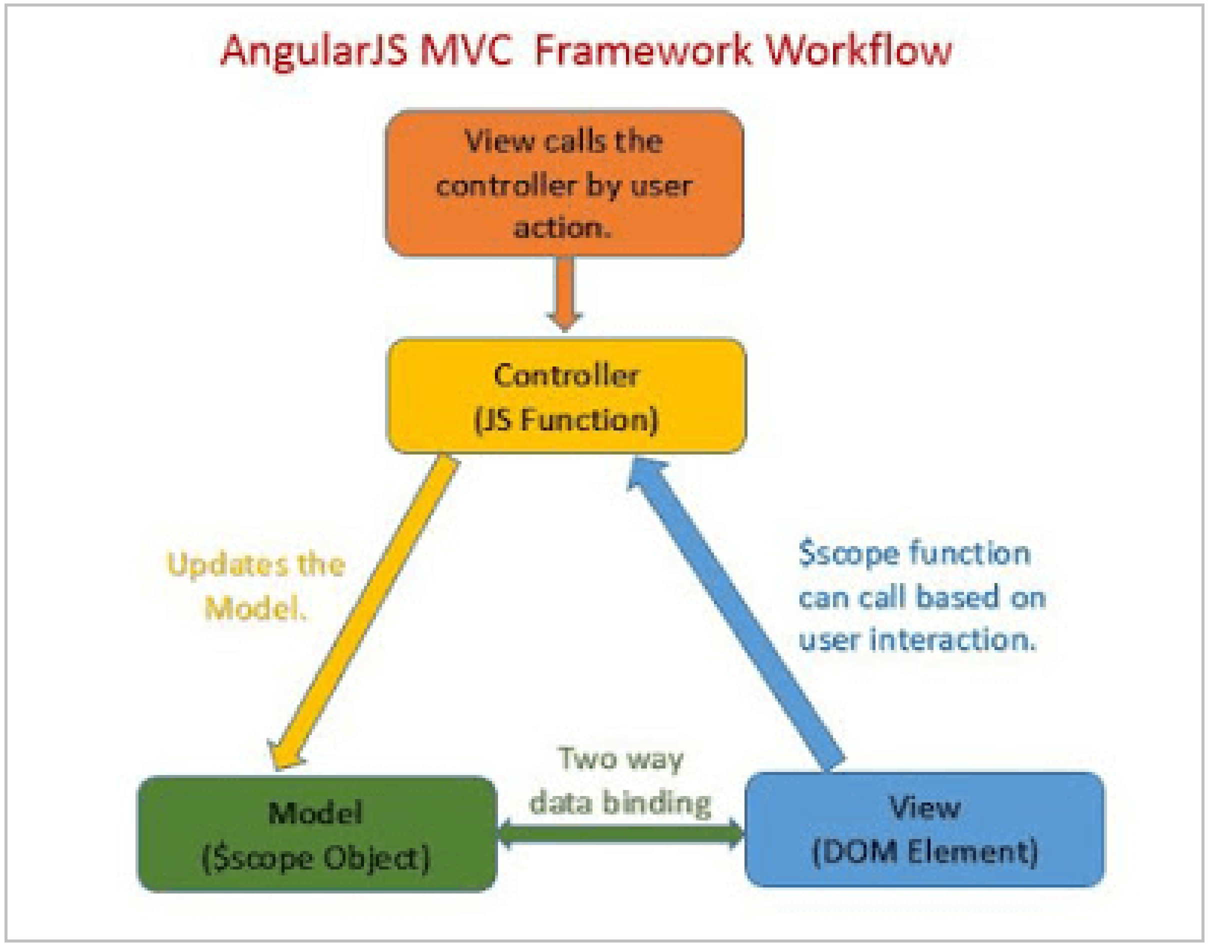
- Injection is the phenomenon of passing a dependency (say an application service) to a dependent object (say a client) that would use it.

- AngularJS provides several core components for achieving this purpose in simplicity.

AngularJS

2) Model View Controller

- AngularJS is used to create Rich Internet Applications (RIA), and two-way data binding is achievable due to the MVC (model view controller) architecture in Angular JS.


AngularJS MVC Framework Workflow
View calls the controller by user action.
Controller (JS Function)
Updates the Model.
$scope function can call based on user interaction.
Two way data binding
Model ($scope Object)
View (DOM Element)

MVC Architecture

Model View Controller 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.

MVC Architecture

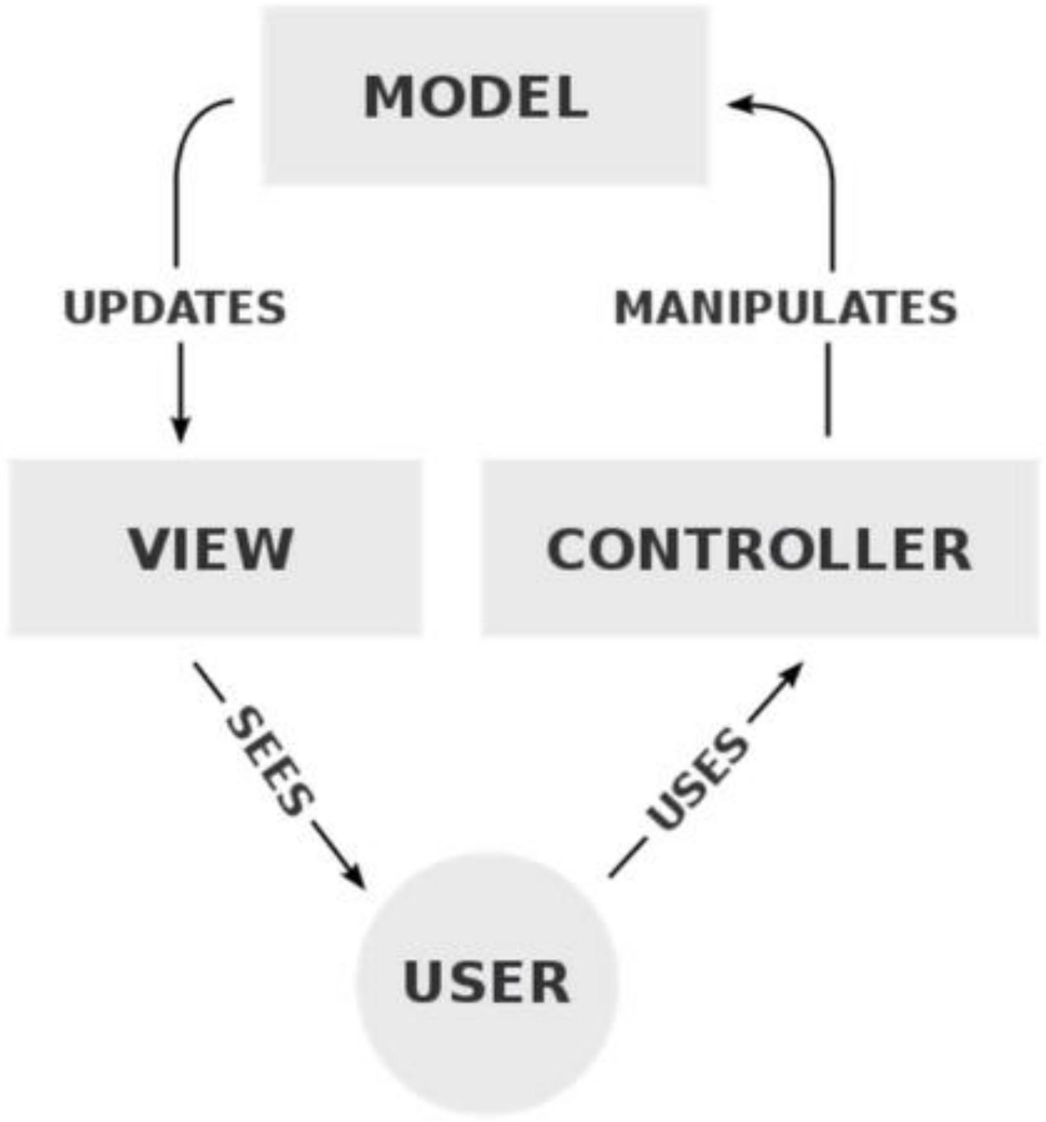
A Model View Controller pattern is made up of the following three parts:

- Model: Where the data of our application is stored
    - Directly manages the data, logic and rules of the application.
    - Stores the data retrieved by the controller and displayed in the view.

- View: What the user sees
    - Any output representation of information, such as chart or diagram
    - Used to generate an output representation to the user.

- Controller: Where the programming work is done.
    - Accepts inputs and converts it to commands for the model or view.
    - Send commands to the model to update the model’s state.
    - It can also send commands to it’s associated view to change the view’s presentation of the model.

MVC Architecture



MVC Architecture

- **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

3) Two-way Data Binding

- Software changes should be responsive, and changes within the system should be catered to the changes in the user interface and conversely, with precision and speed.

- AngularJS offers this kind of binding by synchronizing between the model and the view.

4) Testing

- It is interesting to know the fact that AngularJS was designed keeping testing in mind, right from the beginning.

- Any of the components of AngularJS can be comfortably tested using both unit testing and an end to end testing.

- The application can be transported across browsers for testing purposes.

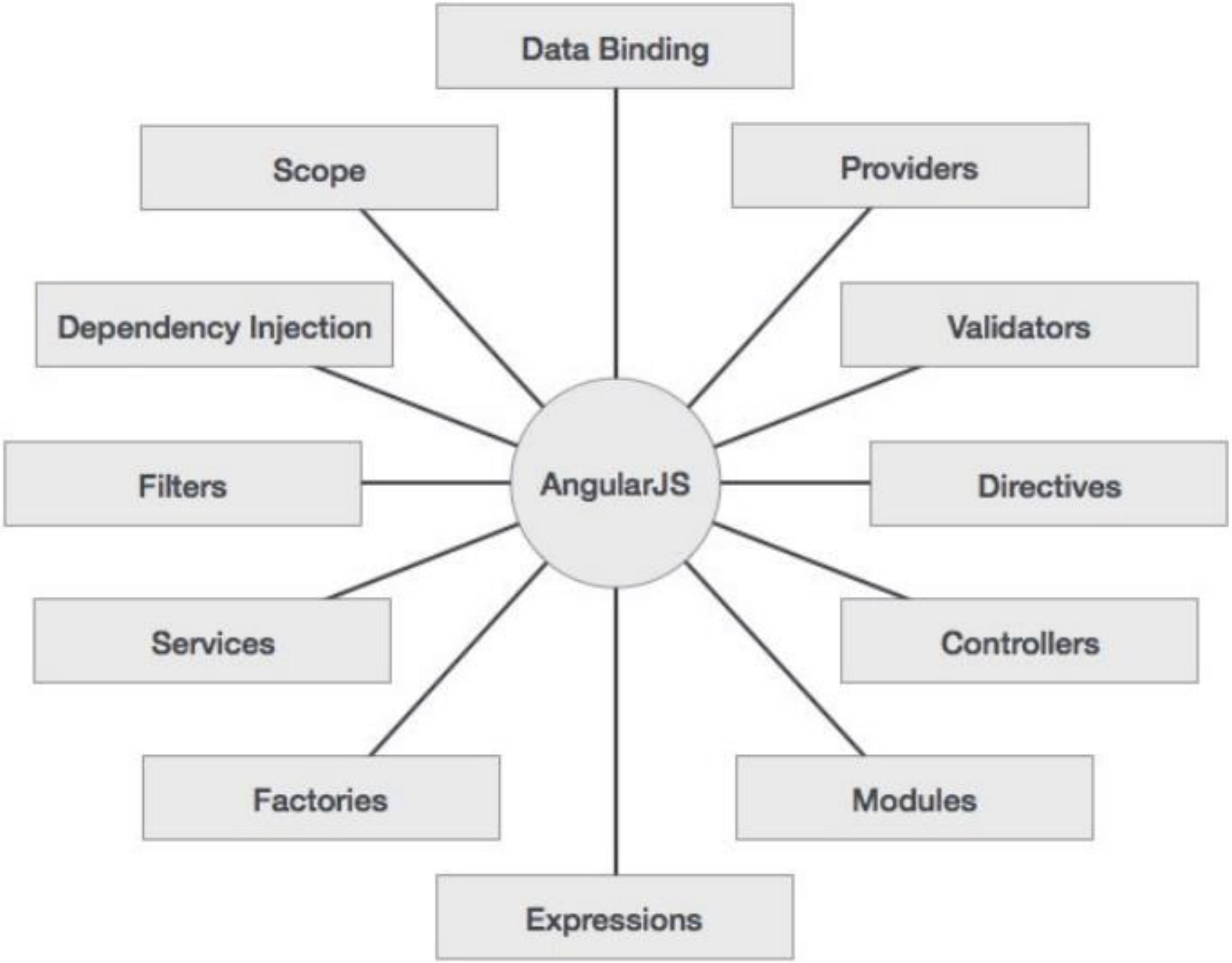
AngularJS

5) Controlling the behavior of DOM elements

- Attributes of AngularJS can be linked to directives so that automatic initialization of the application is possible.

- This means that there is modularity in AngularJS and with the help of its features such as directives and filters, a sense of customization and flexibility can be achieved in the code.

AngularJS Features



AngularJS Features

- 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.

AngularJS Features

- 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.

- 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.

AngularJS Features

- 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).

AngularJS Advantages

- 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.

AngularJS Advantages

- 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.

- AngularJS Disadvantages

- 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 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.

AngularJS Directives

What happens when the page is loaded in the browser ? Let us see:

- HTML document is loaded into the browser, and evaluated by the browser.
- AngularJS JavaScript file is loaded, the angular global object is created.
- The JavaScript which registers controller functions is executed.
- AngularJS scans through the HTML to search for AngularJS apps as well as views.
- Once the view is located, it connects that view to the corresponding controller function.
- AngularJS executes the controller functions.
- It then renders the views with data from the model populated by the controller.
- The page is now ready.

AngularJS: JavaScript Framework

- AngularJS is a JavaScript framework written in JavaScript.
- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
```

AngularJS: JavaScript Framework

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>

<div ng-app="">                                  ---"owner" of an AngularJS application

<p>Input something in the input box:</p>

<p> Name:
        <input type="text" ng-model="name">
</p>                                              ---binds input field to var Name
<p ng-bind="name"></p>                            ---binds <p> field to var Name

</div>

</body>
</html>
```

AngularJS: Directives

- AngularJS directives are extended HTML attributes with the prefix ng-.
- The ng-app directive initializes an AngularJS application.
- The ng-init directive initializes application data.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

AngularJS: Directives example

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="" ng-init="firstName='John'">

<p>The name is
        <span ng-bind="firstName"> </span>
</p>

</div>

</body>
</html>
```

AngularJS: Expression

- AngularJS expressions are written inside double braces: {{ expression }}.
- AngularJS expressions bind AngularJS data to HTML the same way as the ng-bind directive.

Example:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">
<p>My first expression: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">

<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="name"></p>
<p>{{name}}</p>

</div>

</body>
</html>
```

AngularJS: Expression

- 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>
```

Example:

```
<div ng-app = "" ng-init = "quantity = 1;cost = 30;
        student = {firstname:'Mahesh',lastname:'Parashar',rollno:101};
        marks = [80,90,75,73,60]">
        <p>Hello {{student.firstname + " " + student.lastname}}!</p>

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

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

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

 </div>
```

Output:

```
Sample Application
Hello Mahesh Parashar!
Expense on Books : 30 Rs
Roll No: 101
Marks(Math): 73
```

AngularJS Expressions vs. JavaScript Expressions

- Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.

- Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.

- AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.

- AngularJS expressions support filters, while JavaScript expressions do not.

AngularJS: Application

- AngularJS modules define AngularJS applications.
- AngularJS controllers control AngularJS applications.
- The ng-app directive defines the application, the ng-controller directive defines the controller.

AngularJS: Application

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>
```

AngularJS: Application

How AngularJS Integrates with HTML:

- 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: Application

Example:

```
<body>
<p>Try to change the names.</p>
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br><br>
Full Name: {{firstName + " " + lastName}}

</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";
    $scope.lastName= "Doe";
});
</script>
```

AngularJS: 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: Directives

- ng-app directive:
  - The ng-app directive starts an AngularJS Application.
  - It defines the root element.
  - It automatically initializes or bootstraps the application when the web page containing AngularJS Application is loaded.
  - It is also used to load various AngularJS modules in AngularJS Application.

  - In the following example, we define a default AngularJS application using ng-app attribute of a <div> element.

```
<div ng-app = "">
   ...
</div>
```

AngularJS: Directives

- ng-init directive
  - The ng-init directive initializes an AngularJS Application data.
  - It is used to assign values to the variables.

AngularJS: Directives

- ng-model directive
  - The ng-model directive defines the model/variable to be used in AngularJS Application.

  - In the following example, we define a model named name.

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

AngularJS: Directives

- ng-repeat directive
  - The ng-repeat directive repeats HTML elements for each item in a collection.

  - In the following example, we iterate over the array of countries.

```
<div ng-app="" ng-init="names=[
{name:'J',country:'Norway'},
{name:‘K',country:'Sweden'},
{name:‘L',country:'Denmark'}]">

<p>Looping with objects:</p>
<ul>
  <li ng-repeat="x in names">
  {{ x.name + ', ' + x.country }}</li>
</ul>

</div>
```

- ## AngularJS Scope

```
<div ng-app="myApp" ng-controller="myCtrl">

<h1>{{carname}}</h1>

</div>

<script>
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {
  $scope.carname = "Volvo";
});
</script>
```

AngularJS Scope

Understanding the Scope
If we consider an AngularJS application to consist of:

- View, which is the HTML.
- Model, which is the data available for the current view.
- Controller, which is the JavaScript function that makes/changes/removes/controls the data.
- Then the scope is the Model.

The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

- AngularJS Scope

```
<div ng-app="myApp" ng-controller="myCtrl">

<input ng-model="name">

<h1>My name is {{name}}</h1>

</div>

<script>
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {
  $scope.name = “Amit Dev";
});
</script>
```

AngularJS: Controller

- 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: Controller

```
<script>
   function studentController($scope) {
      $scope.student = {
         firstName: "Mahesh",
         lastName: "Parashar",

         fullName: function() {
            var studentObject;
            studentObject = $scope.student;
            return studentObject.firstName + " " + studentObject.lastName;
         }
      };
   }
</script>
```

AngularJS: Controller

- The studentController is defined as a JavaScript object with $scope as an argument.

- The $scope refers to application which uses the studentController object.

- The $scope.student is a property of studentController object.

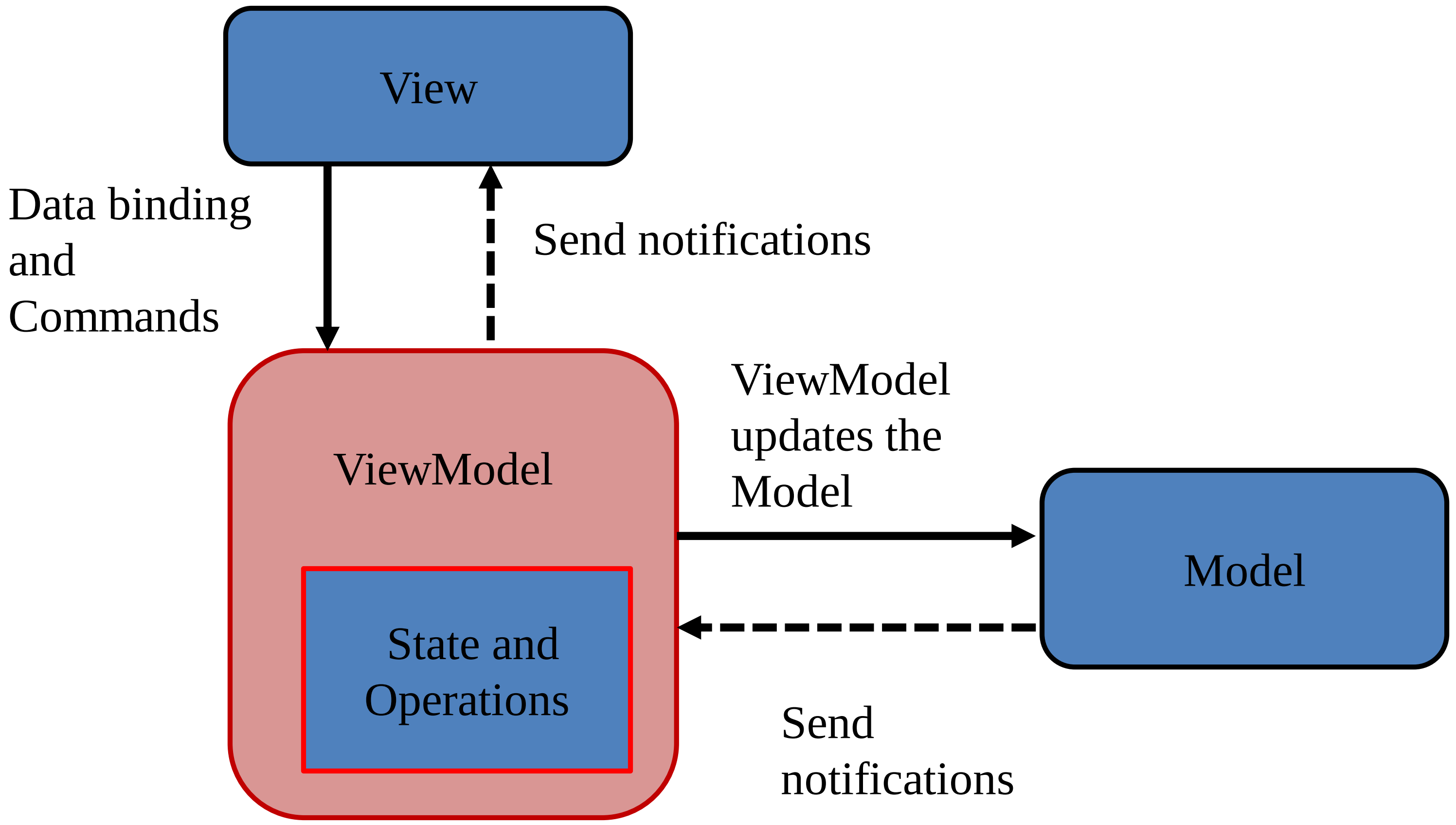
- The firstName and the lastName are two properties of $scope.student object. We pass the default values to them.

- The property fullName is the function of $scope.student object, which returns the combined name.

- In the fullName function, we get the student object and then return the combined name.

The Model-View-ViewModel (MVVM) Pattern

- MVVM facilitates a separation of development of the user interface from development of the business logic or back-end logic (the data model).
- It is derived from the MVC pattern.



The Model-View-ViewModel (MVVM) Pattern

- The view *knows about* the view model, and the view model *knows about* the model, but the model is *unaware* of the view model, and the view model is *unaware* of the view.

- The view model isolates the view from the model classes to evolve independently of the view.

- **Model**
  - The model is an implementation of the application's domain model that includes a data model along with business and validation logic.

- **View**
  - The view is responsible for defining the structure, layout and appearance of what the user sees on the screen.

  - A view gets data from its view model through data bindings, or invoking methods on the view model.

The Model-View-ViewModel (MVVM) Pattern

- **ViewModel**
  - The view model acts as an intermediary between the view and the model, and is responsible for handling the view logic.
  - Typically, the view model interacts with the model by invoking methods in the model classes.
  - The view model then provides data from the model in a form that the view can easily use.
  - The view model retrieves data from the model and then makes the data available to the view, and may reformat the data in some way that makes it simpler for the view to handle.
  - For example, when a user clicks a button in the UI, that action can trigger a command in the view model.

• Implementing MVVM in Angular

Angular supports the principles behind MVVM design pattern in the following approach.

- **Model** — The model is implemented as an Angular service (Factory or Service).

- **View** — The View is implemented using Angular ‘template’ (HTML with data bindings) that is rendered into the Angular view directive (ng-view / ui-view).

  - The View is bound to the ViewModel so that when a property is changed in the ViewModel, it is instantly reflected in the View.

  - Data bindings can work both ways in special cases like HTML forms, where a user can manipulate a property directly.

• Implementing MVVM in Angular

Angular supports the principles behind MVVM design pattern in the following approach.

- **ViewModel** — The ViewModel is implemented as an Angular controller.

  - The ‘ngController’ directive is used to specify a controller for a view. It can expose properties and methods to the view.

  - It can also be attached to the DOM by declaring it in a route definition via the ‘$route’ service.

• Benefits of MVVM

Using MVVM pattern within Angular applications offer the following benefits:

- During the development process, developers and designers can work more independently and concurrently on their components.
- The designers can concentrate on the view, while the developers can work on view model and model components.
- The developers can create unit tests for the view model and model without using the view.
- It is easy to redesign the user interface (UI) of the application without having any impact on the view model and model.

• AngularJS Filters

AngularJS provides filters to transform data:

- 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.

• AngularJS Filters

Adding Filters to Expressions:

Filters can be added to expressions by using the pipe character |, followed by a filter.

The uppercase filter format strings to upper case:

Example:

```
<div ng-app="myApp" ng-controller="personCtrl">

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

</div>
```

• AngularJS Filters

Adding Filters to Expressions:

The lowercase filter format strings to lower case:

Example:

```
<div ng-app="myApp" ng-controller="personCtrl">

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

</div>
```

• AngularJS Filters

Adding Filters to Directives

Filters are added to directives, like ng-repeat, by using the pipe character |, followed by a filter:

Example:
The orderBy filter sorts an array:

```
<div ng-app="myApp" ng-controller="namesCtrl">

<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>
```

• AngularJS Filters

The currency Filter

The currency filter formats a number as currency:

Example

```
<div ng-app="myApp" ng-controller="costCtrl">

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

</div>
```

AngularJS Filters

The filter Filter

The filter filter selects a subset of an array.

The filter filter can only be used on arrays, and it returns an array containing only the matching items.

Example:
Return the names that contains the letter "i":

```
<div ng-app="myApp" ng-controller="namesCtrl">

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

</div>
```