AngularJS

Introduction

Ardhika

Topics

- Introduction
- Controllers
- Use of Directives
- Filters
- Validations

What is AngularJS

- Open source Javascript framework licensed (MIT) by Google
- Pure client side framework for dynamic Web Apps
- HTML templates used
- No browser refreshing entire page
- No DOM refresh
- Tracks user action, browser events and Model (data) changes and updates the view - MVC

What is MVC?

- MVC consists of
  - Model - The Data
  - View - The HTML page seen in the browser
  - Controller - The Magic code
- Controller responds to events and changes data which are picked up by Angular to update the view

Parts of AngularJS

- HTML Templates with Directives
  - Directives - new HTML syntax for elements, Attributes, styles and comments
- Application Logic in Javascript
  - Controllers - event processing and model updates
  - Dependency Injection of Javascript modules
- Data Binding - Two way
  - View to Model
  - Model to View

Hello Angular

- Demo

Hello Angular - Code

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello Angular</title>
  </head>
  <body ng-app>
    <input type="text" ng-model="who"><br/>
    Hello, {{who}}!
  </body>
  <script src="../angular/angular.js">
  </script>
</html>
```

Hello Angular Parts

- Insert the AngularJS script file

  `<script src="../angular/angular.js">`
- Directives
  - ng-app : To indicate the Angular app
  - ng-model : Specify a model (data) with a name (who)
- Template
  - {{who}} : handle-bar style replace the model(data) here
- No controller or application logic here
- Angular manages the 2 way data binding
  - From textbox to model variable “who”
  - And from variable to the template in the view

Hello Controller!

- Controller is a Javascript function which works with an injected object called “$scope”
- Controllers need to be defined on a Javascript module

```
var app = angular.module('myApp', []);
app.controller('HelloController', function($scope){
    $scope.who = "World";
    $scope.greetingText = "Hello, " + $scope.who + "!";
});
```

- myApp is the name of an angular module
- A Controller with a name HelloController is attached to myApp
- The controller is a function which takes the $scope as parameter

Page using the controller

```
   <body ng-app="myApp">
   <div ng-controller="HelloController">
      <input type="text" ng-model="who">
      <p>
         {{greetingText}}
      </p>
   </div>
   <script src="../angular/angular.js"></script>
   <script>
  //app module and controller here
   </script>
```

- myApp is set to ng-app
- Directive `ng-controller` to attach the controller by name
- Controller set the greetingText and angular sets it to the view

Events

- When we type a name it does not reflect in greetingText
- Let us put a button next to the textbook

```
<button type="button" ng-click="greet()">Greet</button>
```

- Directive ng-click is to attach a javascript function in the controller to the click event of a button

```
 app.controller('HelloController', function($scope){
     $scope.who = "World";
     $scope.greetingText = "Hello, " + $scope.who + "!";
     $scope.greet = function () {
        $scope.greetingText = "Hello, " + $scope.who + "!";
     };
 });
```

- Attach the greet() to scope in the controller

Exercise

- Set #1

AngularJS

Angular Given Directives

ng-change

- Used to act on the changing of a view component
- Used as attribute of an element and a function is attached to handle the change

```
<input type="text" ng-model="celcius"
            ng-change="convert()">
```

- Convert is defined inside of a controller

```
$scope.convert = function() {
  //handle the conversion here
}
```

- Alternately we can ask the scope to watch for changes in the controller and execute a function

```
convert = function(){
//Handle conversion
};
$scope.$watch("celcius", convert);
```

ng-submit

- Directive attached to a form
- Function in scope needs to be attached

```
<form ng-submit="computSomething()">
<!— form elements —>
</form>
```

- In controller

```
$scope.computeSomething = function() {
//What to do?
};
```

ng-repeat

- Directive to repeat (iterate) items in a collection

```
app.controller("RepeatController", function($scope){
    $scope.courses = [
        {name:"Maths", code:"MAT101"},
        {name:"Physics", code:"PHY201"},
        {name:"Chemistry", code:"CHE101"}
    ];
});
```

- courses is an array of course objects
- course object has attributes - name & code
- Display this using a `<li>` tag

ng-repeat

```
<ul>
    <li ng-repeat="course in courses">
       {{course.name}}({{course.code}})
    </li>
</ul>
```

- ng-repeat takes an expression "*var in collection*" as a value
- ng-repeat is used to display
  - rows of a table
  - items in a menu
  - List of checkboxes/radio buttons/drop down values

ng-show & ng-hide

- Used as attributes of an element/tag
- These directives show/hide contents of an element

`ng-show="expression"`

- Show the contents when expression is true hide it otherwise

`ng-hide="expression"`

- Hide the content when expression is true show otherwise
- Both do the opposite of the same thing
- These directives work by altering the CSS proper display
- ng-show makes display:block
- ng-hide makes display:none

ng-if

- Similar to ng-show/ng-hide controls visibility
- But adds or removes the element to the DOM
- ng-if is used as attribute to a tag
- Adds the tag to the DOM is expression is true

```
<p ng-if='showPara'>
 This paragraph is added to the DOM
   if showPara is true and removed otherwise
</p>
```

ng-show & ng-hide

```
<div ng-controller="ShowController">
   <p ng-show='showPara'>
 Here, it’s ng-show and ng-hide that do our work. They provide
equivalent but inverse functionality for showing and hiding
based on the expression you pass to them. That is, ng-show will
show its element when its expression is true and hide it when
false. The ng-hide hides when true and shows when false. You
should use whichever makes more sense to express your
intention. These directives work by setting the element styles
to display:block to show and display:none to hide as
appropriate.
  </p>
  <button ng-click=‘toggleVisibility()'>{{caption}}
  </button>
</div>
```

ng-show & ng-hide

```
app.controller("ShowController", function($scope){
   $scope.showPara = false;
   $scope.caption = "Show the secret"

   $scope.toggleVisibility = function() {
      $scope.showPara = !$scope.showPara;
      $scope.caption = ($scope.showPara)?
               "Hide the secret" : "Show the secret";
   };
});
```

ng-class

- Choose or mix n match style classes for elements
- For applying classes you can store the class name in a model variable and set it to the class attribute
  ```
  <p class="{{model-var}}"> ….</p>
  ```
- Here, the model-var can be set in the controller code
- Or use the versatile ng-class directive for a more complex scenario
- ng-class is used as an attribute
  ```
  <p ng-class="model-var">…</p>
  ```
- Note: Expression syntax ({{…}}) is not used here
- model-var is any string value which represents a class in the css file

Ardhika

ng-class scenarios

- List of style classes

  `ng-class="[style1, style2, style3…]"`
- style1, style2 etc are model variables in scope which contains strings (nothing but class names in css)

- Or object style

  `ng-class="{style1:bmv1, style2:bmv2…}"`
- bmv1, bmv2 are model variables of type boolean
- style1 will be applied if bmv1 is true and so on
- Multiple styles can be applied

- List can have object style as element too

ng-class

- Demo string, list and object styles

ng-disabled

- Directive (as an attribute) to disable(enable) an element
  `ng-disabled="expression"`
- Element which has this attribute will get disabled when expression is true (enabled otherwise)

```
<input type="button" value="Ping" ng-disabled="pinged"
            ng-click="pingClicked()">
<input type="button" value="Pong" ng-disabled="ponged"
            ng-click="pongClicked()">
```

- in Controller

```
        $scope.pingClicked = function() {
          $scope.pinged = true;
          $scope.ponged = false;
        };
```

ng-options

- Used for select / options - drop downs/list box
- Can use ng-repeat to add options
- ng-options is an attribute to select tag with versatile syntax

- Sample Data

```
$scope.courses = [
  {name:"Maths", code:"MAT101"},
  {name:"Physics", code:"PHY201"},
  {name:"Chemistry", code:"CHE201"},
  {name:"Zoology", code:"ZOO201"}
];
```

ng-options

- Selecting a course Object

```
<select ng-model="selctedCourse1"
      ng-options="course.name for course in courses">
   <option value="">Select a Course</option>
</select>
Selected Course1 : {{selectedCourse1}}<br/>
```

- When you select a course the model variable selectedCourse1 will have an object

Selected Course1 : {"name":"Maths","code":"MAT101"}

ng-options

- Selecting the code of a course

```
<select ng-model="selctedCourse2"
ng-options="course.code as course.name for course in
courses">
    <option value="">Select another Course</option>
</select>
Selected Course2 : {{selctedCourse2}}<br/>
```

- When you select a course the model variable selctedCourse2 will have the course code

```
Selected Course2 : MAT101
```

ng-include

- How to include a html fragment in another page
  `ng-include="'child.html'"`
- This will come as an attribute to a container tag like div, span, p etc.
- Note the single quote inside the double quote

Exercise

- Set #2

AngularJS

Filters & Validations

Filters

- Used to transform data
- Used along with scope data inside expressions
- Can be used in Javascript code to transform data
- Applying a generic filter

```
<input type="text" ng-model="searchText">
<ul>
   <li ng-repeat="course in courses | filter:searchText">
          {{course.name}} - {{course.code}}
   </li>
</ul>
```

- Here what ever is typed in the textbox will be used for searching

Filters

- How to search on a specific field?
- To search by name

```
<input type="text" ng-model="searchObject.name">
<ul>
   <li ng-repeat="course in courses | filter:searchObject">
       {{course.name}} - {{course.code}}
    </li>
</ul>
```

- Same way to search in code alone use searchObject.code
- To search in any field use searchObject.$

Special Filters - Currency

- Currency filter is used to add a symbol (using locale) and does rounding off

```
expression | currency : symbol : roundoff_decimal_digits


Amount : {{amount | currency :"$": 2}}
```

- If symbol is omitted default symbol is used
- If digits are omitted default number of decimal digits of the currency is used

Special Filters - Date

- Formats date with timezone

```
mydate | date : ″format String″ : ″Timezone″
```

- if format is omitted medium format (MMM d, y) will be used

```
Date : {{date | date : "dd/MM/yy @ hh:mm" : "+530" }}
```

- Shows the date and time in IST timezone

Special Filters

- lowercase / uppercase
  - used to convert case of text strings
  - `text | lowercase` or `text | uppercase`

- limitTo
  - used to restrict length of array, text and number(digits)
  - `data | limitTo : number_to_display : starting_from`

- number
  - display numbers and round of decimal digits
  - `data | number : decimal_to_digits_round_off`

Special Filters

- orderBy
  - Sorts an array using a predicate
  - `ng-repeat="course in courses | orderBy:'name'"`
  - To sort in descending order
  - `ng-repeat="course in courses | orderBy:'-name'"`

- json
  - Used for debugging, Print object content as JSON
  - Used inside <pre> tag
  - Can specify tab stop for indentation
  - `<pre> {{courses | json : 4}}</pre>`

Form validations

- Special directives for validating the input controls inside a form
- Make sure to give a name to form and input controls
- Model variables are used for data binding and form/input names are used in validations

```
<form name="courseForm">
  Course name : <input type="text" ng-model="name"
                   name="courseName" ng-required="true"><br/>
 Course code : <input type="text" ng-model="code"
                   name="courseCode" ng-minlength="6"><br/>
</form>
```

Form validations

- Validation Directives
  - ng-required - sets the required attribute, field entry is mandatory
  - ng-minlength - specify minimum chars to input
  - mg-maxlength - specify maximum chars to input
  - ng-pattern - specify a regular expression to validate
- To check whether a field has invalid value, this expression must be true
  - `<formName>.<fieldName>.$invalid`
- This check can be used to paint the control in a different border color or background to show it contains invalid value

Form validations - Field status

- Apart from $invalid there are 2 more statuses you can check
- $pristine - user has not touched the field (it may have invalid value by default)
- $dirty - user has touched the field
- Use $dirty in conjunction with $invalid to see that the user has entered something and is invalid

- There can be more than one validation directive on a control
- How to check the individual errors and show appropriate message?
- Use $error object

Form validations

- To check if the required field is missing a value, this expression must be true
  - *<formName>.<fieldName>.*$error.required
- Same way minlength, maxlength and pattern can be checked
- To show appropriate error message below controls

```
Course name : <input type="text" ng-model="name"
                  name="courseName" ng-required="true"><br/>
<span ng-show="courseForm.courseName.$error.required">
             Please enter course name</span><br/>
Course code : <input type="text" ng-model="code"
                  name="courseCode" ng-minlength="6"><br/>
<span ng-show="courseForm.courseCode.$error.minlength">
      Please enter course code with minimum 6 chars</span>
```

Exercise

- Set #3

Thank You!

Ardhika

Bala Sundarasamy
bala@ardhika.com