AngularJS

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 1.4.8.
- AngularJS is a structural framework for dynamic web apps.
- It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly.
- Angular's data binding and dependency injection eliminate much of the code you currently have to write.

Features

- AngularJS is a powerful JavaScript based development framework to create RICH Internet Application(RIA).

- AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC(Model View Controller) way.

- Application written in AngularJS is 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 and high performance web application while keeping them as easy-to-maintain.

Core 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 that are bound to a particular scope.
- **Services** − AngularJS come with several built-in services for example $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.

Core Features

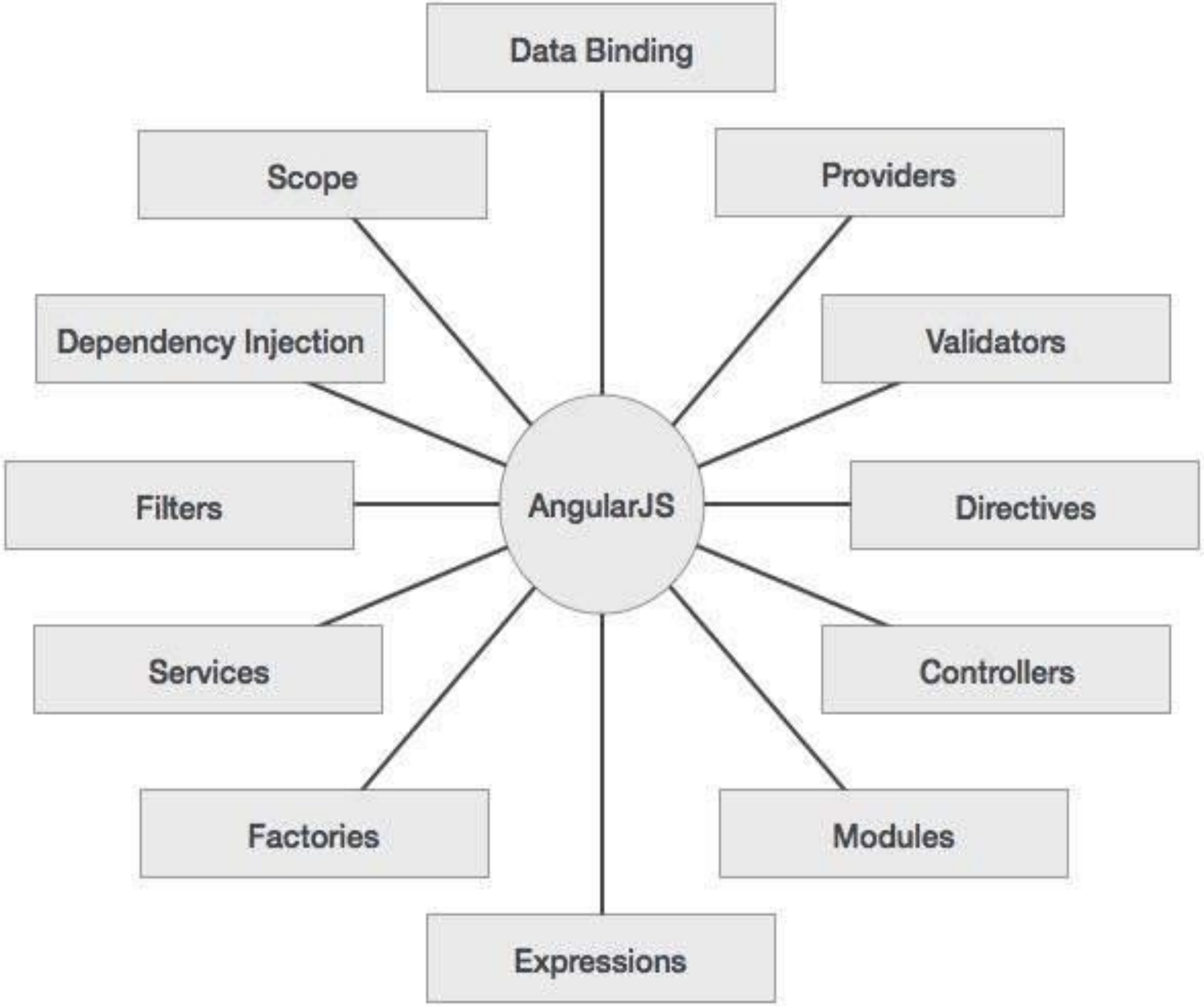
- **Templates** − These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".

- **Routing** − It is concept of switching views.

- **Model View Whatever** − MVC 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 you 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.

Concepts


Data Binding
Scope
Providers
Dependency Injection
Validators
Filters
AngularJS
Directives
Services
Controllers
Factories
Modules
Expressions

Advantages of AngularJS

- AngularJS provides capability to create Single Page Application in a very clean and maintainable way.

- AngularJS provides data binding capability to HTML thus giving 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, developer write less code and get more functionality.

- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

Disadvantages of AngularJS

- **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 your application user disables JavaScript then user will just see the basic page and nothing more.

The AngularJS Components

- **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

- AngularJS directives are HTML attributes with an **ng** prefix.
- The **ng-init** directive initialize AngularJS application variables.

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

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

  </div>
  ```
- You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.

AngularJS Expressions

- AngularJS expressions are written inside double braces:
  - **{{ expression }}**.
- AngularJS will "output" data exactly where the expression is written

-
  ```
  <div ng-app="">
    <p>My first expression: {{ 5 + 5 }}</p>
  </div>
  ```
- AngularJS expressions bind AngularJS data to HTML the same way as the **ng-bind** directive.

AngularJS Applications

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

-
  ```
  <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>
  Full Name: {{firstName + " " + lastName}}
  </div>

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

AngularJS Numbers

-
  ```
  <div ng-app="" ng-init="quantity=1;cost=5">

  <p>Total in dollar: {{ quantity * cost }}</p>

  </div>
  ```

AngularJS Objects

-
  ```
  <div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

  <p>The name is {{ person.lastName }}</p>

  </div>
  ```

AngularJS Arrays

- <div ng-app="" ng-init="points=[1,15,19,2,40]">

  <p>The third result is {{ points[2] }}</p>

  </div>

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 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.

Creating a Module

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

  <script>

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

  </script>
  ```

Adding a Controller

```
<div ng-app="myApp" ng-controller="myCtrl">
            {{ firstName + " " + lastName }}
</div>

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

            app.controller("myCtrl", function($scope) {
             $scope.firstName = "John";
             $scope.lastName = "Doe";
            });

</script>
```

Adding a Directive

- AngularJS lets you extend HTML with new attributes called **Directives**.
- AngularJS has a set of built-in directives which you can use to add functionality to your application.

-
  ```
  <div ng-app="myApp" custom-directive></div>

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

  app.directive("custom-directive", function() {
      return {
          template : “This is made from custom directive"
      };
  });
  </script>
  ```

Modules and Controllers in Files

-
  ```
  <!DOCTYPE html>
  <html>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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>
  ```

myApp.js

- var app = angular.module(**"myApp"**, []);

myCtrl.js

-
  ```
  app.controller("myCtrl", function($scope) {
      $scope.firstName = "John";
      $scope.lastName= "Doe";
  });
  ```

AngularJS Directives

- 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.

Repeating HTML Elements

-
  ```
  <div ng-app="" ng-init="names=[‘Aman’,’Ajay’,’Anil’]">
   <ul>
    <li ng-repeat="x in names">
     {{ x }}
    </li>
   </ul>
  </div>
  ```

Example2

```
<div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">

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

</div>
```

AngularJS is perfect for database CRUD (Create Read Update Delete) applications.
Just imagine if these objects were records from a database.

Restrictions

-
  ```
  var app = angular.module("myApp", []);
  app.directive("w3TestDirective", function() {
      return {
          restrict : "A",
          template : "<h1>Made by a directive!</h1>"
      };
  });
  ```

- The legal restrict values are:
  - E for Element name
  - A for Attribute
  - C for Class
  - M for Comment
  - By default the value is EA, meaning that both Element names and attribute names can invoke the directive.

The ng-model Directive

- Two-Way Binding

- The binding goes both ways. If the user changes the value inside the input field, the AngularJS property will also change it's value:

-
  ```
  <div ng-app="myApp" ng-controller="myCtrl">
      Name: <input ng-model="name">
      <h1>You entered: {{name}}</h1>
  </div>
  ```

Validate User Input

-
  ```
  <form ng-app="" name="myForm">
     Email:
     <input type="email" name="myAddress" ng-model="text">
     <span ng-show="myForm.myAddress.$error.email">Not a valid e-mail address</span>
  </form>
  ```

CSS Classes

```
<style>

input.ng-invalid {
    background-color: lightblue;
}

</style>
<body>

<form ng-app="" name="myForm">
    Enter your name:
    <input name="myAddress" ng-model="text" required>
</form>
```

ng-model directive

- The ng-model directive adds/removes the following classes, according to the status of the form field:

- ng-empty
- ng-not-empty
- ng-touched
- ng-untouched
- ng-valid
- ng-invalid
- ng-dirty
- ng-pending
- ng-pristine

AngularJS Controllers

- AngularJS controllers **control the data** of AngularJS applications.
- AngularJS controllers are regular **JavaScript Objects**.
- AngularJS will invoke the controller with a **$scope** object.
- In AngularJS, $scope is the application object (the owner of application variables and functions).
- The controller creates two properties (variables) in the scope (**firstName** and **lastName**).
- The **ng-model** directives bind the input fields to the controller properties (firstName and lastName).

Controller Methods

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

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

  </div>

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

AngularJS Filters

- Filters can be added in AngularJS to format 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.

Adding Filters to Expressions

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

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

  </div>
  ```

Adding Filters to Directives

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

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

  </div>
  ```

The currency Filter

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

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

  </div>
  ```

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.

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

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

  </div>
  ```

Filter an Array Based on User Input

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

  <p><input type="text" ng-model="test"></p>

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

  </div>
  ```

Sort an Array Based on User Input

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

<table border="1" width="100%">
  <tr>
    <th ng-click="orderByMe('name')">Name</th>
    <th ng-click="orderByMe('country')">Country</th>
  </tr>
  <tr ng-repeat="x in names | orderBy:myOrderBy">
    <td>{{x.name}}</td>
    <td>{{x.country}}</td>
  </tr>
</table>

</div>

<script>
angular.module('myApp', []).controller('namesCtrl', function($scope) {
  $scope.names = [
    {name:'Jani',country:'Norway'},
    {name:'Carl',country:'Sweden'},
    {name:'Margareth',country:'England'},
    {name:'Hege',country:'Norway'},
    {name:'Joe',country:'Denmark'},
    {name:'Gustav',country:'Sweden'},
    {name:'Birgit',country:'Denmark'},
    {name:'Mary',country:'England'},
    {name:'Kai',country:'Norway'}
  ];
  $scope.orderByMe = function(x) {
    $scope.myOrderBy = x;
  }
});
</script>
```

Custom Filters

-
  ```
  <ul ng-app="myApp" ng-controller="namesCtrl">
      <li ng-repeat="x in names">
          {{x | myFormat}}
      </li>
  </ul>

  <script>
  ```

-
  ```
  var app = angular.module('myApp', []);
  app.filter('myFormat', function() {
      return function(x) {
          var i, c, txt = "";
          x = x.split("")
          for (i = 0; i < x.length; i++) {
              c = x[i];
              if (i % 2 == 0) {
                  c = c.toUpperCase();
              }
              txt += c;
          }
          return txt;
      };
  });
  app.controller('namesCtrl', function($scope) {
      $scope.names = ['Jani', 'Carl', 'Margareth', 'Hege', 'Joe', 'Gustav', 'Birgit', 'Mary','Kai'];
  });
  ```

-
  ```
  </script>
  ```

AngularJS Services

- What is a Service?
    - a service is a function, or object, that is available for, and limited to, your AngularJS application.
    - Example
        - $location
        - $http
        - $timeout
        - $interval

$location Service

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $location) {
    $scope.myUrl = $location.absUrl();
});
```

$http Service

- var app = angular.module('myApp', []);

-
  ```
  app.controller('myCtrl', function($scope, $http) {
      $http.get("welcome.txt").then(function (response) {
          $scope.myWelcome = response.data;
      });
  });
  ```

The $timeout Service

-
  ```
  var app = angular.module('myApp', []);
  ```

-
  ```
  app.controller('myCtrl', function($scope, $timeout) {
      $scope.myHeader = "Hello World!";
      $timeout(function () {
          $scope.myHeader = "How are you today?";
      }, 2000);
  });
  ```

The $interval Service

-
  ```
  var app = angular.module('myApp', []);
  ```

-
  ```
  app.controller('myCtrl', function($scope, $interval) {
    $scope.theTime = new Date().toLocaleTimeString();
    $interval(function () {
      $scope.theTime = new Date().toLocaleTimeString();
    }, 1000);
  });
  ```

Create Your Own Service

-
  ```
  app.service('hexafy', function() {
      this.myFunc = function (x) {
          return x.toString(16);
      }
  });
  ```

-
  ```
  app.controller('myCtrl', function($scope, hexafy) {
     $scope.hex = hexafy.myFunc(250);
  });
  ```

- Thank you!