AngularJS Introduction

Angular Concepts and Terminology

Template : HTML with additional markup used to describe what should be displayed

Directive : Allows developer to extend HTML with own elements and attributes (reusable pieces)

Scope : Context where the model data is stored so that templates and controllers can access

Compiler : Processes the template to generate HTML for the browser

Data Binding : Syncing of the data between the Scope and the HTML (two ways)

Dependency Injection : Fetching and setting up all the functionality needed by a component

Module : A container for all the parts of an application

Service : A way of packaging functionality to make it available to any view

Angular Example

```
<!doctype html>
<html ng-app>
<head>
<script src="./angular.min.js"></script>
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name here">
<h1>Hello {{yourName}}!</h1>
</div>
</body>
</html>
```

Angular Bootstrap

```
<!doctype html>
<html ng-app>
<head>
<script src="./angular.min.js"></script>
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name here">
<h1>Hello {{yourName}}!</h1>
</div>
</body>
</html>
```

Angular Compiler Output

```
<!doctype html>
<html ng-app class="ng-scope">
```

```
<head>
<script src="./angular.min.js"></script>
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name here"
class="ng-pristine ng-untouched ng-valid">
<h1 class="ng-binding">Hello !</h1>
</div>
</body>
</html>
```

Changes to template HTML in red. Classes:

`ng-scope` - Angular attached a scope here.

`ng-binding` - Angular bound something here.

`ng-pristine`/`ng-dirty` - User interactions?

`ng-untouched`/`ng-touched` - Blur event?

`ng-valid`/`ng-invalid` - Valid value?

Two-way binding: Type 'D' character into `input` box

```
<!doctype html>
<html ng-app class="ng-scope">
<head>
<script src="./angular.min.js"></script>
</head>
<body>
<div>
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name here"
class="ng-valid ng-dirty ng-valid-parse ng-touched">
<h1 class="ng-binding">Hello D!</h1>
</div> </body> </html>
```

`angular.module` and `Controller`

In a JavaScript file:

```
angular.module("cs142App", [])
.controller('MyCntrl', function($scope) {
$scope.yourName = "";
$scope.greeting = "Hola";
});
```

```
<!doctype html>
<html ng-app="cs142App">
<head>
<script src="./angular.min.js"></script>
</head>
<body ng-controller="MyCntrl">
<div>
```

```
<label>Name:</label>
<input type="text" ng-model="yourName" placeholder="Enter a name here">
<h1>{{greeting}} {{yourName}}!</h1>
</div>
</body>
</html>
```

Templates, Scopes & Controllers

● Best practice: Each **template** component gets a new **scope** and is paired with a **controller**.

● **Expressions** in templates:
`{{foo + 2 * func()}}`
are evaluated in the context of the scope. Controller sets up scope:

```
$scope.foo = ... ;
$scope.func = function() { ... };
```

● Best practice: Keep expressions simple put complexity in controller

● Controllers make model data available to view template

Scope digest and watches

● Two-way binding works by watching when expressions in view template change and updating the corresponding part of the DOM.

● Angular add a **watch** for every variable or function in template expressions

● During the **digest** processing all watched expressions are compared to their previously known value and if different the template is reprocessed and the DOM update

○ Angular automatically runs digest after controller run, etc.

It is possible to:

Add your own watches: (`$scope.$watch(..)`) (e.g. caching in controller)

Trigger a digest cycle: (`$scope.$digest()`) (e.g. model updates in event)

A digression: `camelCase` vs `dash-case`

Word separator in multiword variable name

● Use dash: `active-buffer-entry`

● Capitalize first letter of each word: `activeBufferEntry`

Issue: HTML is case-insensitive so camelCase is a problem

AngularJS solution: You can use either, Angular will map them to the same thing.

Use dash in HTML and camelCase in JavaScript

Example: `ng-model` and `ngModel`

ngRepeat - Directive for looping in templates

● **ngRepeat** - Looping for DOM element creation (`tr`, `li`, `p`, etc.)

```
<ul>
<li ng-repeat="person in peopleArray">
```

```
<span>{{person.name}} nickname {{person.nickname}}</span>
</li>
</ul>
```

- Powerful but opaque syntax. From documentation:

```
<div ng-repeat="model in collection | orderBy: 'id' as
filtered_result track by model.id">
```

ngIf/ngShow - Conditional inclusion in DOM

- **ngIf** - Include in DOM if expression true (dialogs, modals, etc.)

```
<div class="center-box" ng-if="showTrialOverWarning">
{{buyProductAdmonishmentText}}
</div>
```

Note: will create scope/controllers when going to true, exit going to false

- **ngShow** - Like ngIf except uses visibility to hide/show DOM elements
  - Occupies space when hidden
  - Scope & controllers created at startup

ngClick/ngModel - Binding user input to scope

- **ngClick** - Run code in scope when element is clicked

```
<button ng-click="count = count + 1" ng-init="count=0">
Increment
</button>
<span> count: {{count}} </span>
```

- **ngModel -** Bind with `input`, `select`, `textarea` tags

```
<select name="singleSelect" ng-model="data.singleSelect">
<option value="option-1">Option 1</option>
<option value="option-2">Option 2</option>
</select>
```

ngHref & ngSrc

Sometimes need to use ng version of attributes:

- **a** tag

```
<a ng-href="{{linkHref}}">link1</a>
```

- `img` tag

```
<img ng-src="{{imageSrc}}" alt="Description" />
```

ngInclude - Fetches/compile external HTML fragment

- Include partial HTML template (Take angular expression of URL)

```
<div ng-include="'navBarHeader.html'"</div>
```

- CS142 uses for components

```
<div ng-include="'components/example/exampleTemplate.html'"
ng-controller="ExampleController"></div>
```

Directives

- Angular preferred method for building reusable components
  - Package together HTML template and Controller and extend templating language.
  - Ng prefixed items in templates are directives
- Directive can:
  - Be inserted by HTML compiler as:
    - attribute (`<div my-dir="foo">...</div>`)
    - element (`<my-dir arg1="foo">...</my-dir>`)
  - Specify the template and controller to use
  - Accept arguments from the template
  - Run as a child scope or isolated scope
- Powerful but with a complex interface

Example: `<example arg1="fooBar"></example>`

Directives are heavily used in Angular

```
<body layout="row" ng-controller="AppCtrl">
<md-sidenav layout="column" ... >
<md-toolbar ...>
...
</md-toolbar>
<md-list>
<md-item ng-repeat="item in menu">
<md-item-content layout="row" layout-align="start center">
<md-button aria-label="Add" ng-click="showAdd($event)">
</md-item-content>
</md-item>
<md-divider></md-divider>
<md-subheader>Management</md-subheader>
```

Services

- Used to provide code modules across view components
  - Example: shared JavaScript libraries
- Angular has many built-in services
  - Server communication (model fetching)

`$http, $resource, $xhrFactory`

  - Wrapping DOM access (used for testing mocks)

`$location, $window, $document, $timeout, $interval`

  - Useful JavaScript functionality

`$animate, $sce, $log`

  - Angular internal accesses

`$rootScope, $parse, $compile`