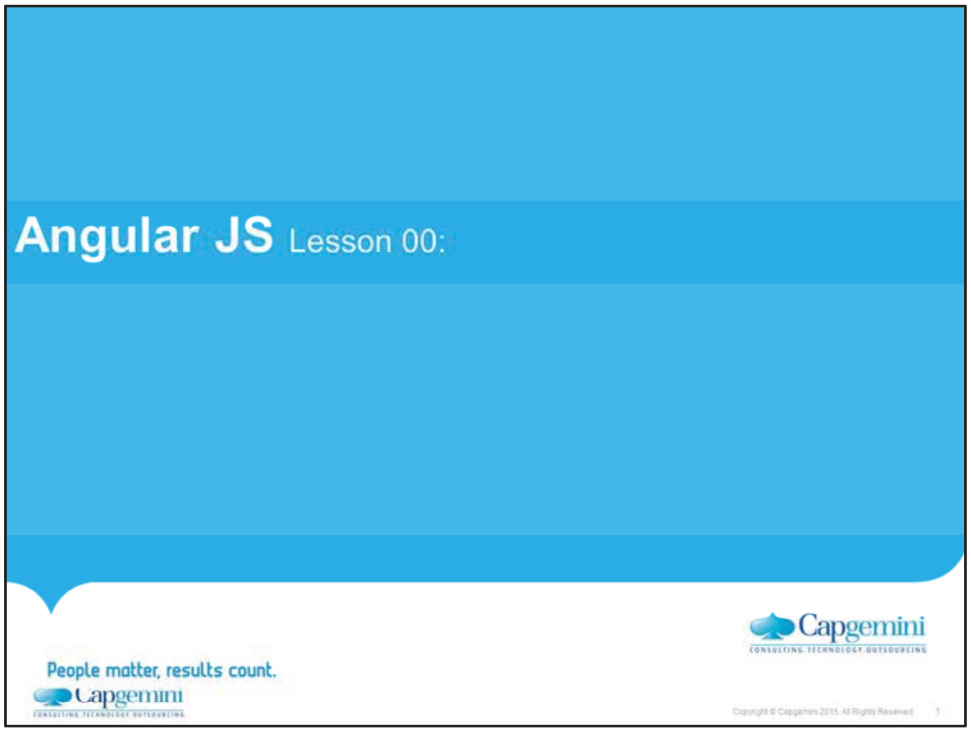
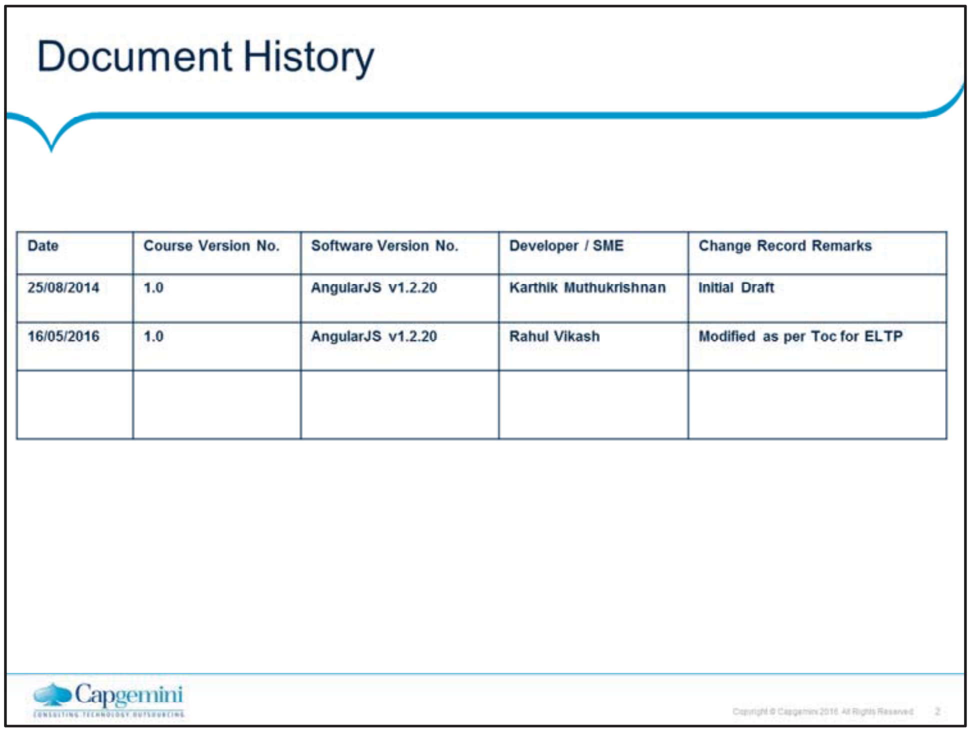


Angular JS Lesson 00:

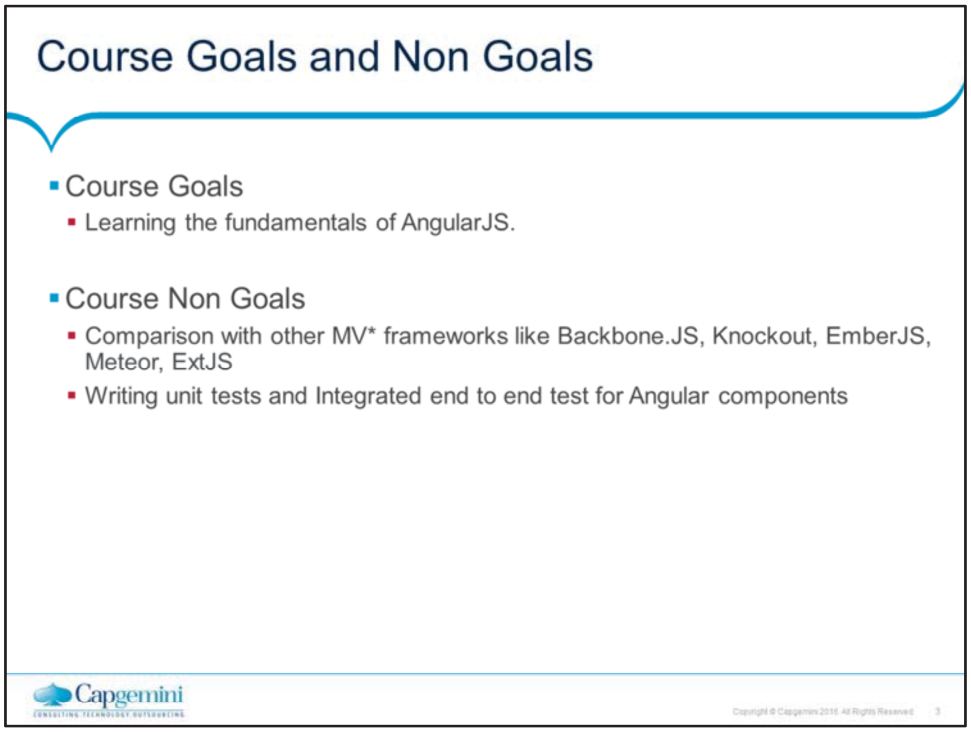
People matter, results count.



Document History

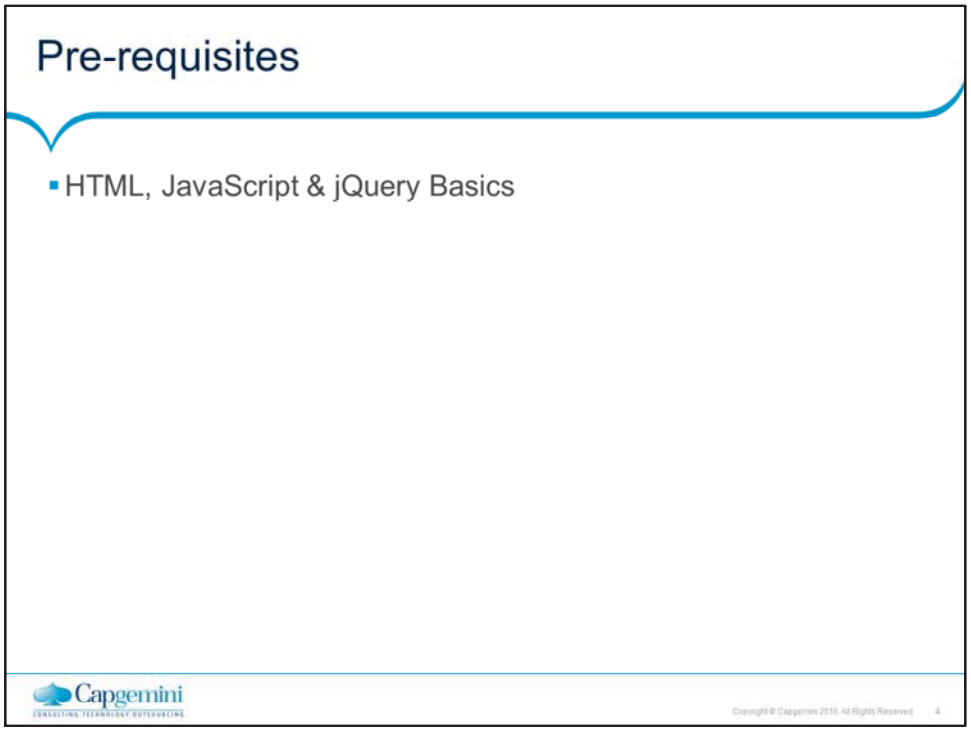
Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
25/08/2014	1.0	AngularJS v1.2.20	Karthik Muthukrishnan	Initial Draft
16/05/2016	1.0	AngularJS v1.2.20	Rahul Vikash	Modified as per Toc for ELTP

2



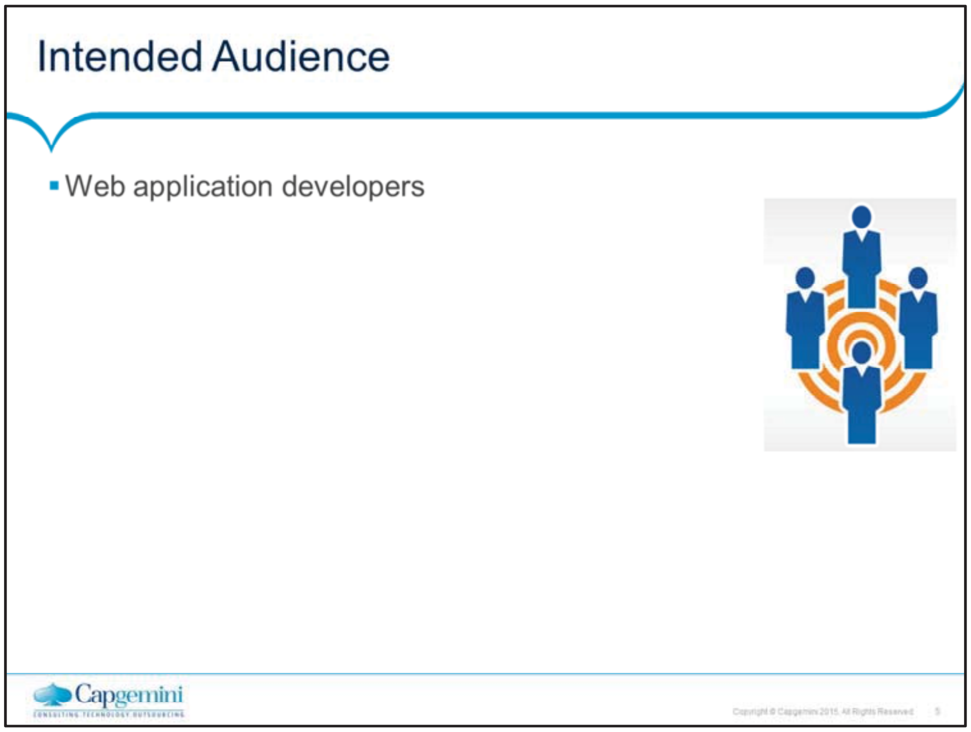
Course Goals and Non Goals

- Course Goals
  - Learning the fundamentals of AngularJS.
- Course Non Goals
  - Comparison with other MV* frameworks like Backbone.JS, Knockout, EmberJS, Meteor, ExtJS
  - Writing unit tests and Integrated end to end test for Angular components



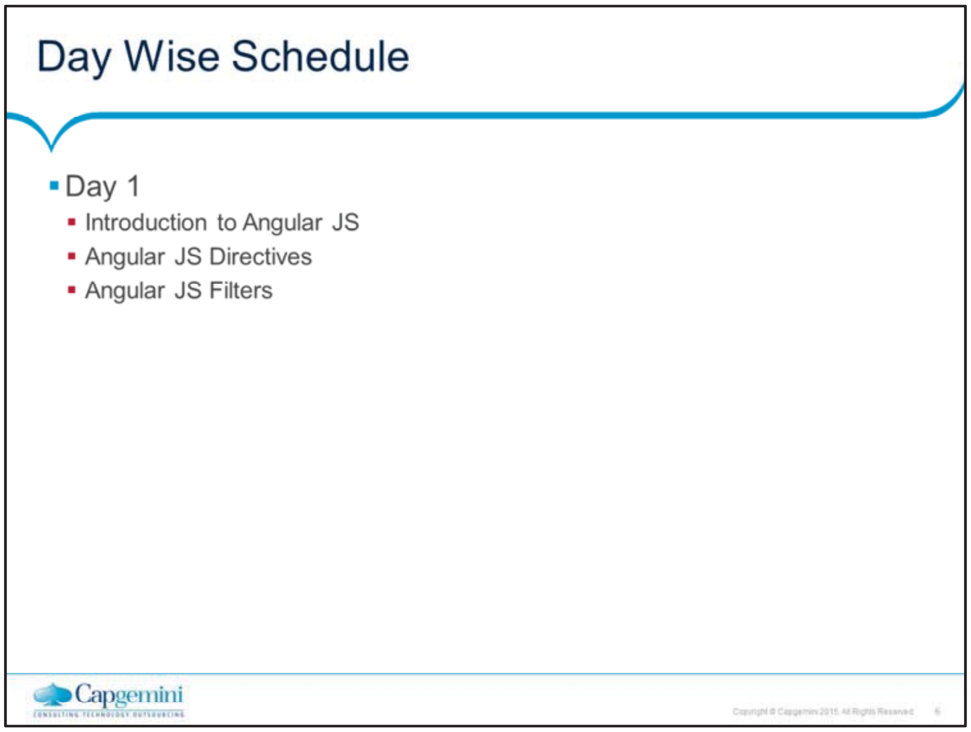
Pre-requisites

- HTML, JavaScript & jQuery Basics



Intended Audience

- Web application developers



Day Wise Schedule

- Day 1
  - Introduction to Angular JS
  - Angular JS Directives
  - Angular JS Filters

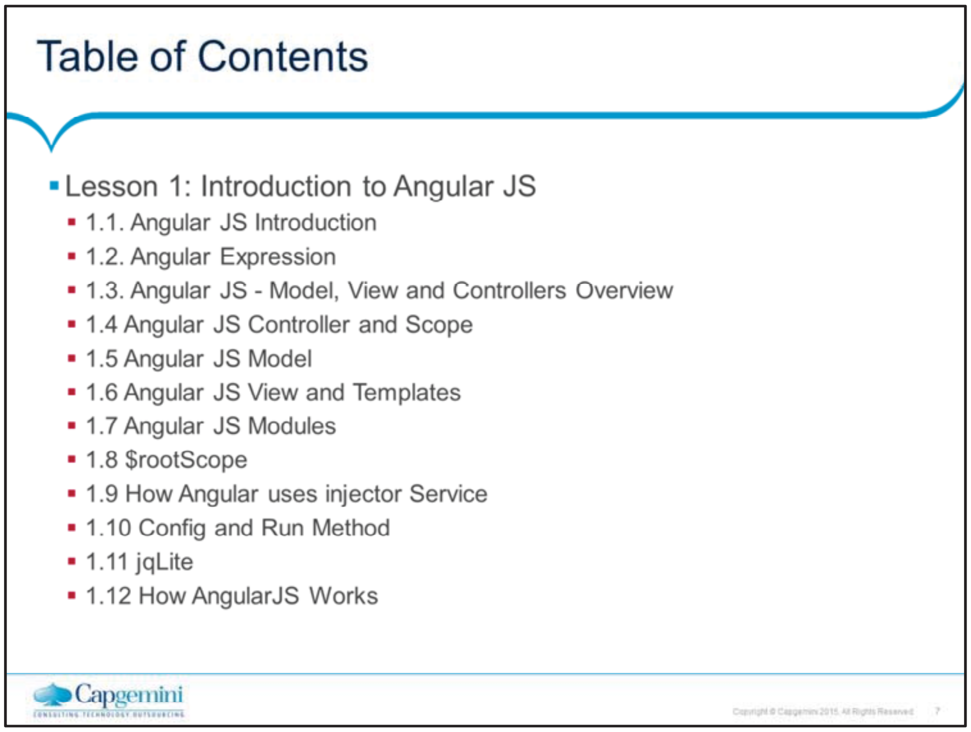


Table of Contents

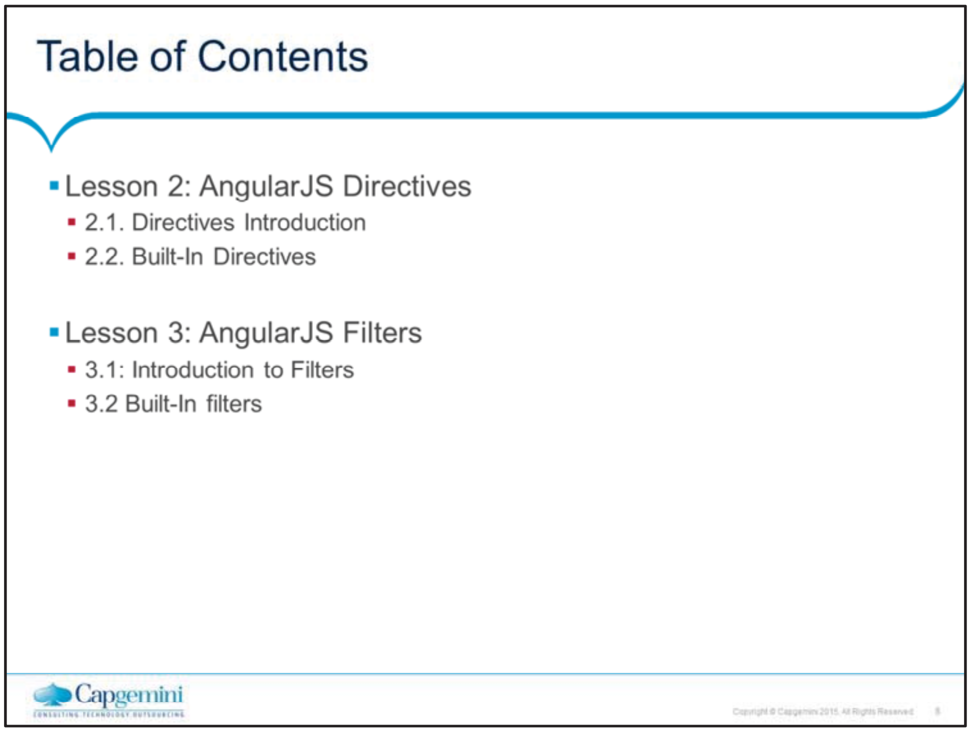
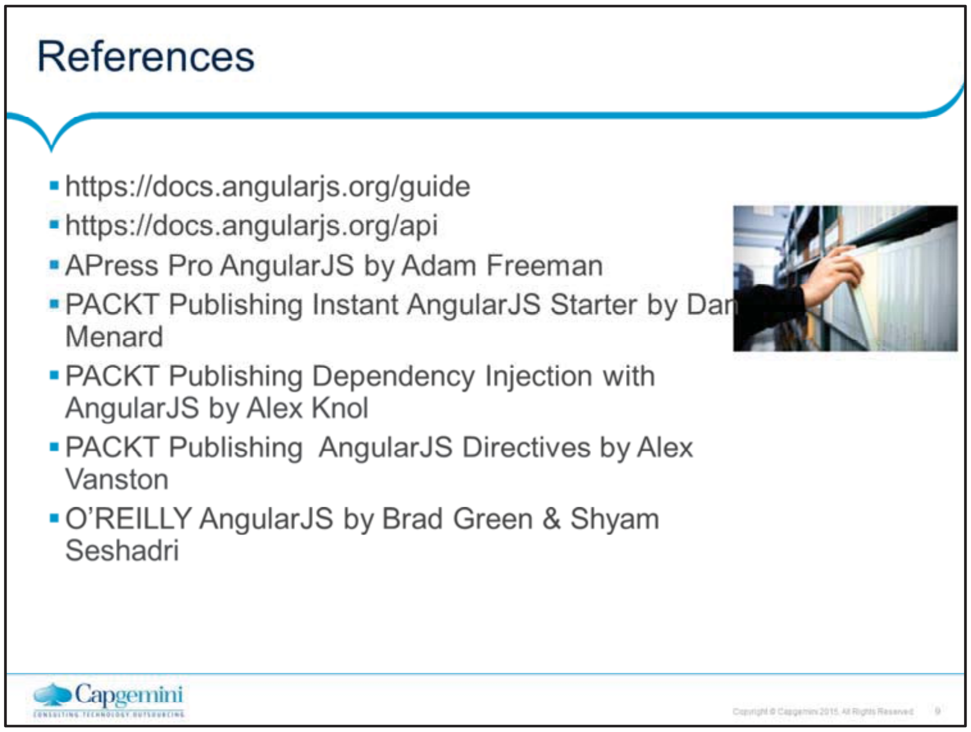
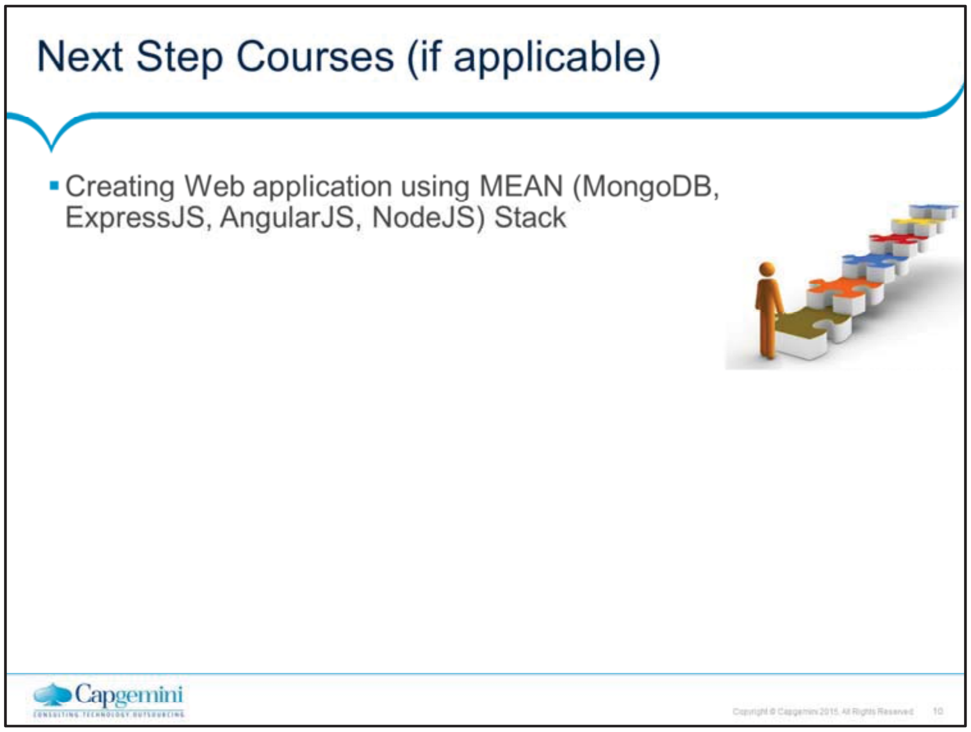


Table of Contents



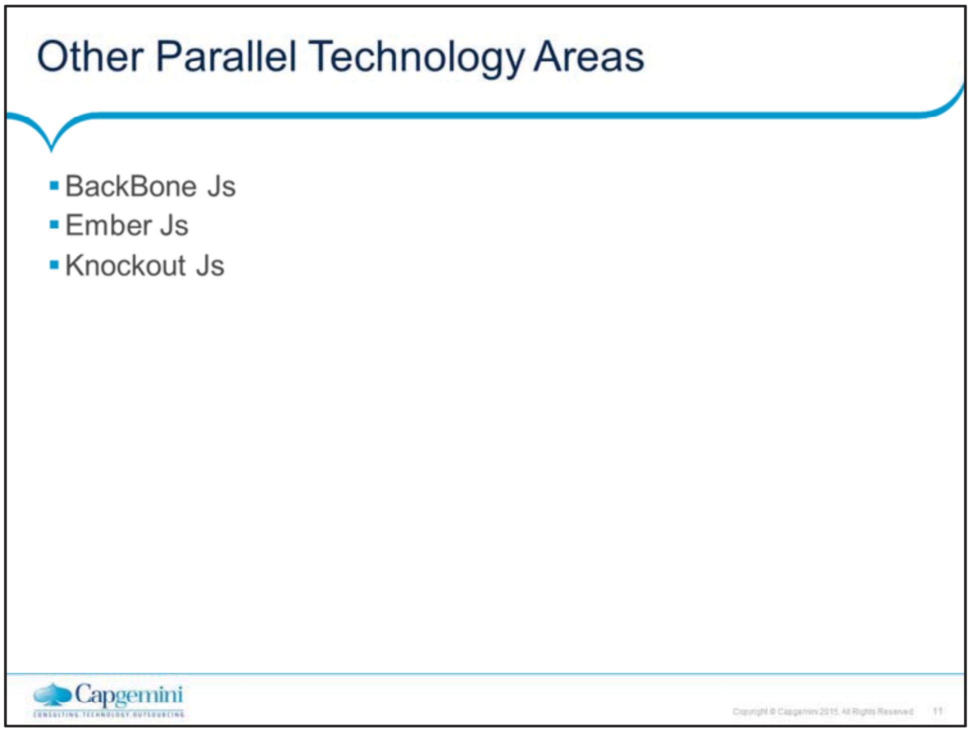
References

- https://docs.angularjs.org/guide
- https://docs.angularjs.org/api
- APress Pro AngularJS by Adam Freeman
- PACKT Publishing Instant AngularJS Starter by Dan Menard
- PACKT Publishing Dependency Injection with AngularJS by Alex Knol
- PACKT Publishing AngularJS Directives by Alex Vanston
- O'REILLY AngularJS by Brad Green & Shyam Seshadri



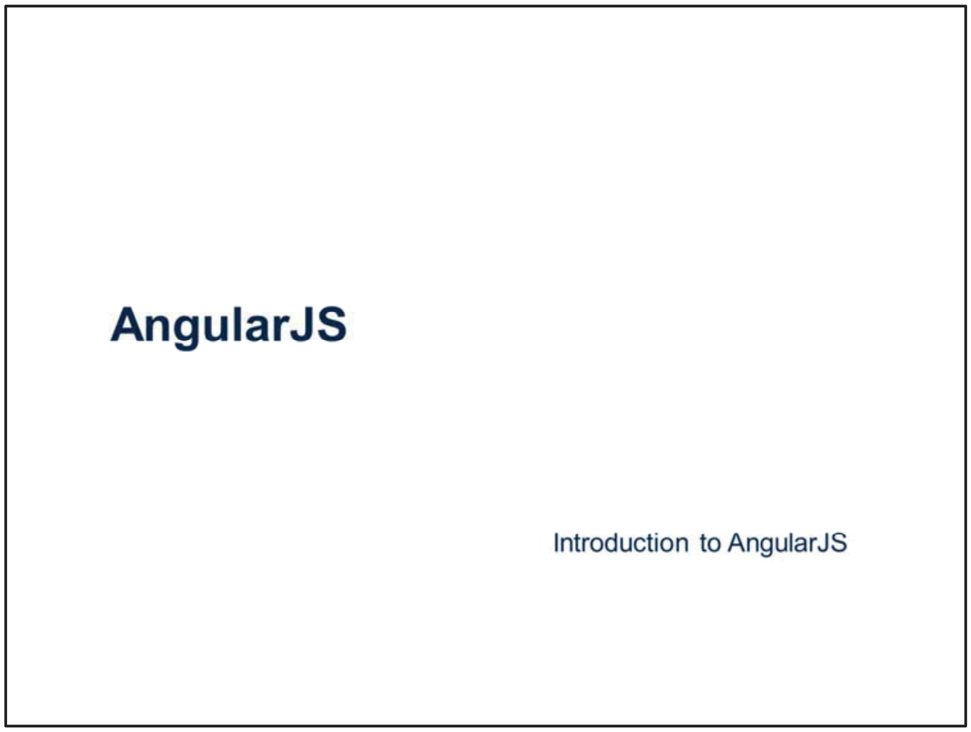
Next Step Courses (if applicable)

- Creating Web application using MEAN (MongoDB, ExpressJS, AngularJS, NodeJS) Stack



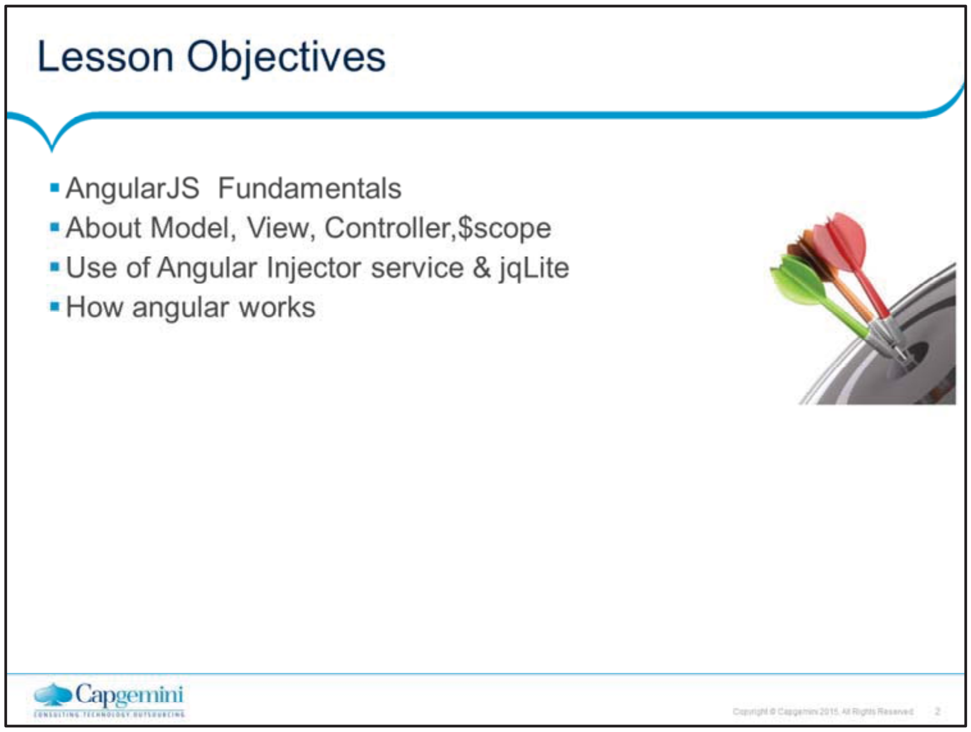
Other Parallel Technology Areas

- BackBone Js
- Ember Js
- Knockout Js



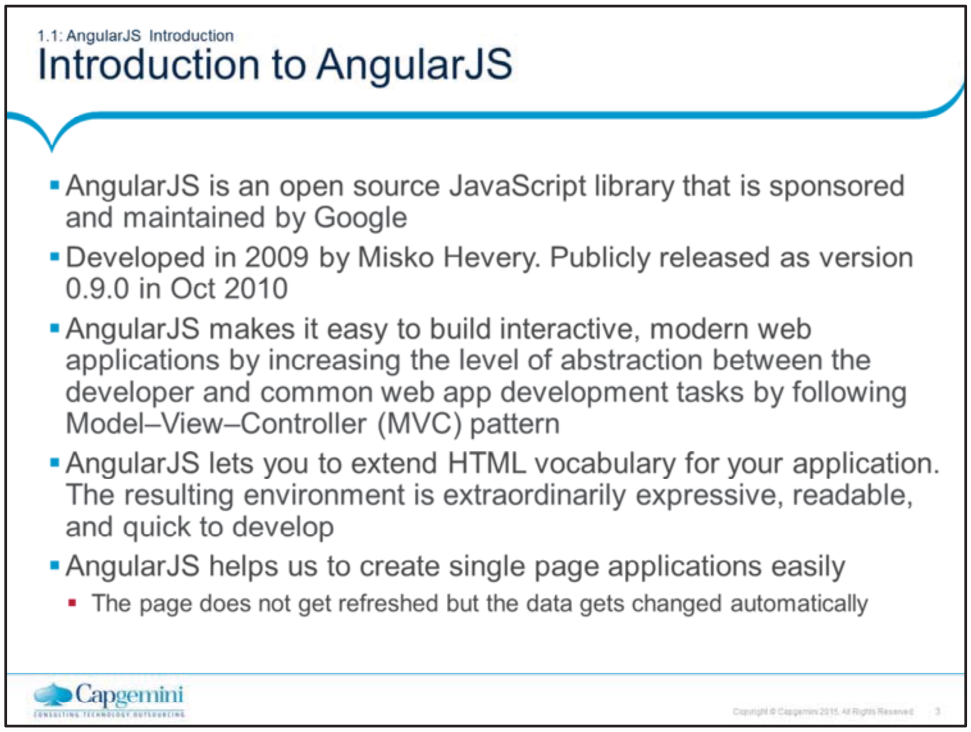
AngularJS

Introduction to AngularJS



Lesson Objectives

- AngularJS Fundamentals
- About Model, View, Controller,$scope
- Use of Angular Injector service & jqLite
- How angular works



1.1: AngularJS Introduction

Introduction to AngularJS

- AngularJS is an open source JavaScript library that is sponsored and maintained by Google
- Developed in 2009 by Misko Hevery. Publicly released as version 0.9.0 in Oct 2010
- AngularJS makes it easy to build interactive, modern web applications by increasing the level of abstraction between the developer and common web app development tasks by following Model–View–Controller (MVC) pattern
- AngularJS lets you to extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop
- AngularJS helps us to create single page applications easily
  - The page does not get refreshed but the data gets changed automatically

Capgemini

Copyright © Capgemini 2015. All Rights Reserved 3

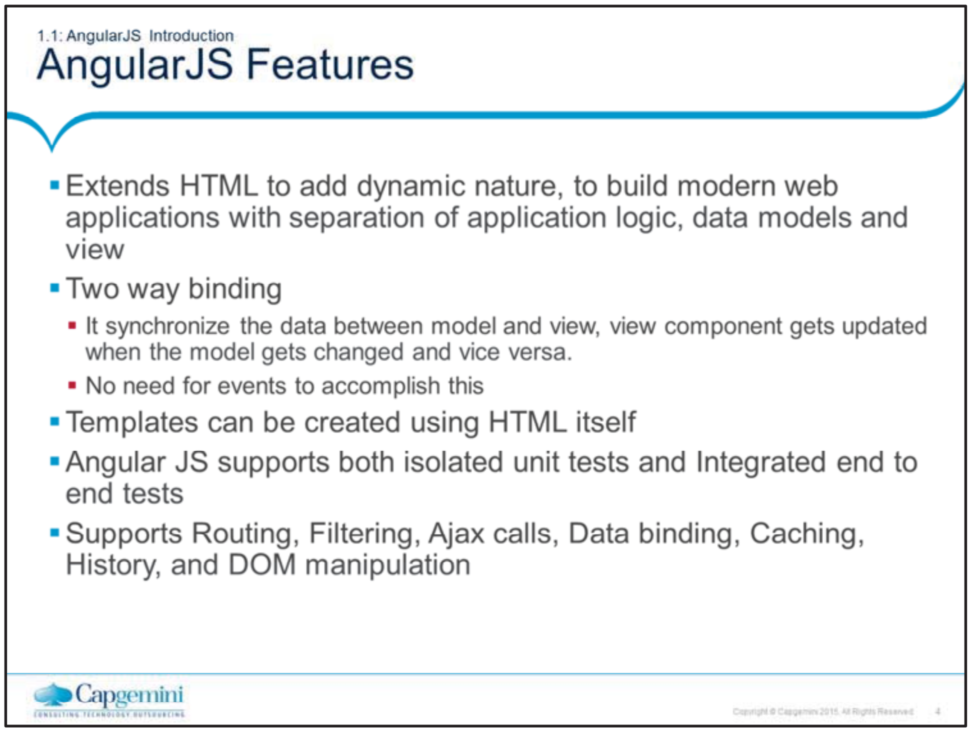
AngularJS is a client-side technology, written entirely in JavaScript.

It works with the long-established technologies of the web (HTML, CSS, and JavaScript) to make the development of web apps easier and faster than ever before.

It is a framework that is primarily used to build single-page web applications. AngularJS makes it easy to build interactive, modern web applications by increasing the level of abstraction between the developer and common web app development tasks.

The AngularJS team describes it as a “structural framework for dynamic web apps.”

Single page applications can be done with just JavaScript and AJAX calls, Angular will make this process easier



1.1: AngularJS Introduction

AngularJS Features

- Extends HTML to add dynamic nature, to build modern web applications with separation of application logic, data models and view
- Two way binding
  - It synchronize the data between model and view, view component gets updated when the model gets changed and vice versa.
  - No need for events to accomplish this
- Templates can be created using HTML itself
- Angular JS supports both isolated unit tests and Integrated end to end tests
- Supports Routing, Filtering, Ajax calls, Data binding, Caching, History, and DOM manipulation

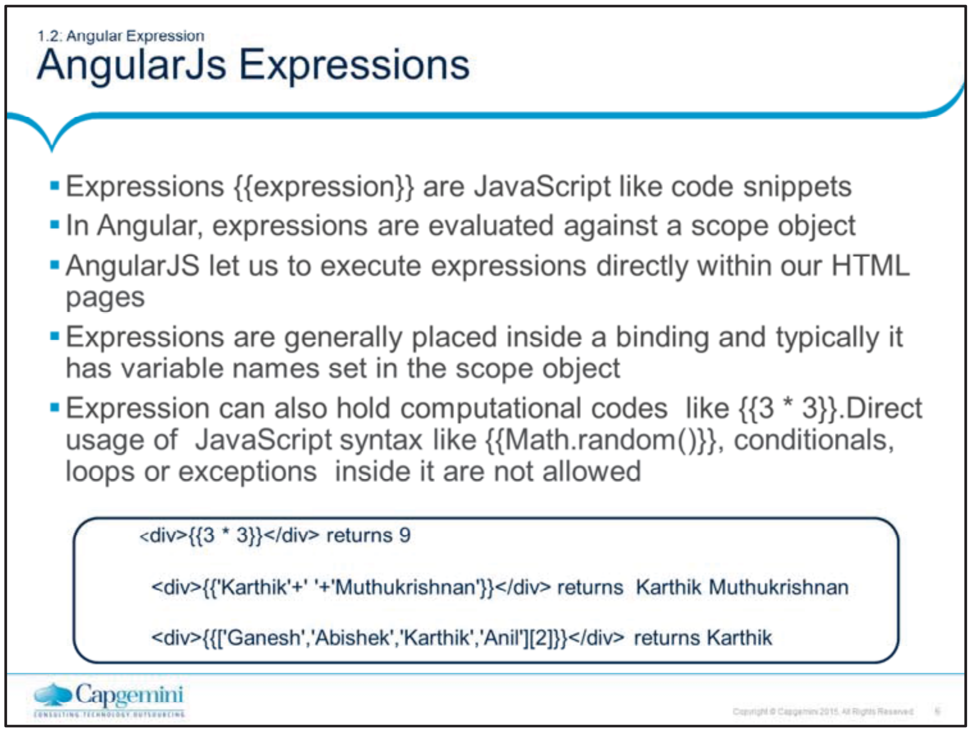
HTML is designed for static webpages, but developing a dynamic site with HTML we need to do many tricks to achieve what we want. With AngularJS extending the HTML it is really simple to make a dynamic site in a proper MVC structure.
With AngularJS we can achieve
Create a template and reuse it in application multiple times.
Can bind data to any element in two ways
Can directly call the code-behind code in your html.
Easily validate forms and input fields before submitting it.
Can control complete DOM structure show/hide, changing everything with AngularJS properties.
In other JavaScript frameworks, we are forced to extend from custom JavaScript objects and manipulate the DOM from the outside. For instance, using jQuery, to add a button in the DOM, we'll have to know where we're putting the element and insert it in the appropriate place.

```
var $btn = $("<button>jQuery Button</button>");
$("#target").append($btn);
```

AngularJS, on the other hand, augments HTML to give it native Model-View-Controller (MVC) capabilities. It enables the developer, to encapsulate a portion of entire page as one application, rather than forcing the entire page to be an AngularJS application.

This distinction is particularly beneficial, if the web application already includes another framework or if there is a need to make a portion of the page dynamic while the rest operates as a static page or is controlled by another JavaScript framework.
AJAX stands for Asynchronous JavaScript and XML. In a nutshell, it is the use of the XMLHttpRequest object to communicate with server-side scripts. It can send as well as receive information in a variety of formats, including JSON, XML, HTML, and even text files. AJAX's most appealing characteristic, however, is its "asynchronous" nature, which means it can do all of this without having to refresh the page. This lets you update portions of a page based upon user events.

Routing and Filtering would be covered in the next subsequent lessons.

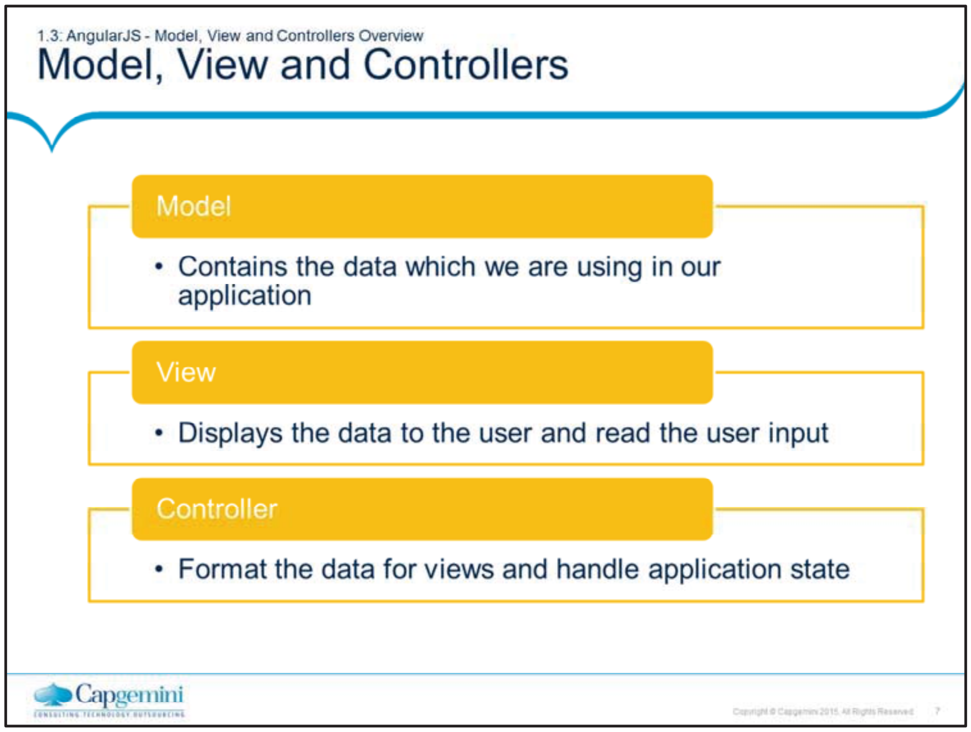


1.2: Angular Expression

AngularJs Expressions

- Expressions {{expression}} are JavaScript like code snippets
- In Angular, expressions are evaluated against a scope object
- AngularJS let us to execute expressions directly within our HTML pages
- Expressions are generally placed inside a binding and typically it has variable names set in the scope object
- Expression can also hold computational codes like {{3 * 3}}.Direct usage of JavaScript syntax like {{Math.random()}}, conditionals, loops or exceptions inside it are not allowed

```
<div>{{3 * 3}}</div> returns 9

<div>{{'Karthik'+' '+'Muthukrishnan'}}</div> returns  Karthik Muthukrishnan

<div>{{['Ganesh','Abishek','Karthik','Anil'][2]}}</div>  returns Karthik
```



1.3: AngularJS - Model, View and Controllers Overview

Model, View and Controllers

Model

- Contains the data which we are using in our application

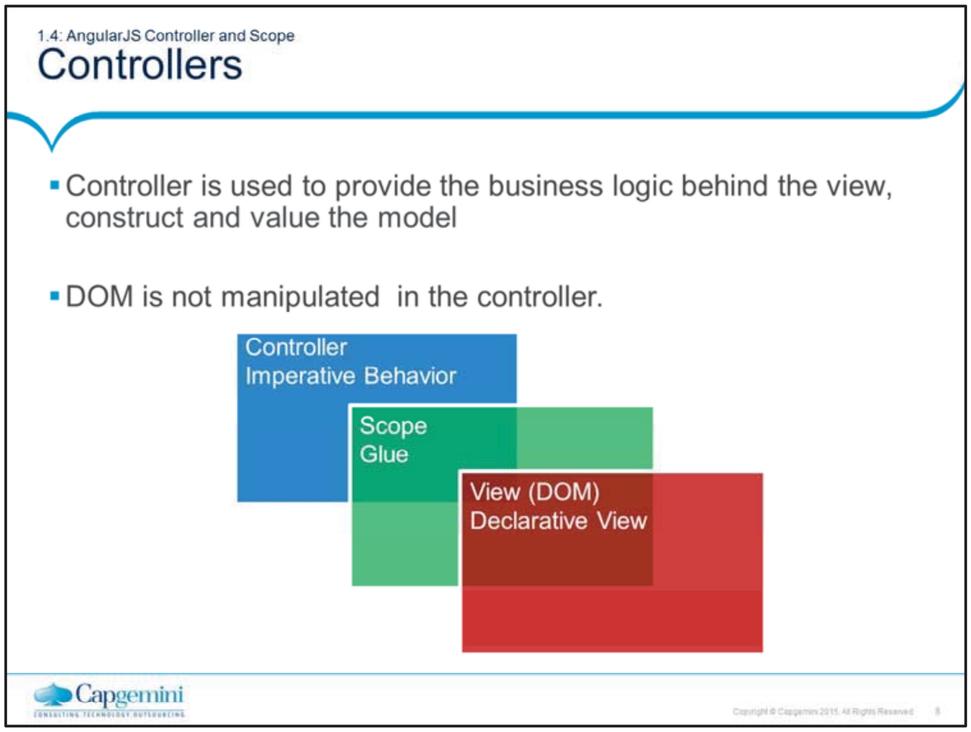
View

- Displays the data to the user and read the user input

Controller

- Format the data for views and handle application state

Capgemini CONSULTING. TECHNOLOGY. OUTSOURCING



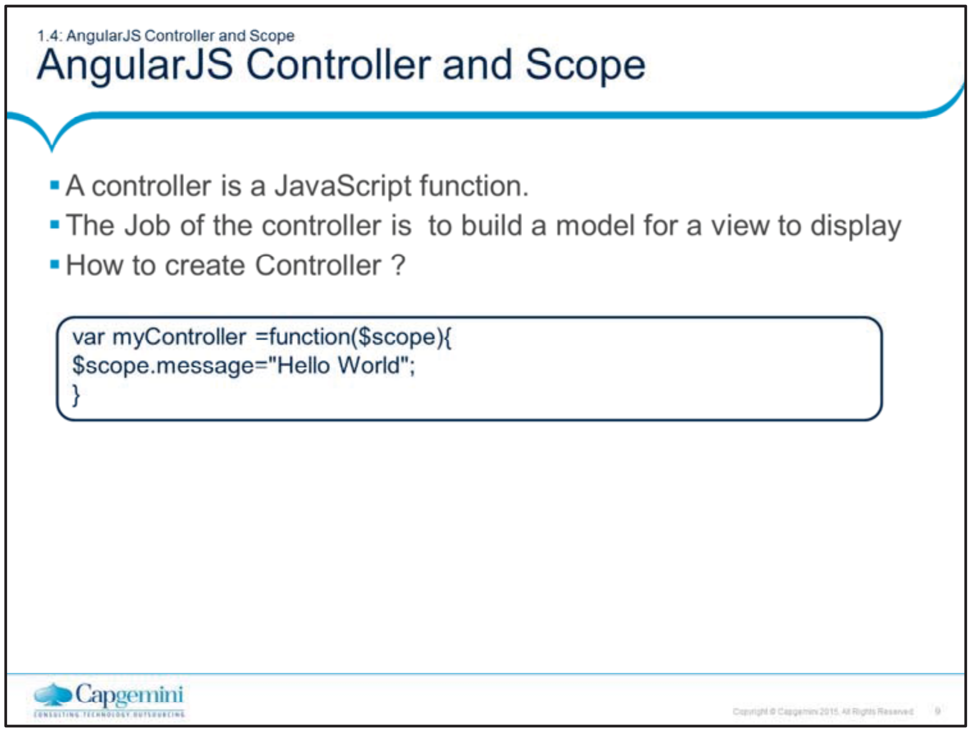
1.4: AngularJS Controller and Scope

Controllers

- Controller is used to provide the business logic behind the view, construct and value the model
- DOM is not manipulated in the controller.

Controller
Imperative Behavior
Scope
Glue
View (DOM)
Declarative View

.

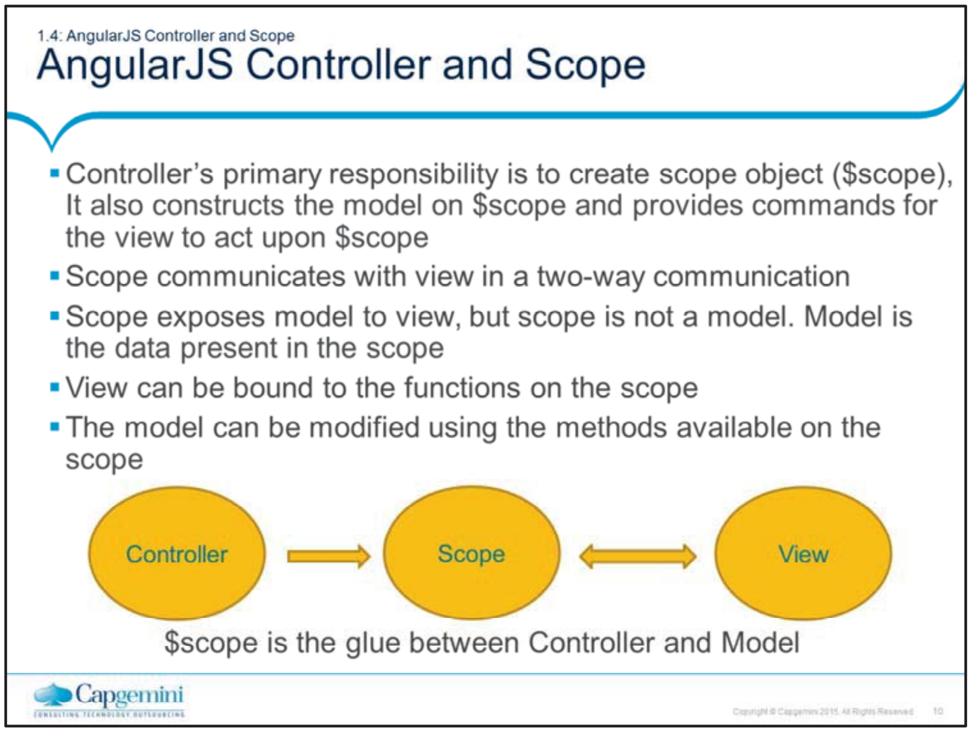


1.4: AngularJS Controller and Scope

AngularJS Controller and Scope

- A controller is a JavaScript function.
- The Job of the controller is to build a model for a view to display
- How to create Controller ?

```
var myController =function($scope){
$scope.message="Hello World";
}
```



1.4: AngularJS Controller and Scope

AngularJS Controller and Scope

- Controller's primary responsibility is to create scope object ($scope), It also constructs the model on $scope and provides commands for the view to act upon $scope
- Scope communicates with view in a two-way communication
- Scope exposes model to view, but scope is not a model. Model is the data present in the scope
- View can be bound to the functions on the scope
- The model can be modified using the methods available on the scope

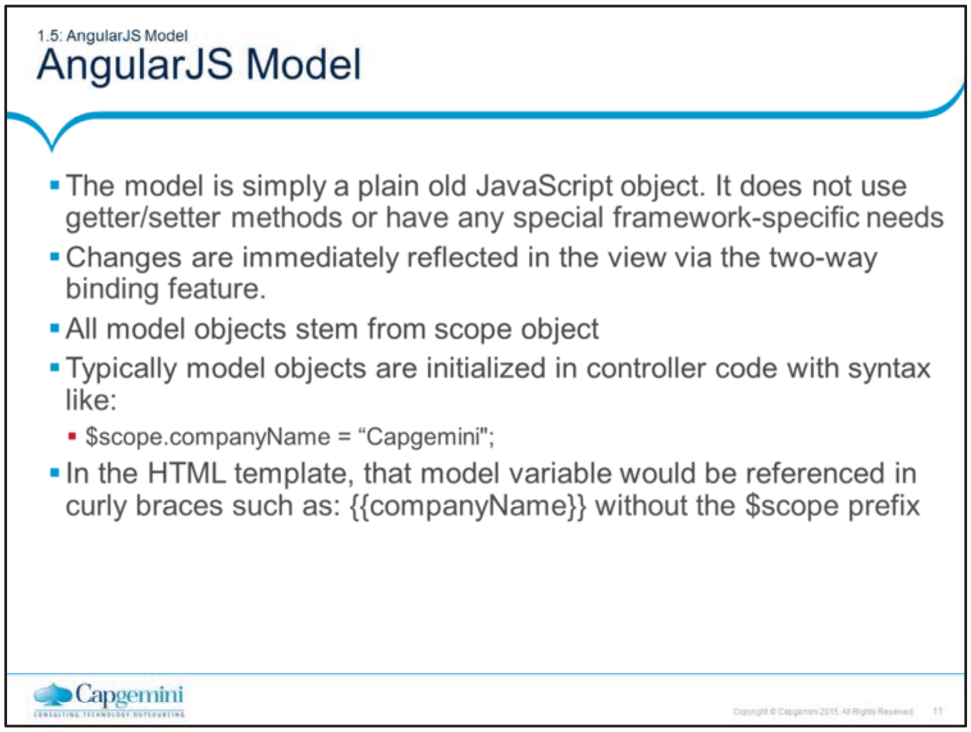
$scope is the glue between Controller and Model

Scopes are a core fundamental of any Angular app. They are used all over the framework. $scope object is where we define the business functionality of the application, the methods in our controllers, and properties in the views.

Scopes are the source of truth for the application state. Because of this live binding, we can rely on the $scope to update immediately when the view modifies it, and we can rely on the view to update when the $scope changes.

$scopes in AngularJS are arranged in a hierarchical structure so that we can reference properties on parent $scopes.

It is ideal to contain the application logic in a controller and the working data on the scope of the controller.

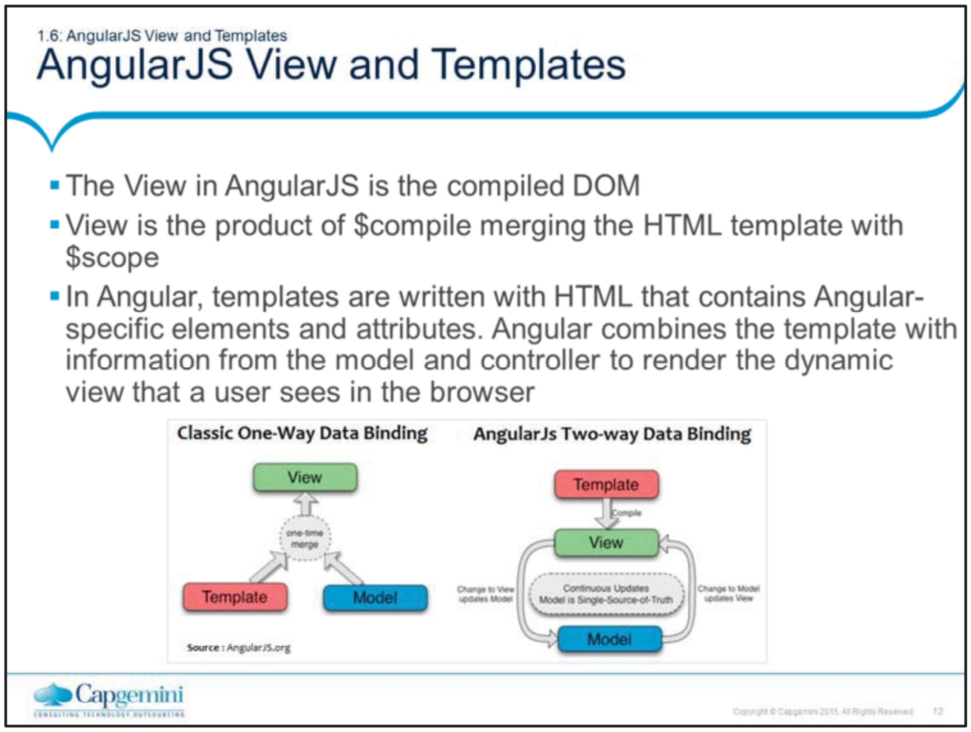


1.5: AngularJS Model

AngularJS Model

- The model is simply a plain old JavaScript object. It does not use getter/setter methods or have any special framework-specific needs
- Changes are immediately reflected in the view via the two-way binding feature.
- All model objects stem from scope object
- Typically model objects are initialized in controller code with syntax like:
  - $scope.companyName = “Capgemini";
- In the HTML template, that model variable would be referenced in curly braces such as: {{companyName}} without the $scope prefix

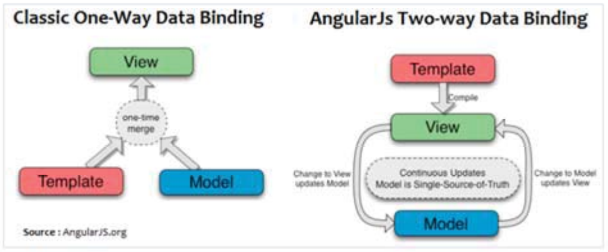
11



1.6: AngularJS View and Templates

AngularJS View and Templates

- The View in AngularJS is the compiled DOM
- View is the product of $compile merging the HTML template with $scope
- In Angular, templates are written with HTML that contains Angular-specific elements and attributes. Angular combines the template with information from the model and controller to render the dynamic view that a user sees in the browser

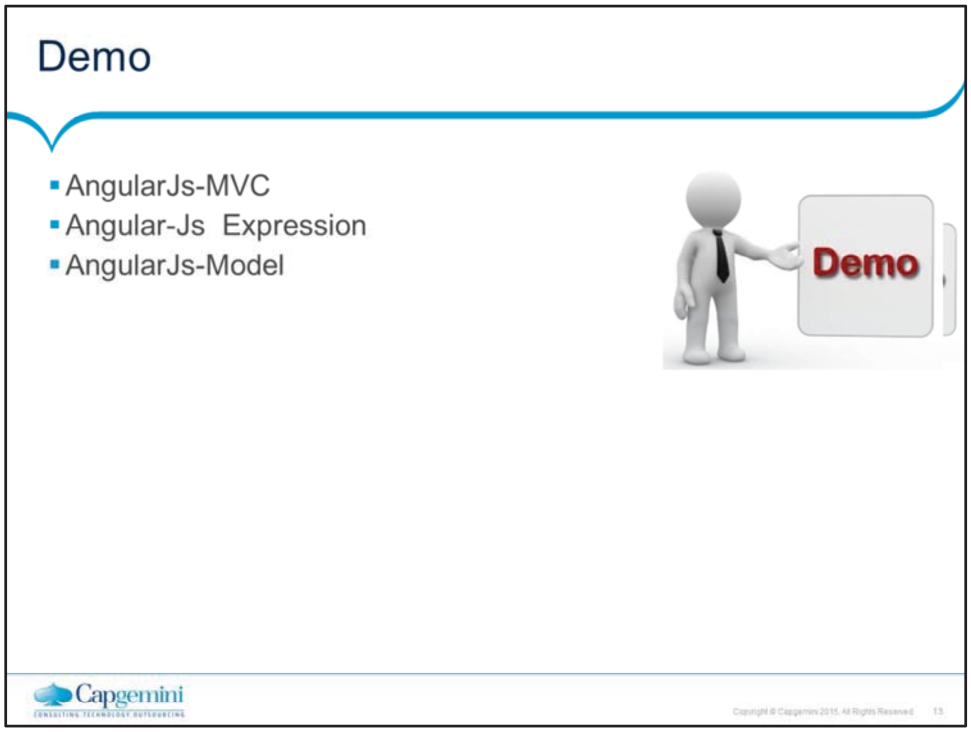

Classic One-Way Data Binding
View
one-time merge
Template
Model
AngularJs Two-way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates Model is Single-Source-of-Truth
Change to Model updates View
Model

Source : AngularJS.org

Most other templating systems consume a static string template and combine it with data, resulting in a new string. The resulting text is then innerHTMLed into an element. This means that any changes to the data need to be re-merged with the template and then innerHTMLed into the DOM.

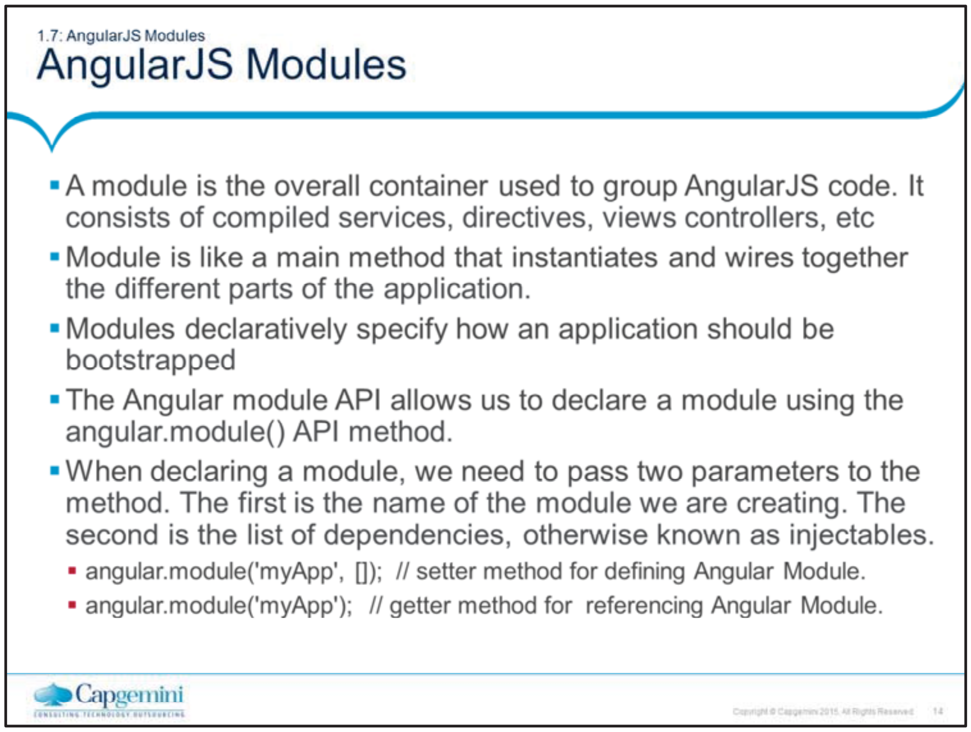
Angular is different. The Angular compiler consumes the DOM, not string templates. The result is a linking function, which when combined with a scope model results in a live view. The view and scope model bindings are transparent. The developer does not need to make any special calls to update the view. And because innerHTML is not used, you won't accidentally clobber user input. Furthermore, Angular directives can contain not just text bindings, but behavioral constructs as well.

The Angular approach produces a stable DOM. The DOM element instance bound to a model item instance does not change for the lifetime of the binding. This means that the code can get hold of the elements and register event handlers and know that the reference will not be destroyed by template data merge.
$compile: Compiles an HTML string or DOM into a template and produces a template function, which can then be used to link scope and the template together.



Demo

- AngularJs-MVC
- Angular-Js Expression
- AngularJs-Model



1.7: AngularJS Modules

AngularJS Modules

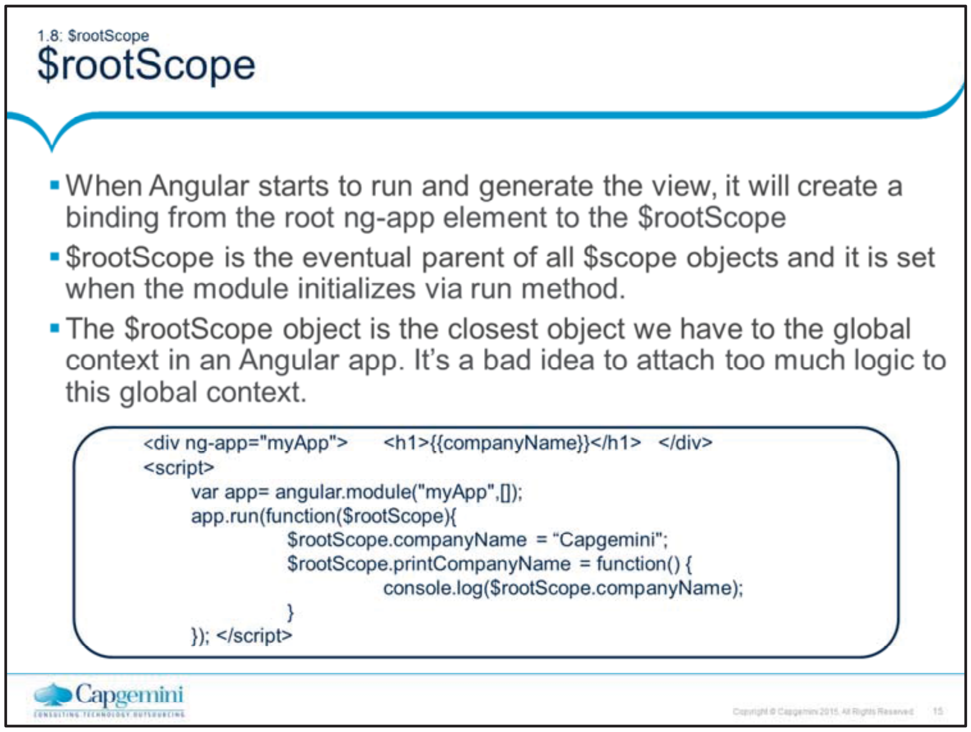
- A module is the overall container used to group AngularJS code. It consists of compiled services, directives, views controllers, etc
- Module is like a main method that instantiates and wires together the different parts of the application.
- Modules declaratively specify how an application should be bootstrapped
- The Angular module API allows us to declare a module using the angular.module() API method.
- When declaring a module, we need to pass two parameters to the method. The first is the name of the module we are creating. The second is the list of dependencies, otherwise known as injectables.
  - angular.module('myApp', []); // setter method for defining Angular Module.
  - angular.module('myApp'); // getter method for referencing Angular Module.

Advantages of Modules

- Keeping our global namespace clean
- Making tests easier to write and keeping them clean so as to more easily target isolated functionality
- Making it easy to share code between applications
- Allowing our app to load different parts of the code in any order

When writing large applications, we can create several different modules to contain our logic. Creating a module for each piece of functionality gives us the advantage of isolation to write and test.

We would be covering services, directives in the next subsequent lessons



1.8: $rootScope

$rootScope

- When Angular starts to run and generate the view, it will create a binding from the root ng-app element to the $rootScope
- $rootScope is the eventual parent of all $scope objects and it is set when the module initializes via run method.
- The $rootScope object is the closest object we have to the global context in an Angular app. It's a bad idea to attach too much logic to this global context.

```
<div ng-app="myApp">      <h1>{{companyName}}</h1>   </div>
<script>
      var app= angular.module("myApp",[]);
      app.run(function($rootScope){
                  $rootScope.companyName  = "Capgemini";
                  $rootScope.printCompanyName  = function() {
                              console.log($rootScope.companyName);
                  }
      }); </script>
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

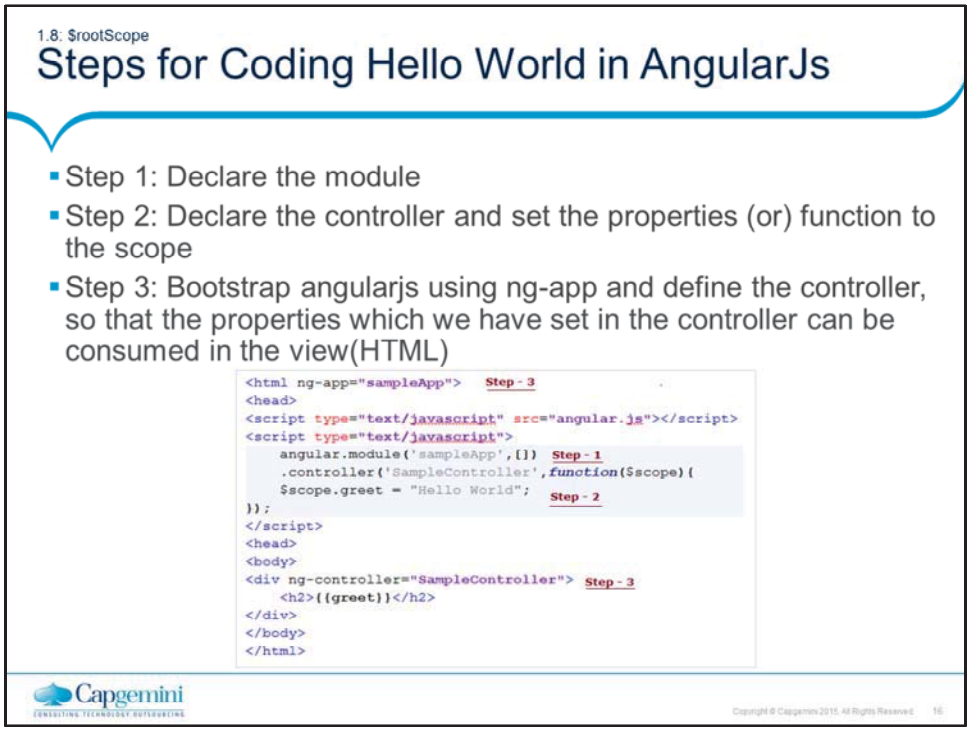
Copyright © Capgemini 2015. All Rights Reserved 15

ng-app is used to mark the beginning of the angular application in our web page.
Run blocks - get executed after the injector is created and are used to kickstart the application. Only instances and constants can be injected into run blocks. This is to prevent further system configuration during application run time.

Run blocks are the closest thing in Angular to the main method. It is executed after all of the service have been configured and the $injector($injector is inbuilt service to the angular framework to inject other services on demand - a form of dependency injection) has been created. Run block can be thought of as an init() method.

Variables set at the root-scope are available to the controller scope via inheritance.
Note: anything prefixed in angular with '$' is a service.

```
<div ng-app="myApp" ng-controller="myCntrl">
   {{company}}
</div>

var app= angular.module("myApp",[]);
app.run(function($rootScope){
            $rootScope.companyName = "Microsoft";
            $rootScope.printCompanyName = function() {
                                    console.log($rootScope.companyName);
            }
});

app.controller("myCntrl",function($scope,$rootScope){
            console.debug($rootScope.companyName);
            $rootScope.companyName= "IGATE"; //Changing Value
            $scope.company = $rootScope.companyName;
});
```



1.8: $rootScope

Steps for Coding Hello World in AngularJs

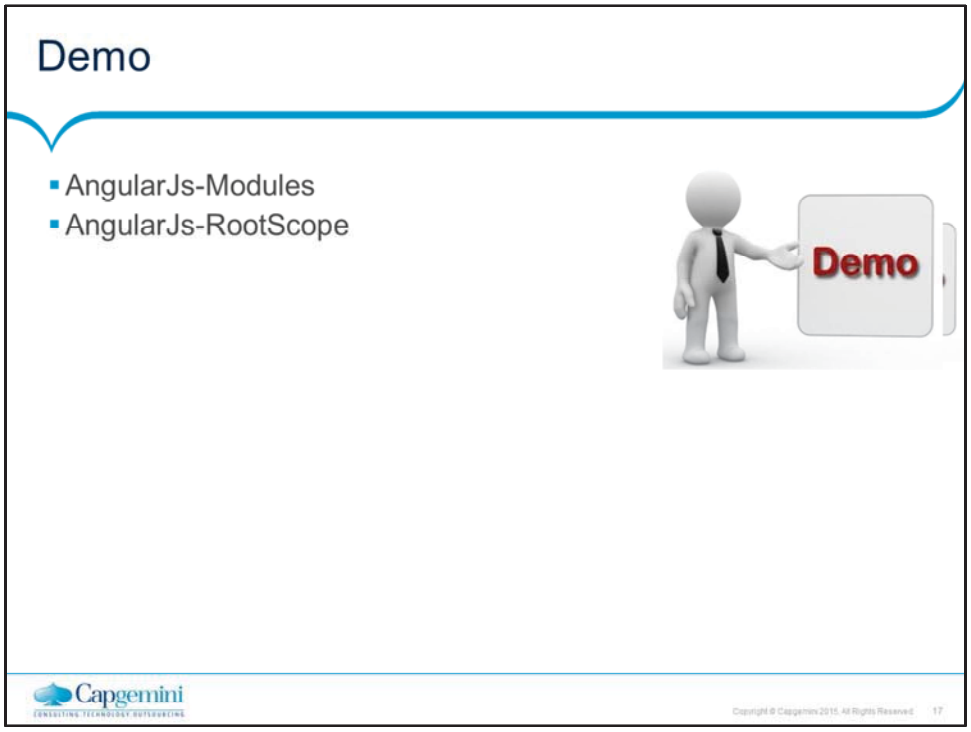
- Step 1: Declare the module
- Step 2: Declare the controller and set the properties (or) function to the scope
- Step 3: Bootstrap angularjs using ng-app and define the controller, so that the properties which we have set in the controller can be consumed in the view(HTML)

```
<html ng-app="sampleApp">    Step - 3
<head>
<script type="text/javascript" src="angular.js"></script>
<script type="text/javascript">
    angular.module('sampleApp',[])  Step - 1
    .controller('SampleController',function($scope){
    $scope.greet = "Hello World";   Step - 2
});
</script>
<head>
<body>
<div ng-controller="SampleController">  Step - 3
    <h2>{{greet}}</h2>
</div>
</body>
</html>
```

Capgemini

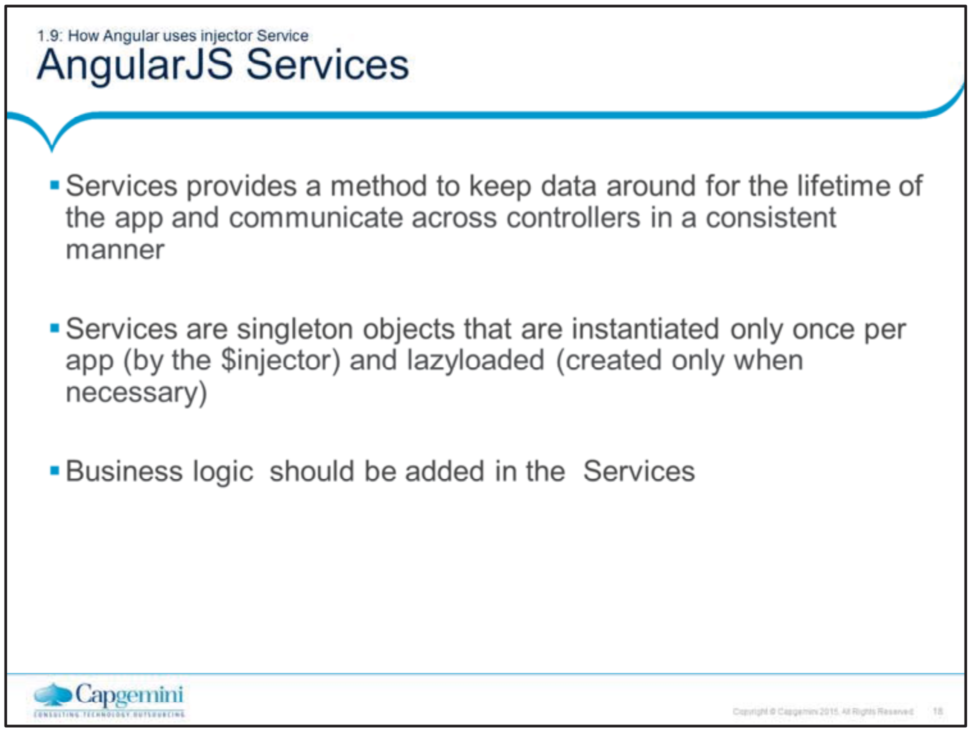
Copyright © Capgemini 2015. All Rights Reserved 16

Step 1 Declare a Module by using angular.module ,'sampleApp' this sampleApp have same name as ng-app
Step 2 Attach with controller 'SampleController' have same name what we write in ng-controller



Demo

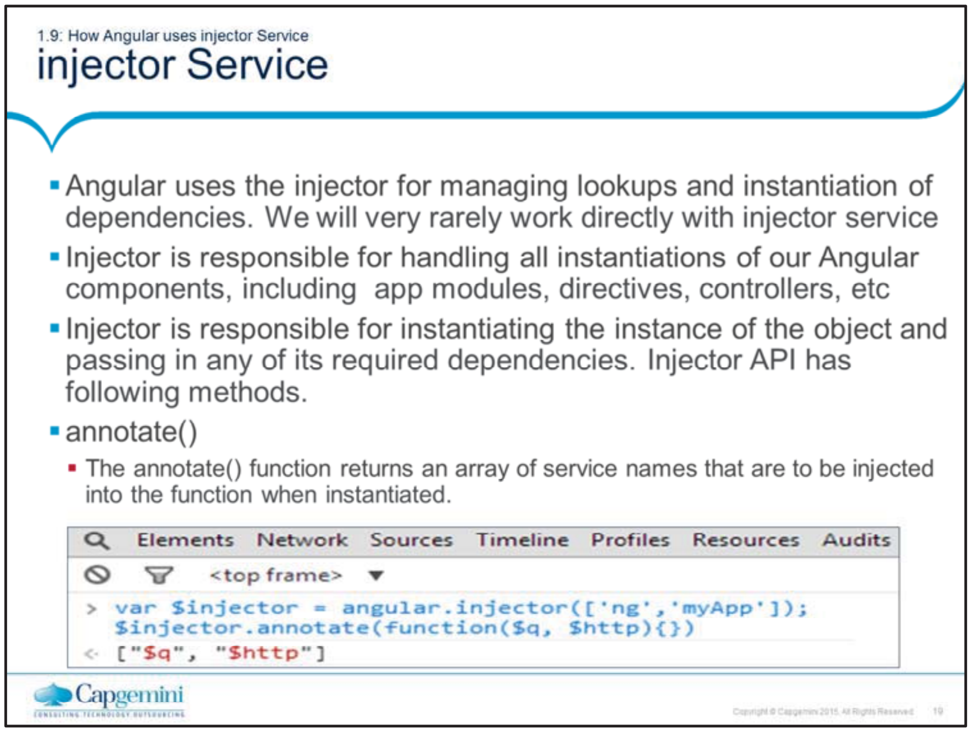
- AngularJs-Modules
- AngularJs-RootScope



1.9: How Angular uses injector Service

AngularJS Services

- Services provides a method to keep data around for the lifetime of the app and communicate across controllers in a consistent manner
- Services are singleton objects that are instantiated only once per app (by the $injector) and lazyloaded (created only when necessary)
- Business logic should be added in the Services


Elements
Network
Sources
Timeline
Profiles
Resources
Audits
<top frame>

1.9: How Angular uses injector Service

injector Service

- Angular uses the injector for managing lookups and instantiation of dependencies. We will very rarely work directly with injector service
- Injector is responsible for handling all instantiations of our Angular components, including app modules, directives, controllers, etc
- Injector is responsible for instantiating the instance of the object and passing in any of its required dependencies. Injector API has following methods.
- annotate()
  - The annotate() function returns an array of service names that are to be injected into the function when instantiated.

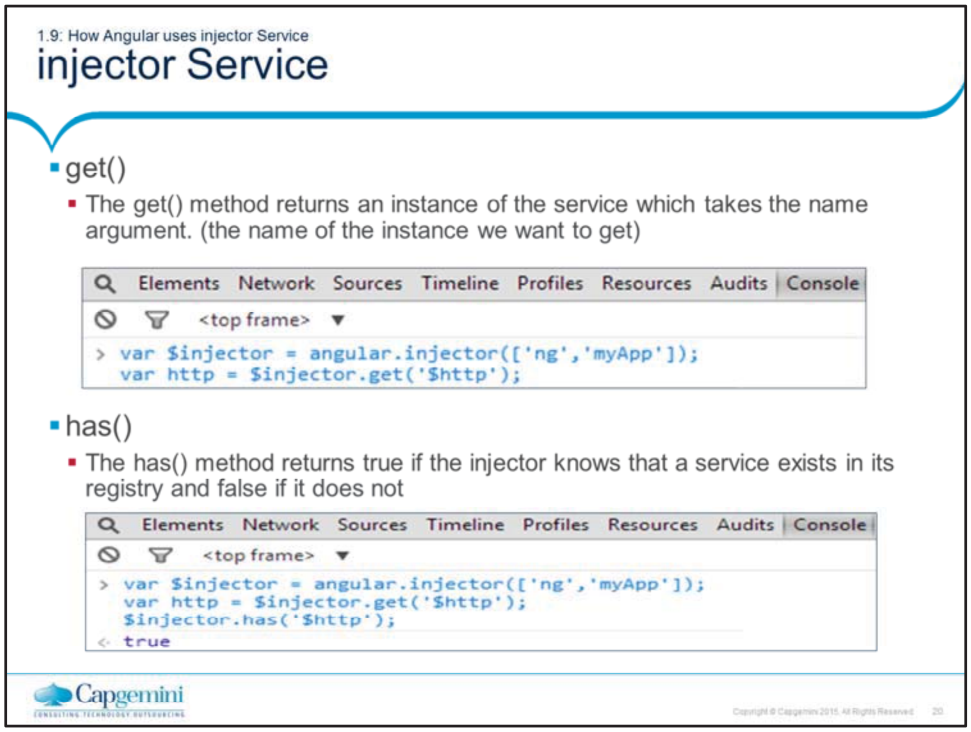
```
> var $injector = angular.injector(['ng','myApp']);
  $injector.annotate(function($q, $http){})
<- ["$q", "$http"]
```

$injector is used to retrieve object instances as defined by instantiate types, invoke methods, and load modules. In JavaScript calling toString() on a function returns the function definition. The definition can then be parsed and the function arguments can be extracted. This method of discovering annotations is disallowed when the injector is in strict mode. By adding an $inject property onto a function the injection parameters can be specified.

annotate(fn, [strictDi]);
Returns an array of service names which the function is requesting for injection. This API is used by the injector to determine which services need to be injected into the function when the function is invoked. There are three ways in which the function can be annotated with the needed dependencies.

1. Argument names
2. The $inject property
3. The array notation

We shall discuss about $http & $q in Lesson 4



1.9: How Angular uses injector Service

injector Service

- get()
  - The get() method returns an instance of the service which takes the name argument. (the name of the instance we want to get)

Elements Network Sources Timeline Profiles Resources Audits Console

<top frame> ▼

```
> var $injector = angular.injector(['ng','myApp']);
  var http = $injector.get('$http');
```

- has()
  - The has() method returns true if the injector knows that a service exists in its registry and false if it does not

Elements Network Sources Timeline Profiles Resources Audits Console

<top frame> ▼

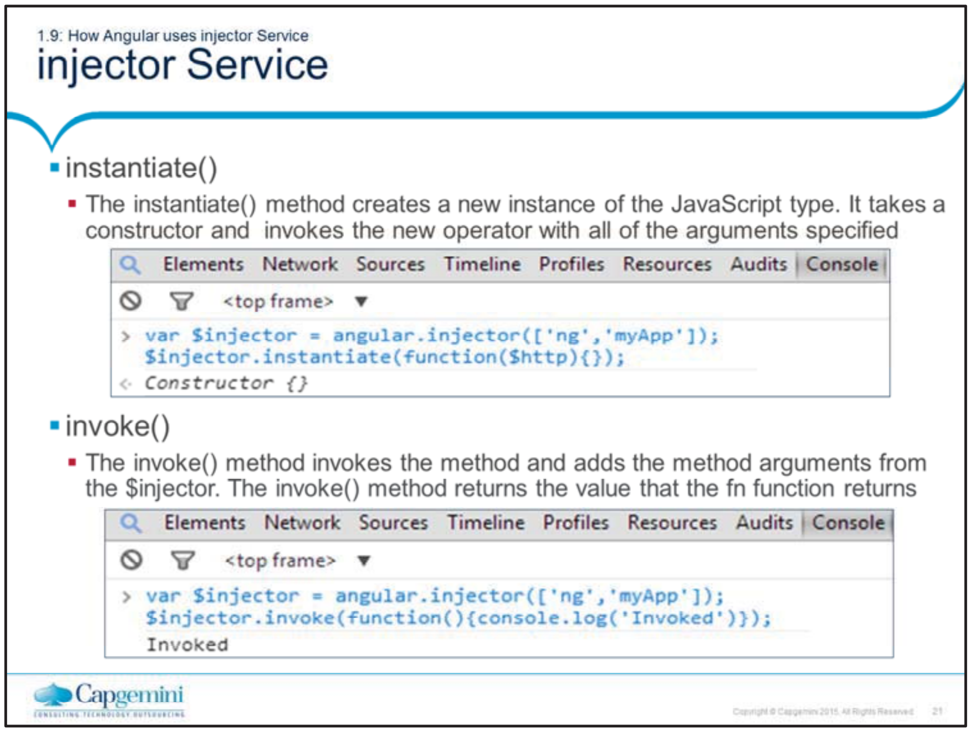
```
> var $injector = angular.injector(['ng','myApp']);
  var http = $injector.get('$http');
  $injector.has('$http');
<· true
```

Capgemini

get(name, [caller]);
Return an instance of the service.

Caller is optional

has(name);
Allows the user to query if the particular service exists.



1.9: How Angular uses injector Service

injector Service

- instantiate()
  - The instantiate() method creates a new instance of the JavaScript type. It takes a constructor and invokes the new operator with all of the arguments specified

Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>

```
> var $injector = angular.injector(['ng','myApp']);
  $injector.instantiate(function($http){});
< Constructor {}
```

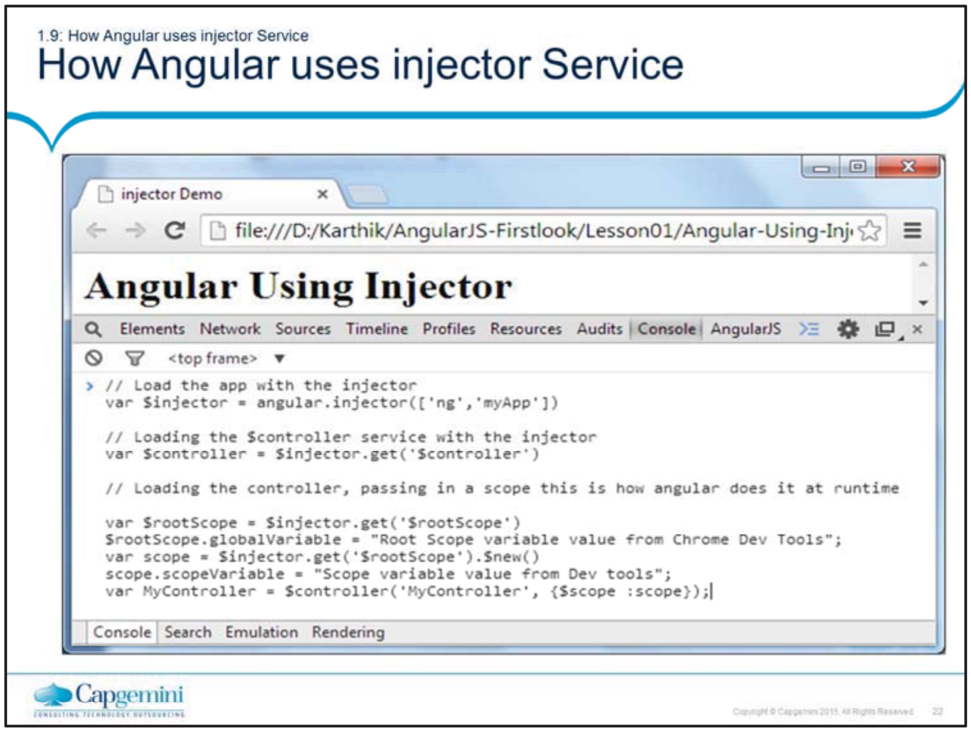
- invoke()
  - The invoke() method invokes the method and adds the method arguments from the $injector. The invoke() method returns the value that the fn function returns

Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>

```
> var $injector = angular.injector(['ng','myApp']);
  $injector.invoke(function(){console.log('Invoked')});
  Invoked
```

invoke(fn, [self], [locals]);
Invoke the method and supply the method arguments from the $injector.

instantiate(Type, [locals]);
Create a new instance of JS type. The method takes a constructor function, invokes the new operator, and supplies all of the arguments to the constructor function as specified by the constructor annotation.


1.9: How Angular uses injector Service

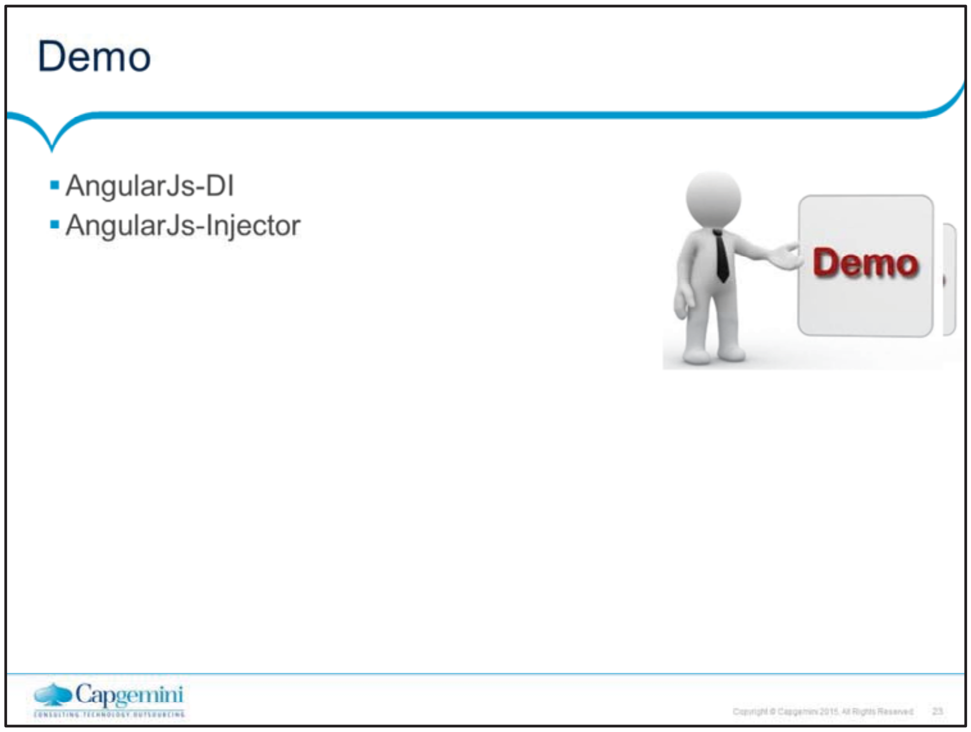
How Angular uses injector Service

injector Demo
file:///D:/Karthik/AngularJS-Firstlook/Lesson01/Angular-Using-Inj
Elements Network Sources Timeline Profiles Resources Audits Console AngularJS
<top frame>
Console Search Emulation Rendering

Angular Using Injector

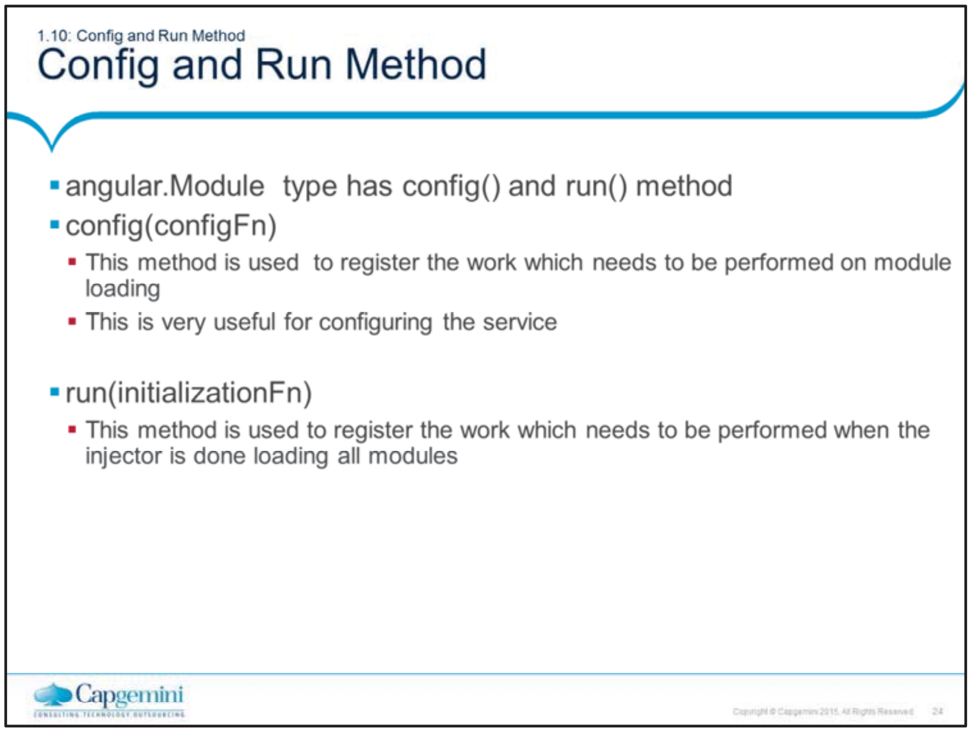
```
// Load the app with the injector
var $injector = angular.injector(['ng','myApp'])

// Loading the $controller service with the injector
var $controller = $injector.get('$controller')

// Loading the controller, passing in a scope this is how angular does it at runtime

var $rootScope = $injector.get('$rootScope')
$rootScope.globalVariable = "Root Scope variable value from Chrome Dev Tools";
var scope = $injector.get('$rootScope').$new()
scope.scopeVariable = "Scope variable value from Dev tools";
var MyController = $controller('MyController', {$scope :scope});
```

```
// Load the app with the injector
var $injector = angular.injector(['ng','myApp'])

// Loading the $controller service with the injector
var $controller = $injector.get('$controller')

// Loading the controller, passing in a scope this is how angular
does it at runtime

var $rootScope = $injector.get('$rootScope')
$rootScope.globalVariable = "Root Scope variable value from
Chrome Dev Tools";
var scope = $injector.get('$rootScope').$new()
scope.scopeVariable = "Scope variable value from Dev tools";
var MyController = $controller('MyController', {$scope :scope});
```



Demo

- AngularJs-DI
- AngularJs-Injector



1.10: Config and Run Method

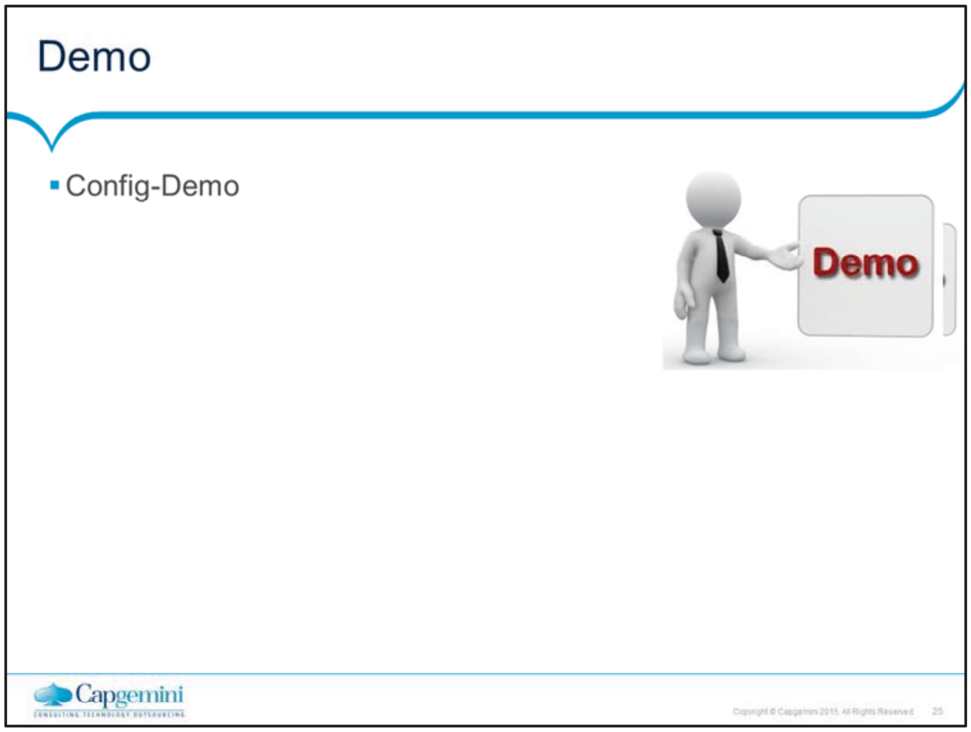
Config and Run Method

- angular.Module type has config() and run() method
- config(configFn)
  - This method is used to register the work which needs to be performed on module loading
  - This is very useful for configuring the service
- run(initializationFn)
  - This method is used to register the work which needs to be performed when the injector is done loading all modules

Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

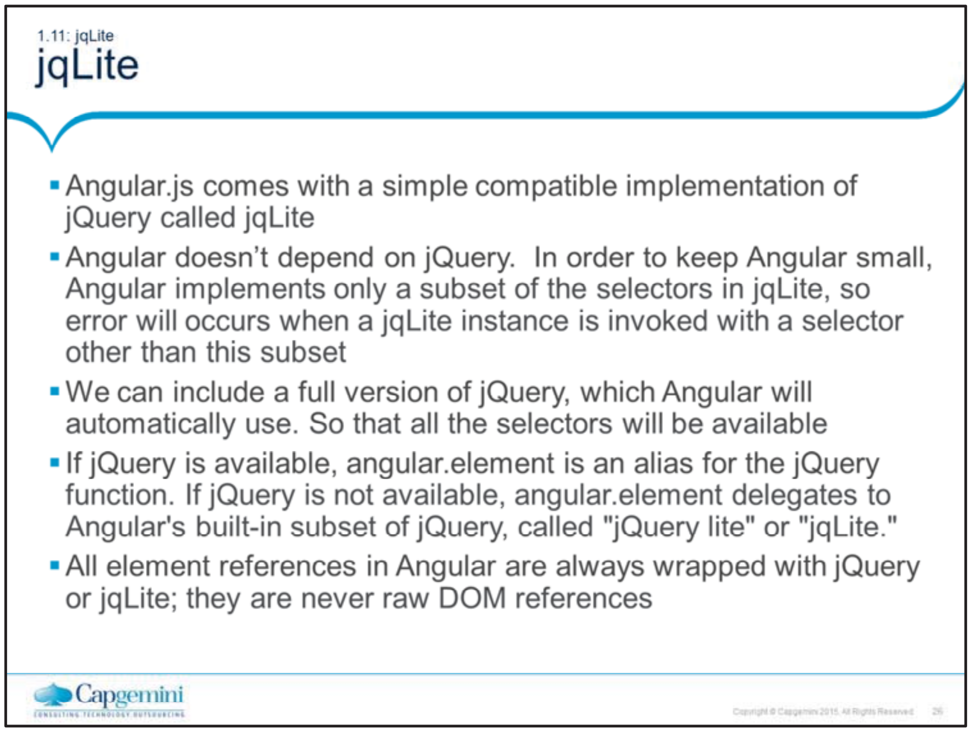
Copyright © Capgemini 2015. All Rights Reserved 24

A module is a collection of configuration and run blocks which get applied to the application during the bootstrap process. In its simplest form the module consists of a collection of two kinds of blocks:
Configuration blocks - get executed during the provider registrations and configuration phase. Only providers and constants can be injected into configuration blocks. This is to prevent accidental instantiation of services before they have been fully configured.
Run blocks - get executed after the injector is created and are used to kickstart the application. Only instances and constants can be injected into run blocks. This is to prevent further system configuration during application run time.



Demo

- Config-Demo



1.11: jqLite

jqLite

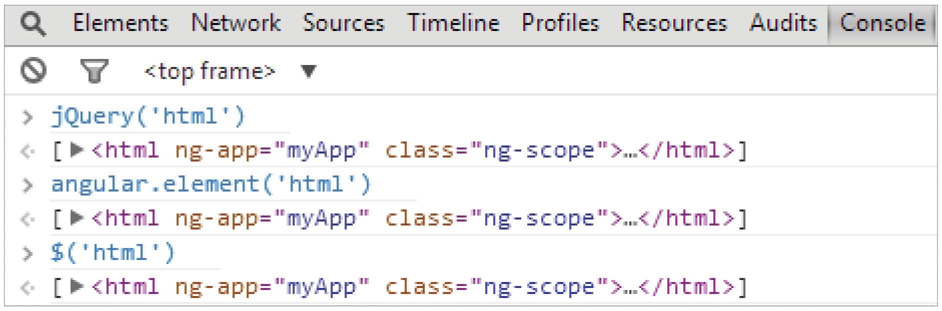
- Angular.js comes with a simple compatible implementation of jQuery called jqLite
- Angular doesn't depend on jQuery. In order to keep Angular small, Angular implements only a subset of the selectors in jqLite, so error will occurs when a jqLite instance is invoked with a selector other than this subset
- We can include a full version of jQuery, which Angular will automatically use. So that all the selectors will be available
- If jQuery is available, angular.element is an alias for the jQuery function. If jQuery is not available, angular.element delegates to Angular's built-in subset of jQuery, called "jQuery lite" or "jqLite."
- All element references in Angular are always wrapped with jQuery or jqLite; they are never raw DOM references

Angular's jqLite

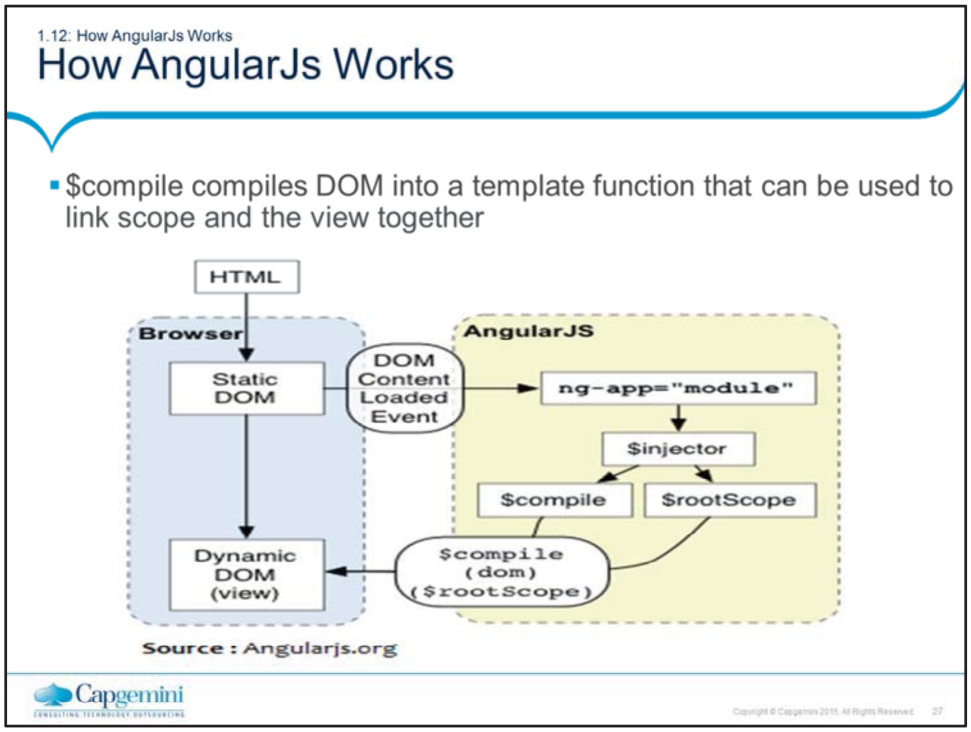
jqLite provides only the following jQuery methods:

addClass(), after() , append() , attrclone(), contents() ,
css(), data(), detach(), empty(), eq(), hasClass(),
html(), prepend(), prop(), ready(), remove(),
removeAttr(), removeClass(), removeData(), replaceWith() ,
text(), toggleClass(), val() & wrap()

children(), parent(), next() - Does not support selectors,
find() - Limited to lookups by tag name,
bind() - Does not support namespaces, selectors or eventData,
on() - Does not support namespaces, selectors or eventData,
off() - Does not support namespaces or selectors,
one() - Does not support namespaces or selectors
unbind() - Does not support namespaces
triggerHandler() - Passes a dummy event object to handlers.


Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>

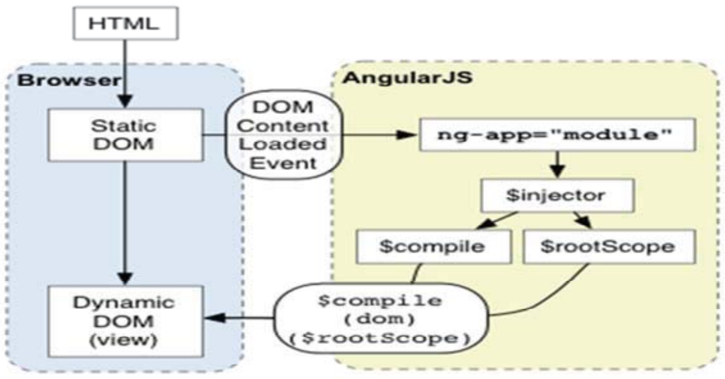
```
> jQuery('html')
< [▶<html ng-app="myApp" class="ng-scope">…</html>]
> angular.element('html')
< [▶<html ng-app="myApp" class="ng-scope">…</html>]
> $('html')
< [▶<html ng-app="myApp" class="ng-scope">…</html>]
```



1.12: How AngularJs Works

How AngularJs Works

- $compile compiles DOM into a template function that can be used to link scope and the view together



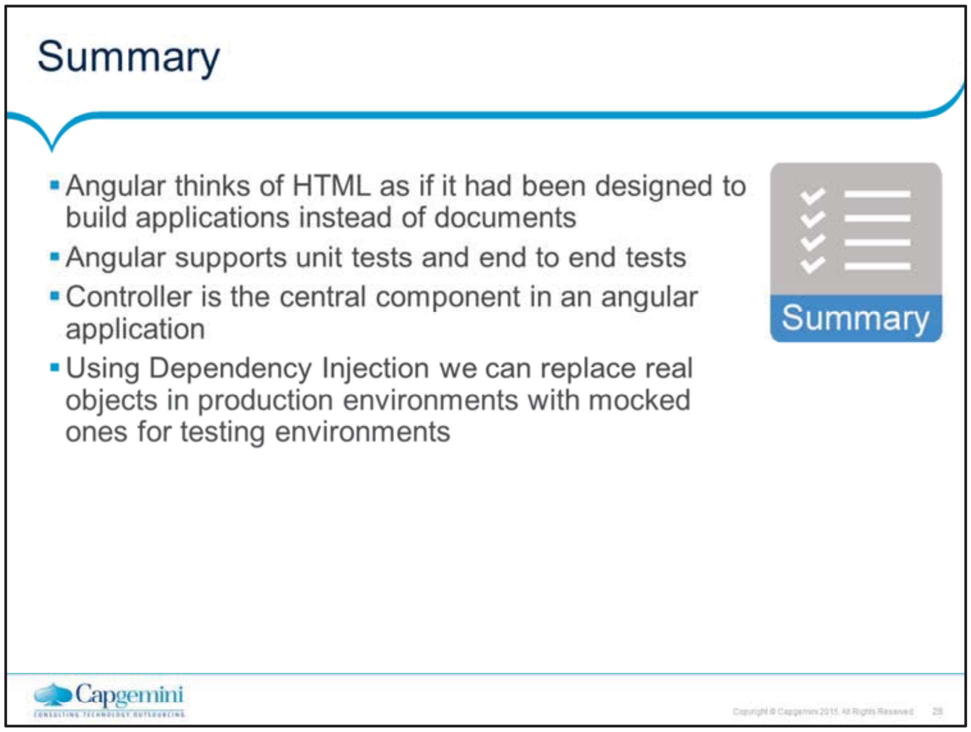
Source : Angularjs.org

Angular initializes automatically upon DOMContentLoaded event or when the angular.js script is evaluated if at that time document.readyState is set to 'complete'.

Following are key method invocations that happen as part of initializing angular app and rendering the same.

- angularInit method which checks for ng-app module
- bootstrap method which is invoked once an ng-app module is found. Following are key invocations from within bootstrap method:
  - createInjector method which returns dependency injector. On dependency injector instance, invoke method is called.
  - compile method which collects directives
  - Composite linking method which is returned from compile method. Scope object is passed to this composite linking method
  - $apply method invoked on scope object finally does the magic and renders the view.

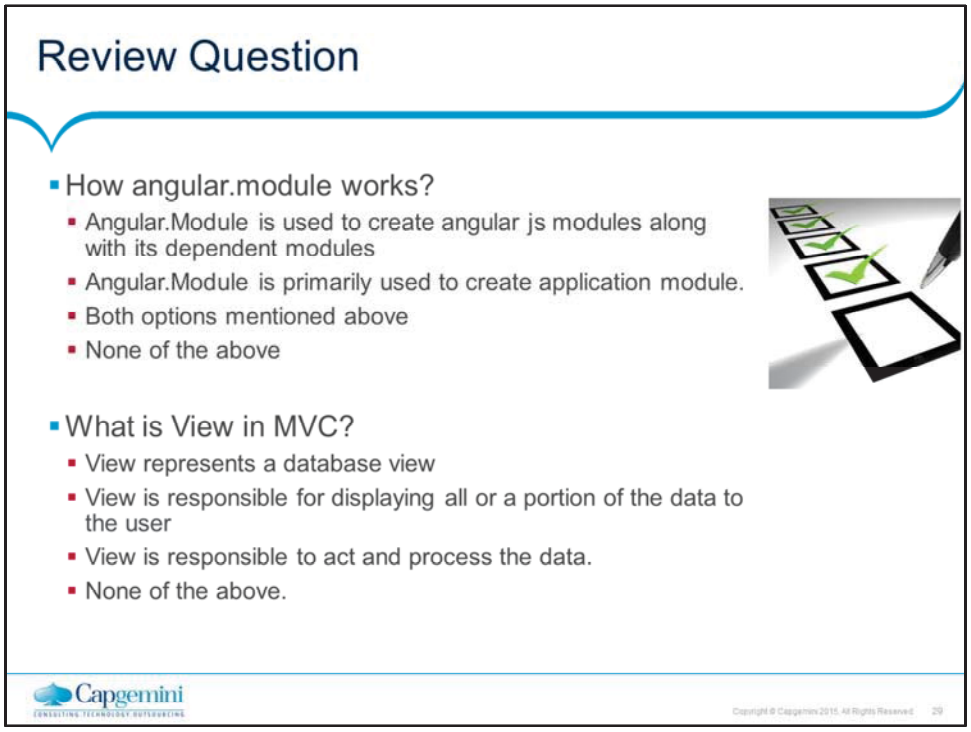
  Note: above mentioned steps are inherently being invoked when ng-app is being bootstrapped.



Summary

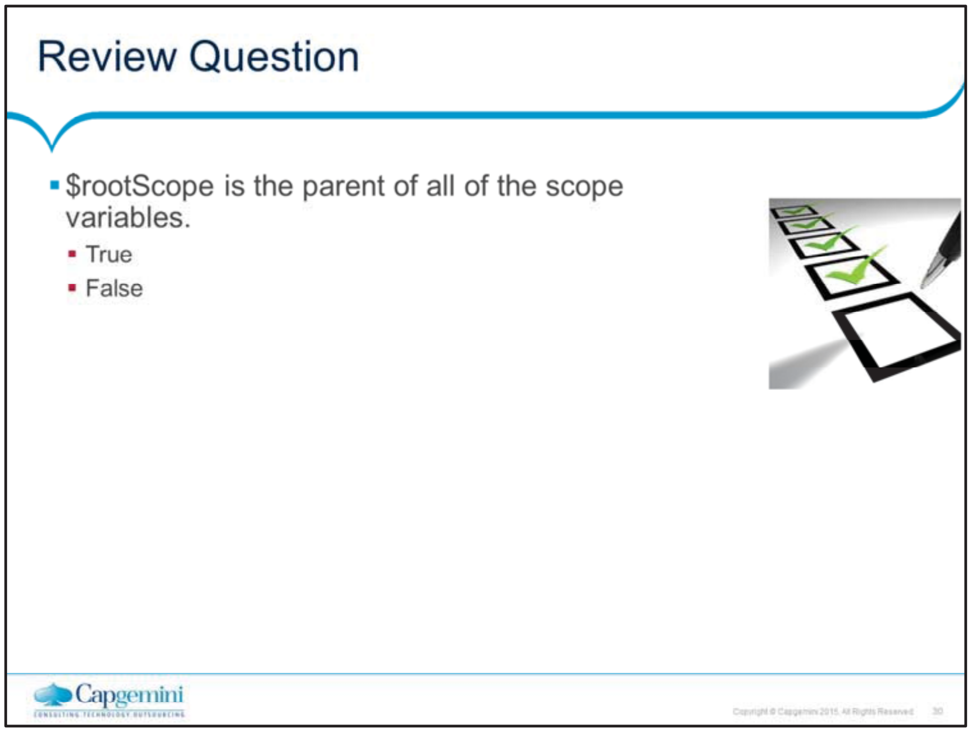
- Angular thinks of HTML as if it had been designed to build applications instead of documents
- Angular supports unit tests and end to end tests
- Controller is the central component in an angular application
- Using Dependency Injection we can replace real objects in production environments with mocked ones for testing environments

Add the notes here.



Review Question

- How angular.module works?
  - Angular.Module is used to create angular js modules along with its dependent modules
  - Angular.Module is primarily used to create application module.
  - Both options mentioned above
  - None of the above
- What is View in MVC?
  - View represents a database view
  - View is responsible for displaying all or a portion of the data to the user
  - View is responsible to act and process the data.
  - None of the above.

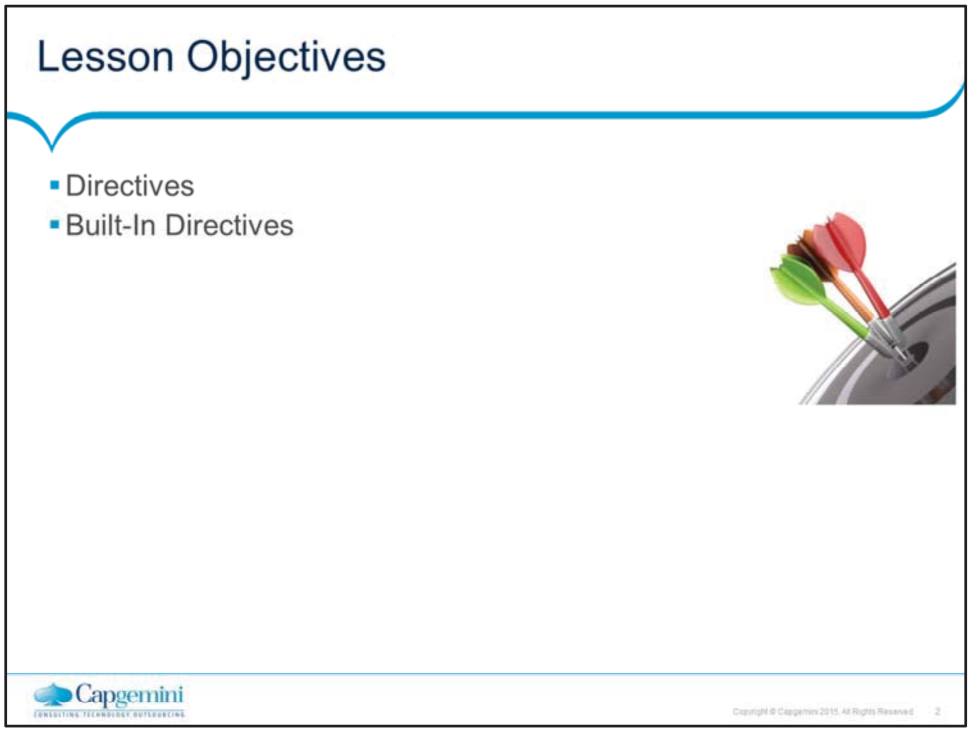


Review Question

- $rootScope is the parent of all of the scope variables.
  - True
  - False

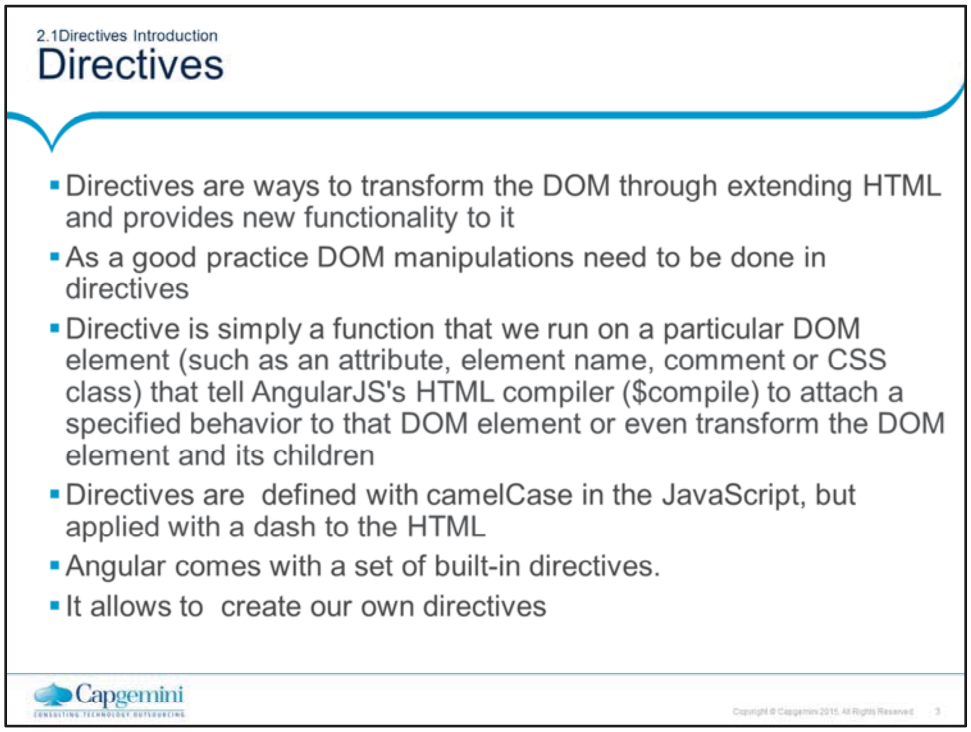
AngularJS

AngularJS Directives



Lesson Objectives

- Directives
- Built-In Directives



2.1Directives Introduction

Directives

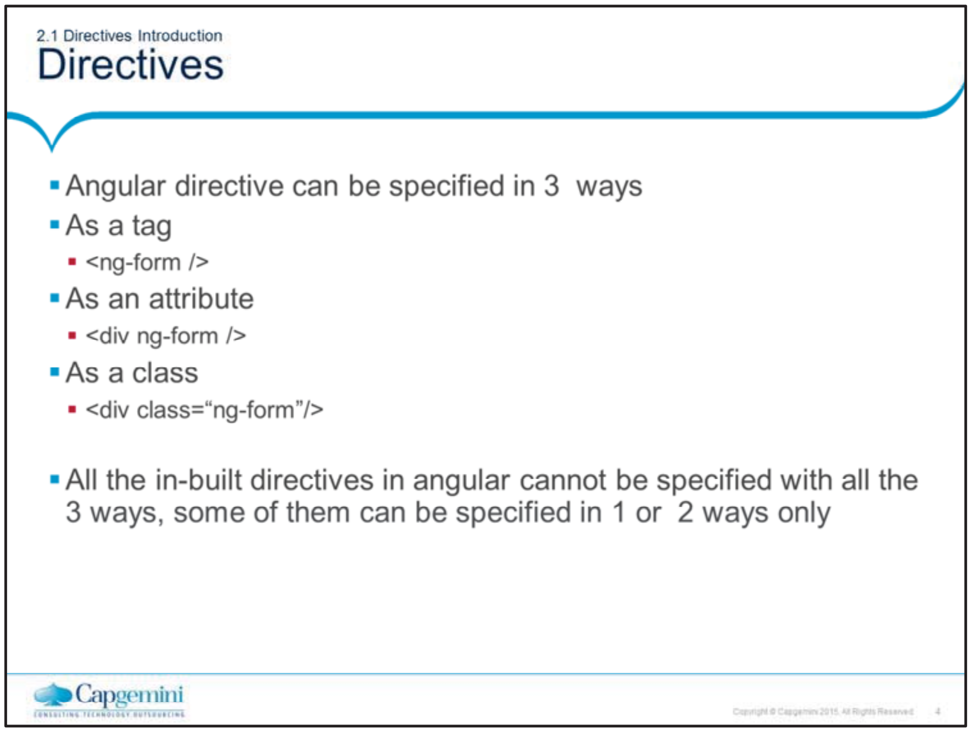
- Directives are ways to transform the DOM through extending HTML and provides new functionality to it
- As a good practice DOM manipulations need to be done in directives
- Directive is simply a function that we run on a particular DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler ($compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children
- Directives are defined with camelCase in the JavaScript, but applied with a dash to the HTML
- Angular comes with a set of built-in directives.
- It allows to create our own directives

Angular normalizes an element's tag and attribute name to determine which elements match which directives. We typically refer to directives by their case-sensitive camelCase normalized name (e.g. ngModel). However, since HTML is case-insensitive, we refer to directives in the DOM by lower-case forms, typically using dash-delimited attributes on DOM elements (e.g. ng-model).

The normalization process is as follows:

Strip x- and data- from the front of the element/attributes. Convert the :, -, or _-delimited name to camelCase. Here are some equivalent examples of elements that match ngBind:

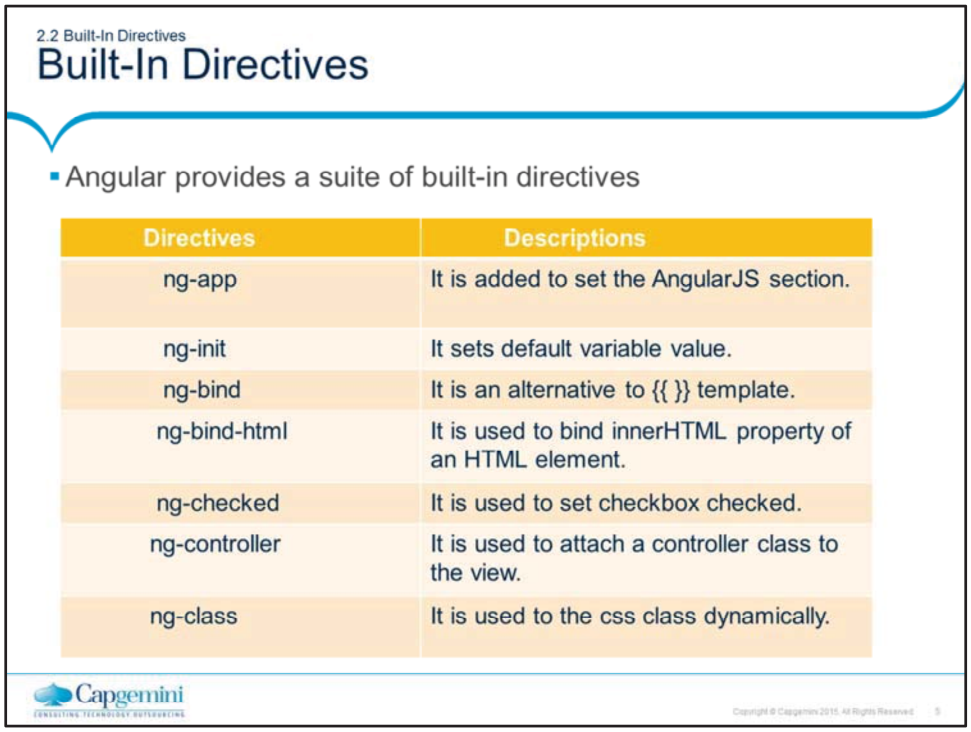
❑ng-bind
❑ng:bind
❑ng_bind
❑data-ng-bind
❑x-ng-bind



2.1 Directives Introduction

Directives

- Angular directive can be specified in 3 ways
- As a tag
  - <ng-form />
- As an attribute
  - <div ng-form />
- As a class
  - <div class="ng-form"/>
- All the in-built directives in angular cannot be specified with all the 3 ways, some of them can be specified in 1 or 2 ways only



2.2 Built-In Directives

Built-In Directives

- Angular provides a suite of built-in directives

Directives	Descriptions
ng-app	It is added to set the AngularJS section.
ng-init	It sets default variable value.
ng-bind	It is an alternative to {{ }} template.
ng-bind-html	It is used to bind innerHTML property of an HTML element.
ng-checked	It is used to set checkbox checked.
ng-controller	It is used to attach a controller class to the view.
ng-class	It is used to the css class dynamically.

Capgemini

ngApp

Placing ng-app on any DOM element marks that element as the beginning of the $rootScope.

$rootScope is the beginning of the scope chain, and all directives nested under the ng-app in your HTML inherit from it.

$rootScope can be accessed via the run method

Using $rootScope is like using global scope hence it is not a best practice.

We can use ng-app once per document.

ngInit

The ngInit directive is used to set up the state inside the scope of a directive when that directive is invoked.

It is a shortcut for using ng-bind without needing to create an element; therefore, it is most commonly used with inline text

ngBind

The ngBind attribute tells Angular to replace the text content of the specified HTML element with the value of a given expression and to update the text content when the value of that expression changes.

It is preferable to use ngBind instead of {{ expression }}. if a template is momentarily displayed by the browser in its raw state before Angular compiles it. Since ngBind is an element attribute, it makes the bindings invisible to the user while the page is loading.

ngBindHtml

Creates a binding with innerHTML so that the result of evaluating the expression into the current element comes in a secure way.

ngSanitize (angular-sanitize.js) need to be included as module's dependencies to evaluate in a secure way, else it will throw error “Attempting to use an unsafe value in a safe context”.

We can bypass sanitization by binding to an explicitly trusted value **via $sce.trustAsHtml.**

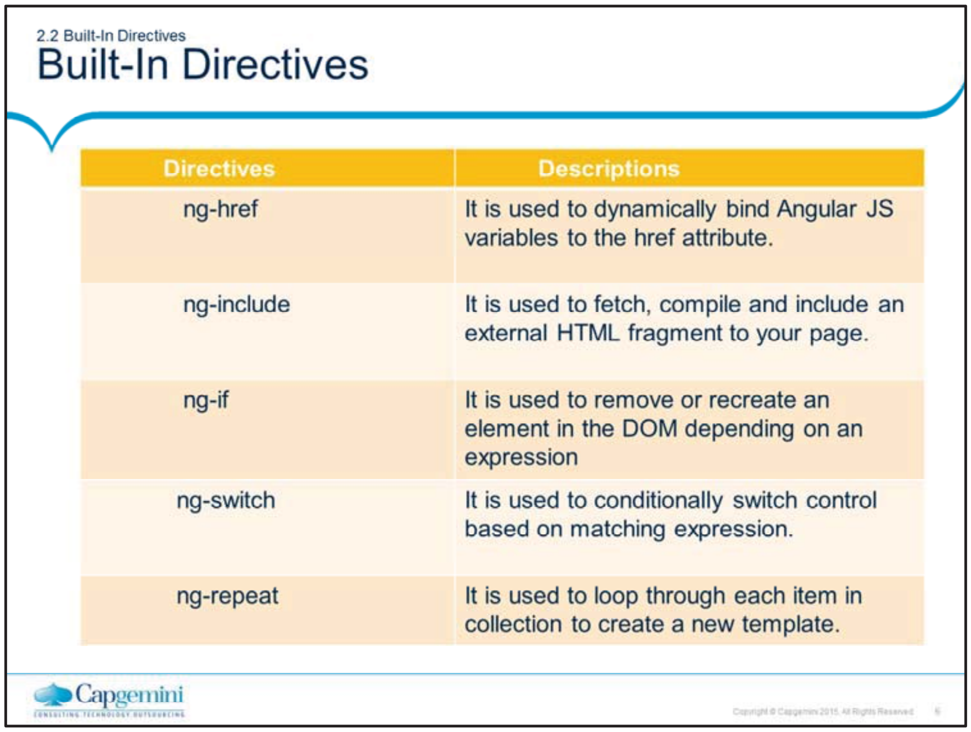
ngClass

ngClass directive allows you to dynamically set CSS classes on an HTML element by data binding an expression that represents all classes to be added.

ngController

This directive is used to place a controller on a DOM element.

Instead of defining actions and models on $rootScope, use ng-controller



2.2 Built-In Directives

Built-In Directives

Directives	Descriptions
ng-href	It is used to dynamically bind Angular JS variables to the href attribute.
ng-include	It is used to fetch, compile and include an external HTML fragment to your page.
ng-if	It is used to remove or recreate an element in the DOM depending on an expression
ng-switch	It is used to conditionally switch control based on matching expression.
ng-repeat	It is used to loop through each item in collection to create a new template.

ngHref

Angular waits for the interpolation to take place and then activates the link's behavior.

It is recommended to use in place of href.

ngInclude

ngInclude directive is used to fetch, compile and include an external HTML fragment into your current application.

By default, the template URL is restricted to the same domain and protocol as the application document. This is done by calling *$sce.getTrustedResourceUrl* on it

The URL of the template is restricted to the same domain and protocol as the application document unless white listed or wrapped as trusted values.

To access file locally in chrome use *chrome.exe -allow-file-access-from-files -disable-web-security*

ngIf

ng-if directive is used to completely remove or recreate an element in the DOM based on an expression. If the expression assigned to ng-if evaluates to a false value, then the element is removed from the DOM, otherwise a clone of the element is reinserted into the DOM.

Using ng-if when an element is removed from the DOM, its associated scope is destroyed. when it comes back into being, a new scope is created

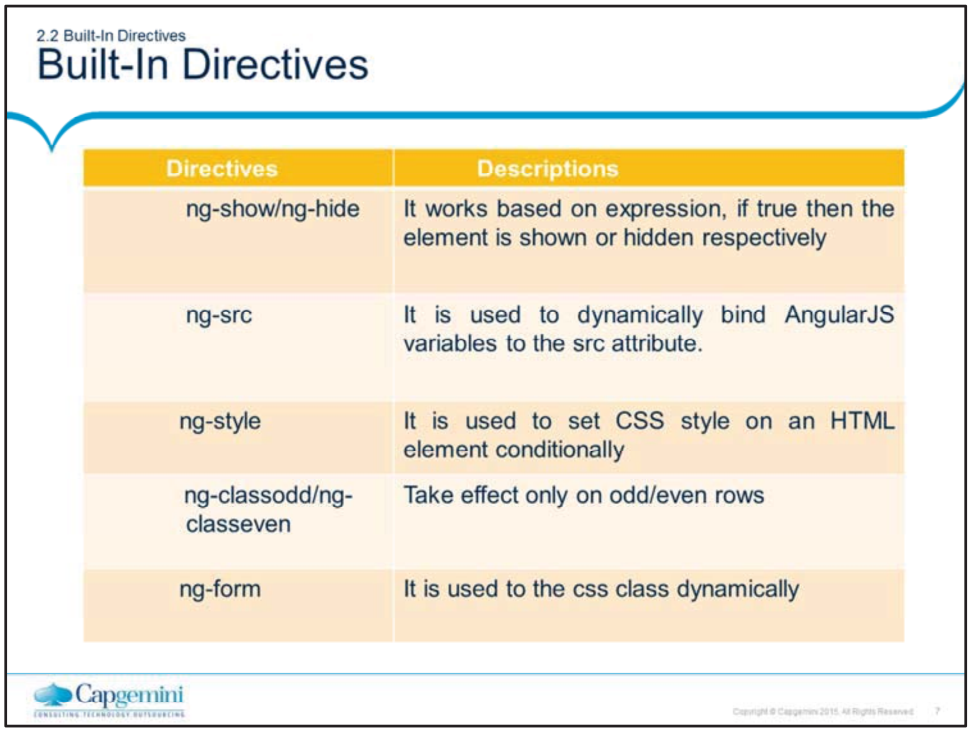
ngSwitch

The ngSwitch directive is used to conditionally swap DOM structure on your template based on a scope expression

It is used in conjunction with ng-switch-when and on="propertyName" to switch which directives render in our view when the given propertyName changes.

ngRepeat

ngRepeat directive instantiates a template once per item from a collection. Each template instance gets its own scope, where the given loop variable is set to the current collection item, and $index is set to the item index or key



2.2 Built-In Directives

Built-In Directives

Directives	Descriptions
ng-show/ng-hide	It works based on expression, if true then the element is shown or hidden respectively
ng-src	It is used to dynamically bind AngularJS variables to the src attribute.
ng-style	It is used to set CSS style on an HTML element conditionally
ng-classodd/ng-classeven	Take effect only on odd/even rows
ng-form	It is used to the css class dynamically

Capgemini CONSULTING. TECHNOLOGY. OUTSOURCING

ngShow

The ngShow directive shows or hides the given HTML element based on the expression provided to the ngShow attribute.

When the ngShow expression evaluates to a false value then the ng-hide CSS class is added to the class attribute on the element causing it to become hidden. When true, the ng-hide CSS class is removed from the element causing the element not to appear hidden.

ngHide

The ngHide directive shows or hides the given HTML element based on the expression provided to the ngHide attribute.

When the ngHide expression evaluates to a truthy value then the .ng-hide CSS class is added to the class attribute on the element causing it to become hidden. When falsy, the ng-hide CSS class is removed from the element causing the element not to appear hidden.

ngSrc

Angular will tell the browser not to fetch the image via the given URL until all expressions provided to ng-src have been interpolated.

It is recommended to use in place of src.

ngStyle

ngStyle directive allows you to set CSS style on an HTML element.

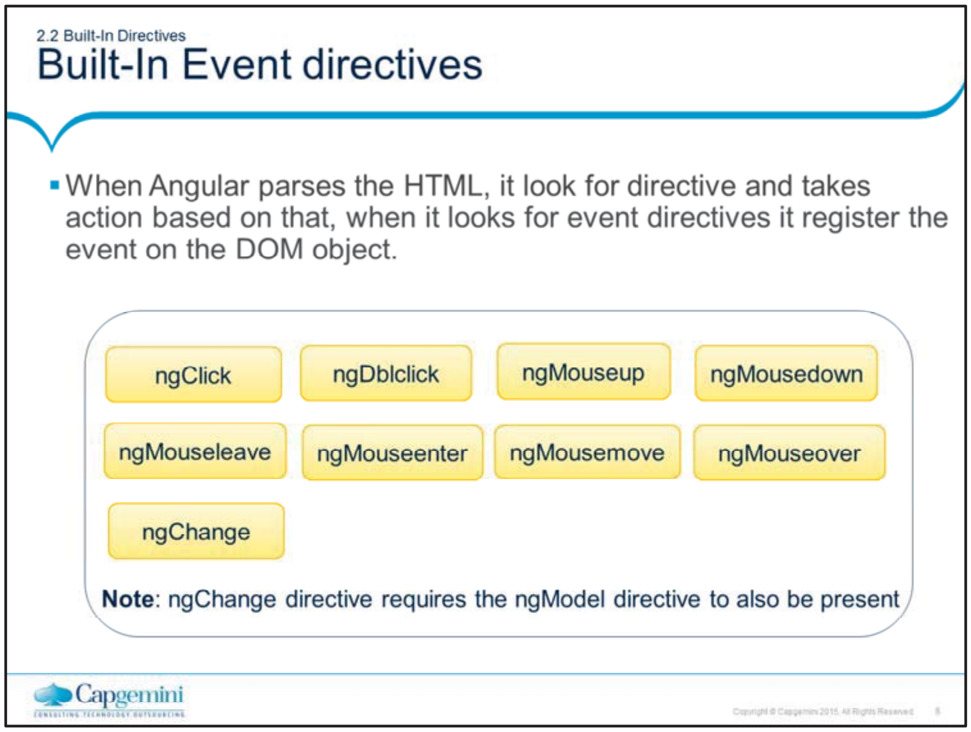
CSS style names and values must be quoted.

ngClassEven

Works exactly as ngClass, except they work in conjunction with ngRepeat and take effect only on even row elements i.e. 0,2,4,6 …

ngClassOddS

Works exactly as ngClass, except they work in conjunction with ngRepeat and take effect only on odd row elements i.e. 1,3,5,7…



2.2 Built-In Directives

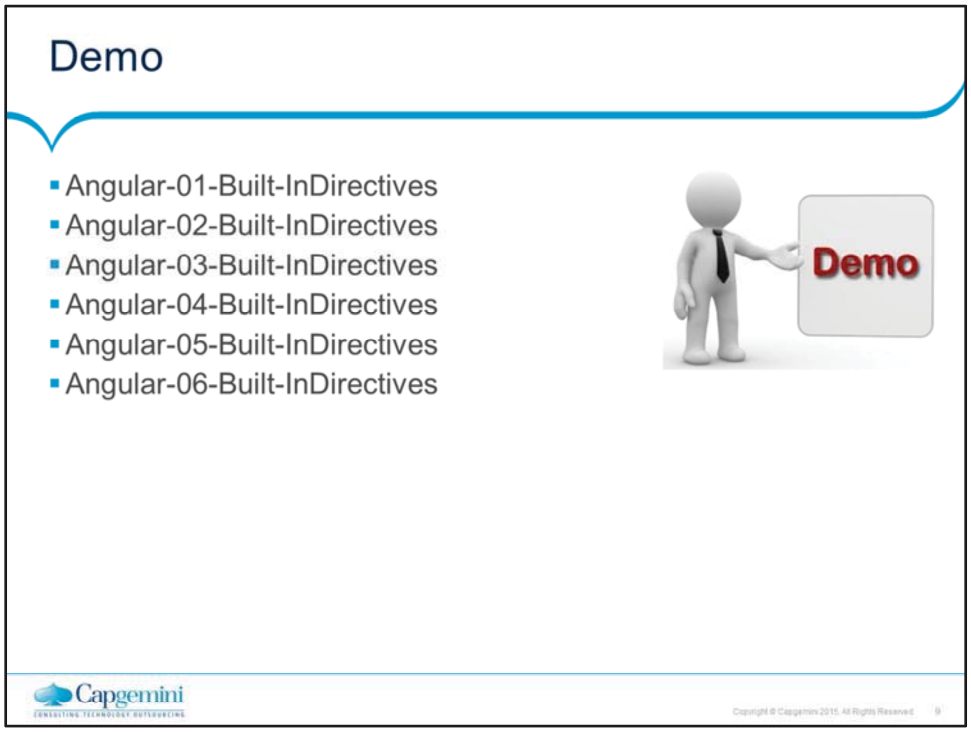
Built-In Event directives

- When Angular parses the HTML, it look for directive and takes action based on that, when it looks for event directives it register the event on the DOM object.

ngClick	ngDblclick	ngMouseup	ngMousedown
ngMouseleave	ngMouseenter	ngMousemove	ngMouseover
ngChange			

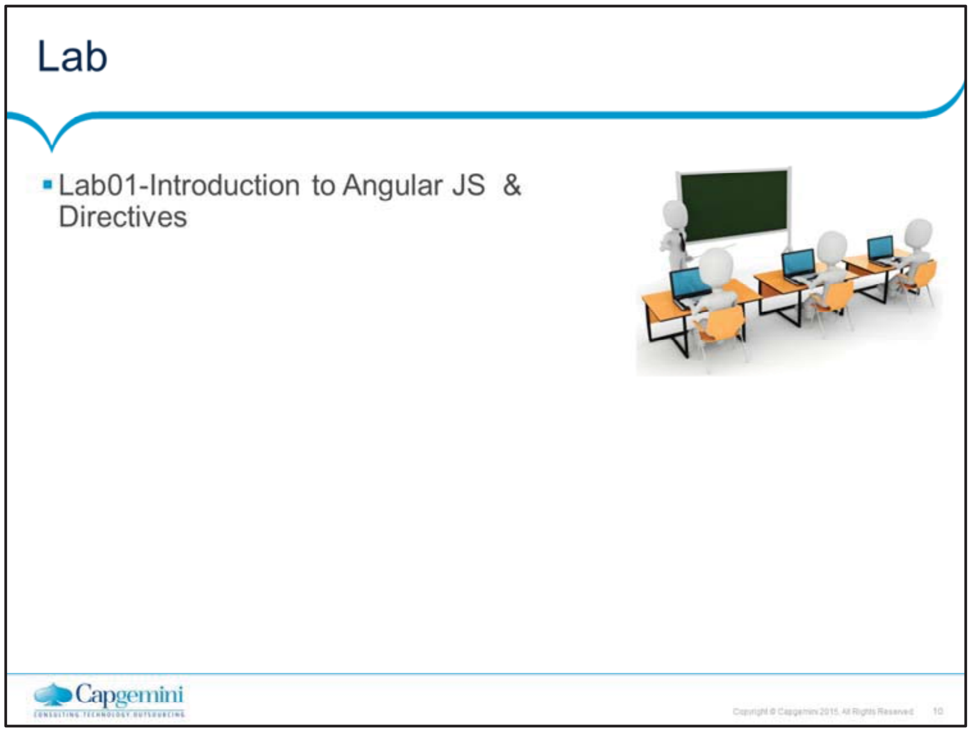
Note: ngChange directive requires the ngModel directive to also be present

ngCLick:The ngClick directive allows you to specify custom behavior when an element is clicked.
ngMousemove:Expression to evaluate upon mousemove. (Event object is available as $event)
ngMouseupSpecify custom behavior on mouseup event.



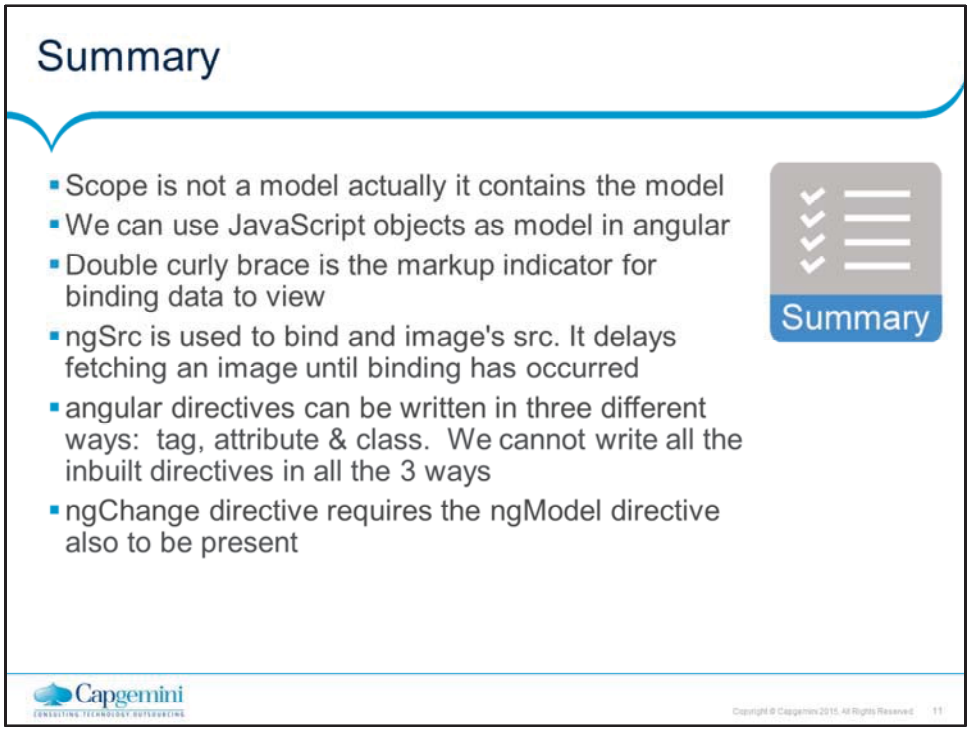
Demo

- Angular-01-Built-InDirectives
- Angular-02-Built-InDirectives
- Angular-03-Built-InDirectives
- Angular-04-Built-InDirectives
- Angular-05-Built-InDirectives
- Angular-06-Built-InDirectives



Lab

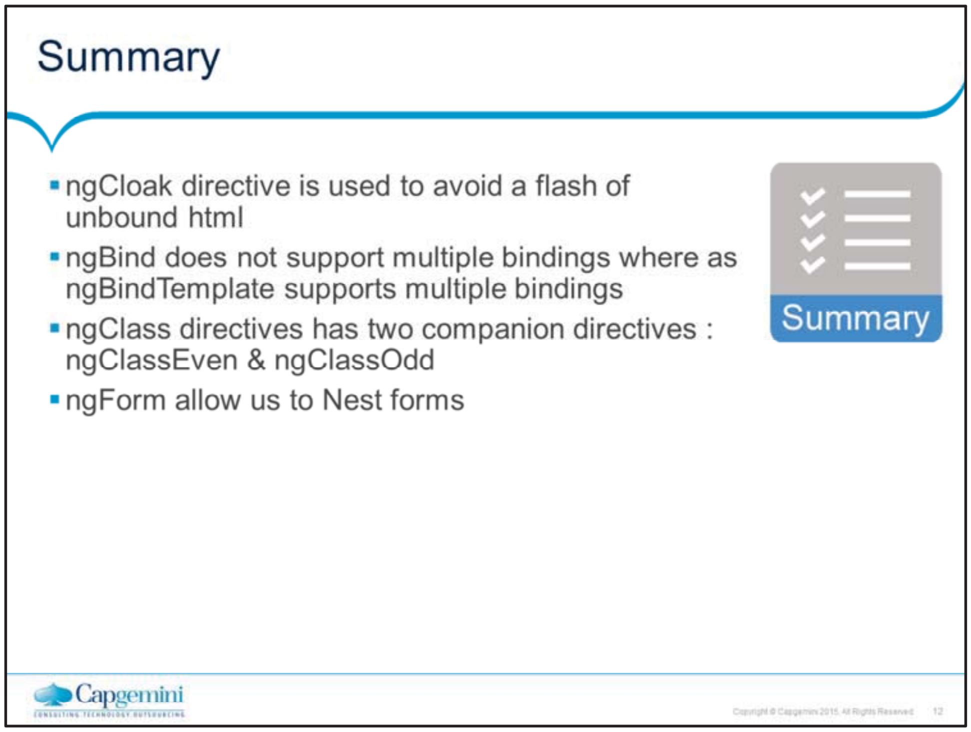
- Lab01-Introduction to Angular JS & Directives



Summary

- Scope is not a model actually it contains the model
- We can use JavaScript objects as model in angular
- Double curly brace is the markup indicator for binding data to view
- ngSrc is used to bind and image's src. It delays fetching an image until binding has occurred
- angular directives can be written in three different ways: tag, attribute & class. We cannot write all the inbuilt directives in all the 3 ways
- ngChange directive requires the ngModel directive also to be present

Add the notes here.

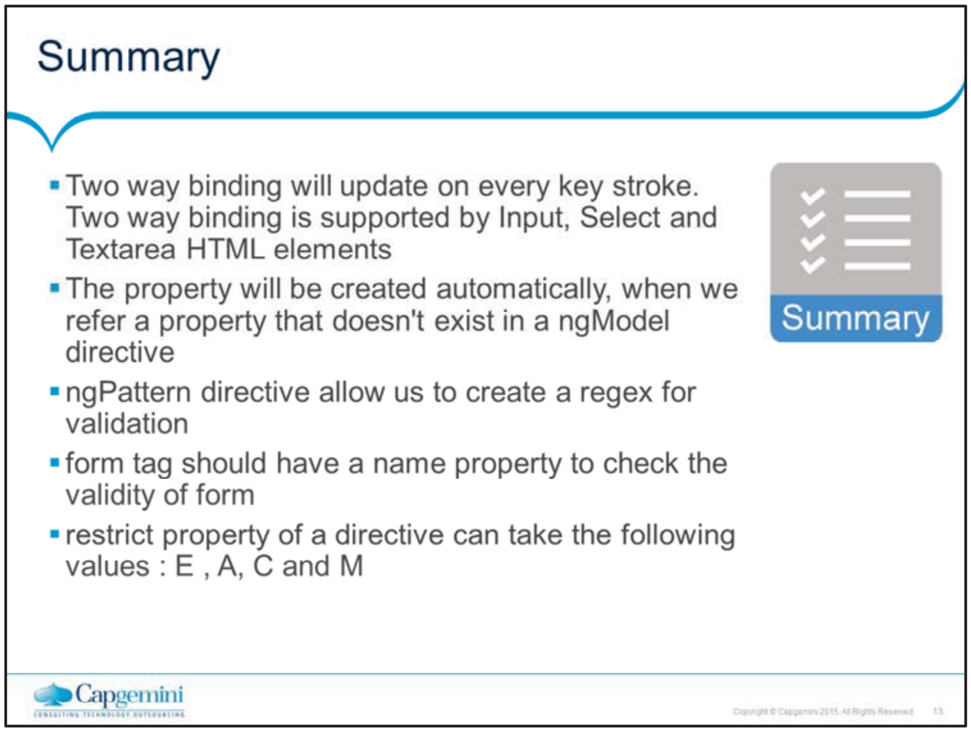


Summary

- ngCloak directive is used to avoid a flash of unbound html
- ngBind does not support multiple bindings where as ngBindTemplate supports multiple bindings
- ngClass directives has two companion directives : ngClassEven & ngClassOdd
- ngForm allow us to Nest forms

12

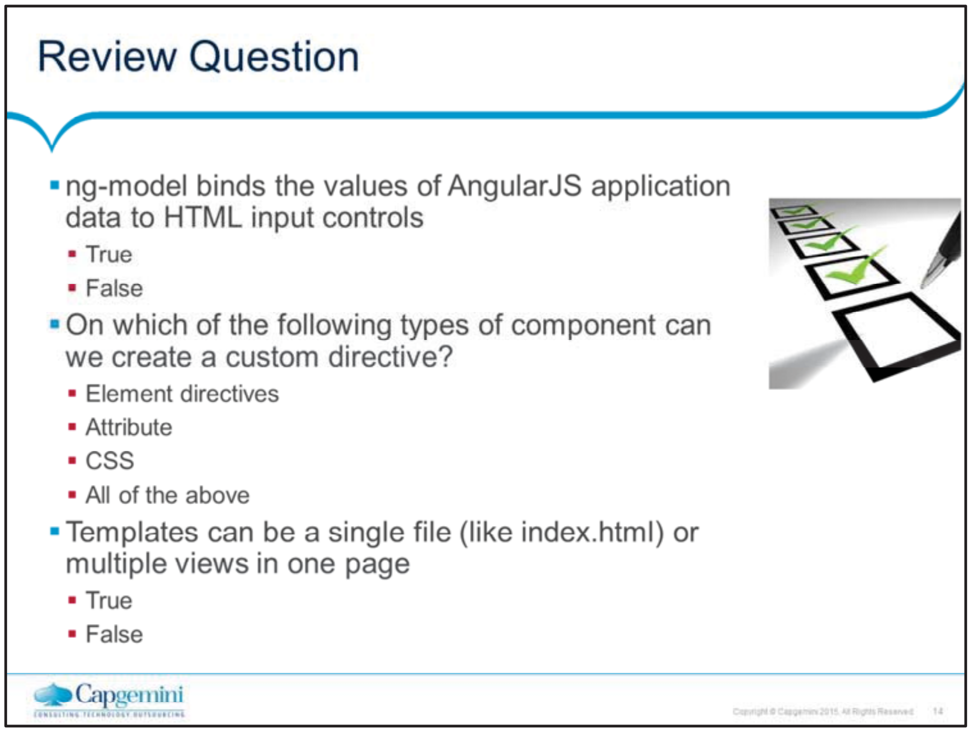
Add the notes here.



Summary

- Two way binding will update on every key stroke. Two way binding is supported by Input, Select and Textarea HTML elements
- The property will be created automatically, when we refer a property that doesn't exist in a ngModel directive
- ngPattern directive allow us to create a regex for validation
- form tag should have a name property to check the validity of form
- restrict property of a directive can take the following values : E , A, C and M

Add the notes here.

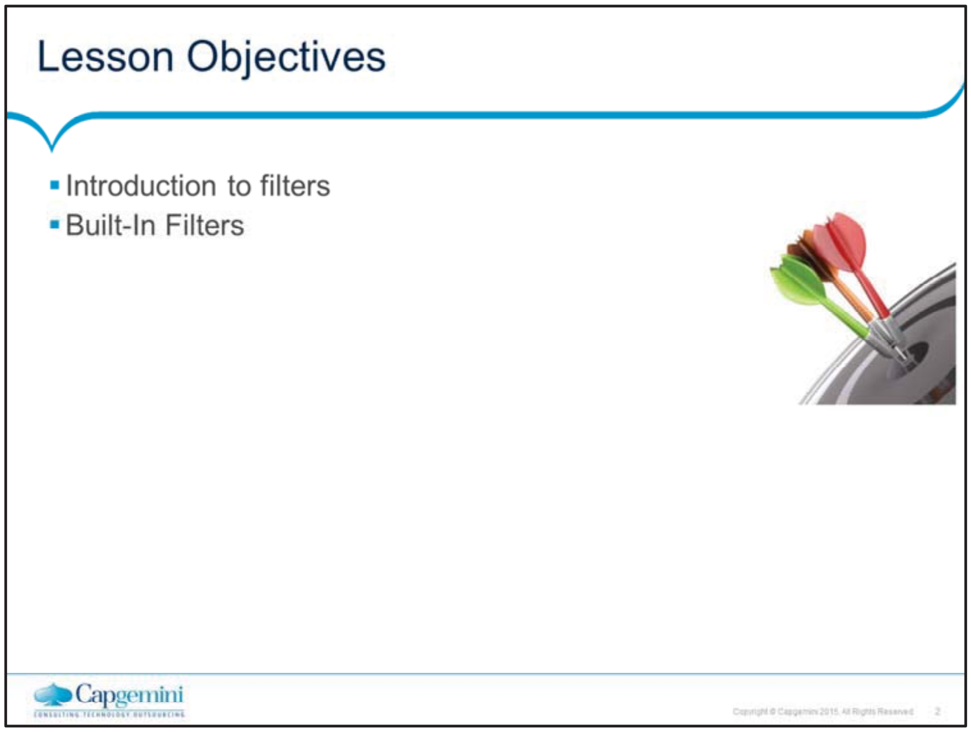


Review Question

- ng-model binds the values of AngularJS application data to HTML input controls
  - True
  - False
- On which of the following types of component can we create a custom directive?
  - Element directives
  - Attribute
  - CSS
  - All of the above
- Templates can be a single file (like index.html) or multiple views in one page
  - True
  - False

AngularJS

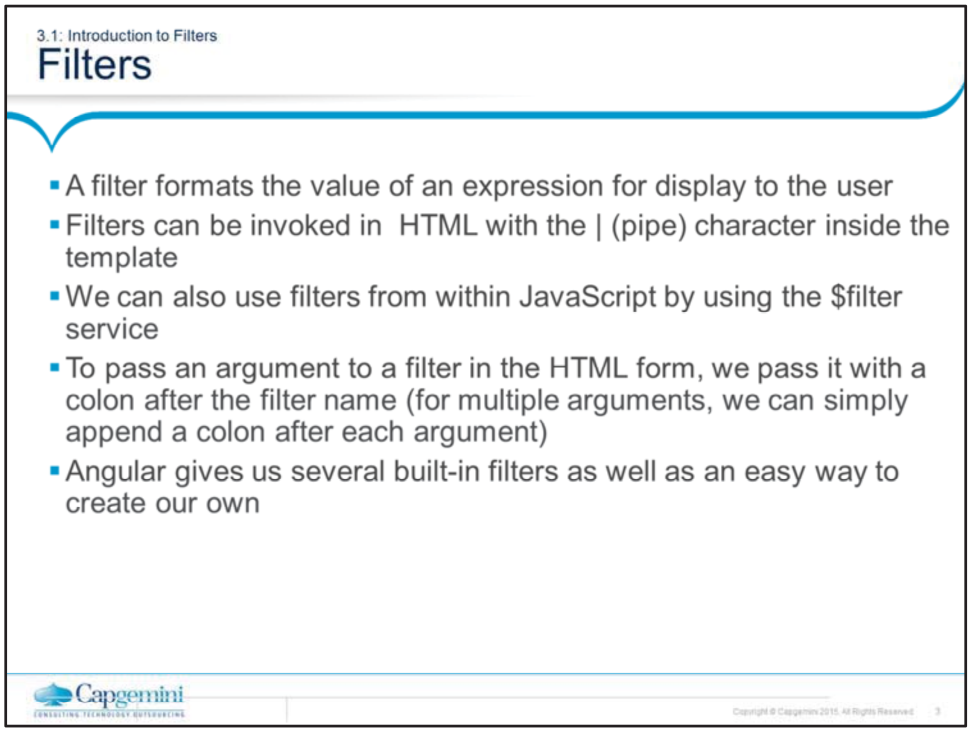
AngularJS Filters



Lesson Objectives

- Introduction to filters
- Built-In Filters

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



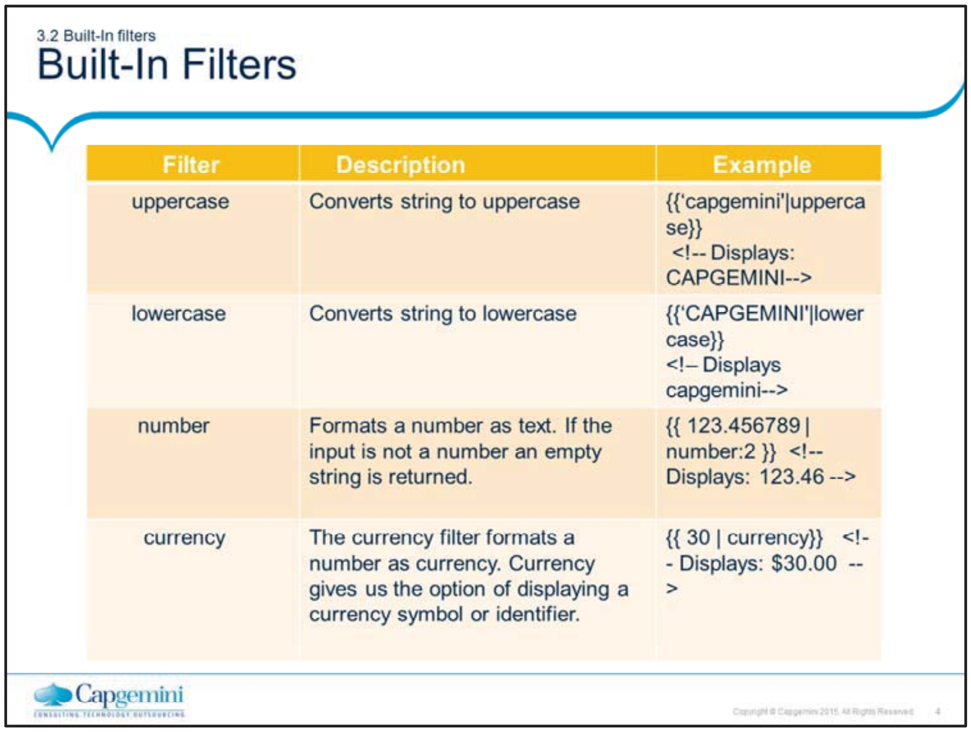
3.1: Introduction to Filters

Filters

- A filter formats the value of an expression for display to the user
- Filters can be invoked in HTML with the | (pipe) character inside the template
- We can also use filters from within JavaScript by using the $filter service
- To pass an argument to a filter in the HTML form, we pass it with a colon after the filter name (for multiple arguments, we can simply append a colon after each argument)
- Angular gives us several built-in filters as well as an easy way to create our own

A filter formats the value of an expression for display to the user. They can be used in view templates, controllers or services and it is easy to define your own filter.
Filters can be applied to expressions in view templates using the following syntax:

```
{{ expression | filter }}
{{ expression | filter1 | filter2 | ... }}
{{ expression | filter:argument1:argument2:... }}
```



3.2 Built-In filters

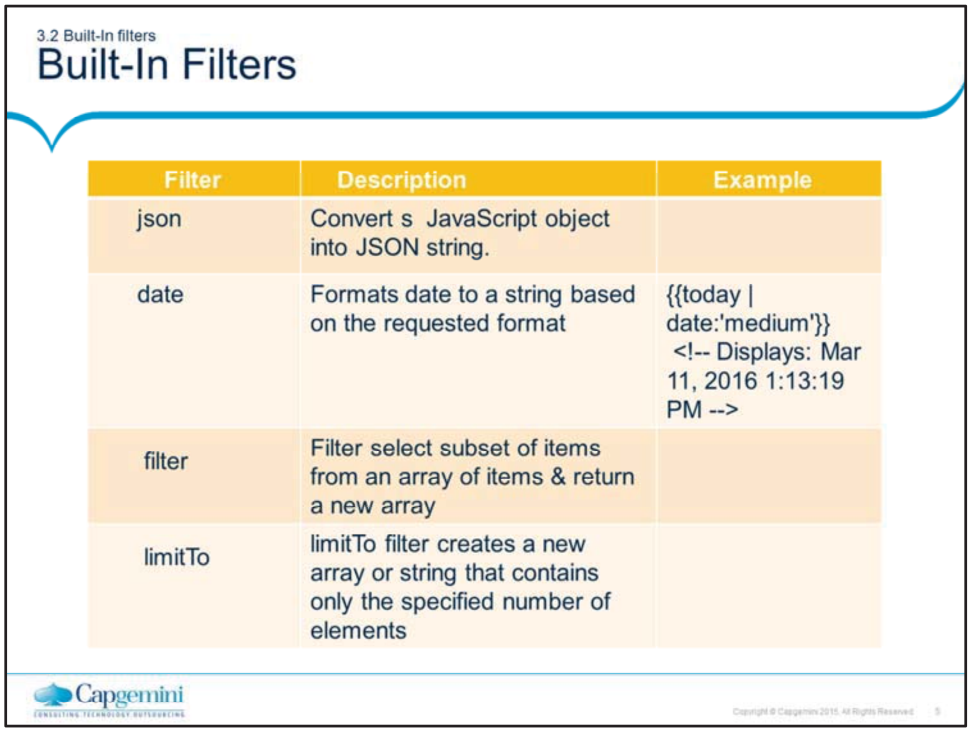
Built-In Filters

Filter	Description	Example
uppercase	Converts string to uppercase	{{'capgemini'|uppercase}} <!-- Displays: CAPGEMINI-->
lowercase	Converts string to lowercase	{{'CAPGEMINI'|lowercase}} <!– Displays capgemini-->
number	Formats a number as text. If the input is not a number an empty string is returned.	{{ 123.456789 | number:2 }} <!-- Displays: 123.46 -->
currency	The currency filter formats a number as currency. Currency gives us the option of displaying a currency symbol or identifier.	{{ 30 | currency}} <!-- Displays: $30.00 -->

Currency :The default currency option is that of the current locale we can also pass in a currency to display.

{{ 30 | currency}} <!-- Displays: $30.00 -->

{{ 30 | currency : "Rs."}} <!-- Displays: Rs.30.00 -->



3.2 Built-In filters

Built-In Filters

Filter	Description	Example
json	Convert s JavaScript object into JSON string.	
date	Formats date to a string based on the requested format	{{today | date:'medium'}} <!-- Displays: Mar 11, 2016 1:13:19 PM -->
filter	Filter select subset of items from an array of items & return a new array	
limitTo	limitTo filter creates a new array or string that contains only the specified number of elements	

Json Filter This filter is mostly useful for debugging

{{{ 'Id':714709,'Name':'ABCD'} | json }}

<!-- Displays: { "Id": 714709, "Name": “ABCD"} -->

date Filter

Date to be formatted can be either a Date object or milliseconds (string or number)

The date formatter provides us several built-in options. If no date format is passed, then it set to the default mediumDate.

```
{{today | date:'medium'}} <!-- Displays: Jul 30, 2014 5:13:19 PM -->
{{today | date:'short'}} <!-- Displays: 7/30/14 5:13 PM -->
{{today | date:‘fullDate'}} <!-- Displays: Wednesday, July 30, 2014-->
{{today | date:‘longDate'}} <!-- Displays: July 30, 2014-->
{{today | date:'mediumDate'}} <!-- Displays: Jul 30, 2014-->
{{today | date:‘shortDate'}} <!-- Displays: 7/30/14 -->
{{today | date:‘mediumTime'}} <!-- Displays: 5:22:21 PM -->
{{today | date:‘mediumTime'}} <!-- Displays: 5:23 PM -->
{{today | date:'d-M-y'}}  <!-- Displays: 30-7-2014 -->
{{today | date:'d-M-yyyy'}} <!-- Displays: 30-7-2014 -->
{{today | date:'dd-MM-yy'}} <!-- Displays: 30-07-14 -->
{{today | date:'EEEE dd, MMMM yyyy'}}  <!-- Displays: Wednesday 30, July 2014 -->
{{today | date:'EEE dd MMM yyyy'}}  <!-- Displays: Wed 30 Jul 2014 -->
{{ today | date:'hh:mm:ss.sss a' }}  <!-- Displays: 05:35:31.951 PM  -->
{{ today | date:'hh:mm:ss a' }}   <!-- Displays: 05:35:31 PM  -->
{{ today | date:'H:m:s a' }} <!-- Displays: 17:35:31 PM  -->
{{ today | date:'Z' }}  <!-- Displays: +0530 -->
```

```
<div >
<span><b>Printing Data FOR FILTER: </b></span>
<span ng-repeat="product in products | filter:1000 ">{{product.name }}</span>
</div>
```

Example2:

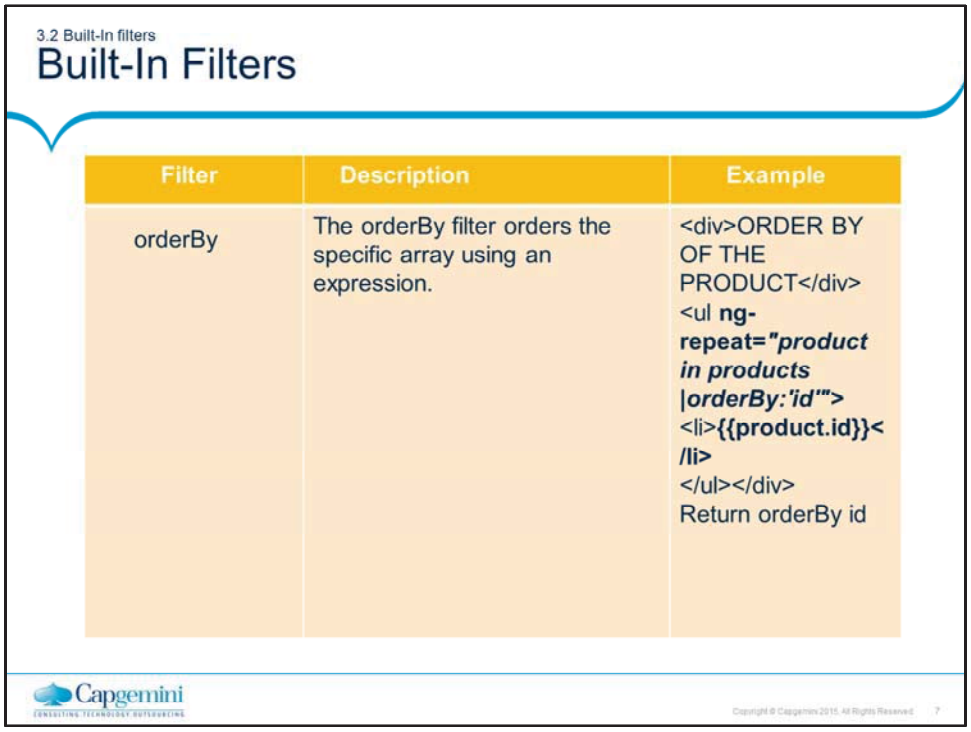
filter based on a string **:**
```
        {{ ['Anil', 'Latha', 'Mahima', 'Sachin', 'Veena'] | filter:'e' }}
        <!-- returns array ["Veena"]        !e' returns other than 'Veena'     -->
```

filter based on a function:
```
        angular.module('filterApp',[])
        .controller('FilterController', function($scope,$filter) {
                        var pattern = /^\d{6}$/;
                        $scope.getSixDigitsPattern = function(item) {
                                        return pattern.test(item);
                        }
        });
{{ ['714709', '562A', '044-235', '801234','ABC'] | filter:getSixDigitsPattern }}
<!-- return array ["714709","801234"] -->
```

filter based on a object:
```
        {{
            [
                        {"Id":1,"Name":"Anil","Location":"Mumbai"},
                                        {"Id":2,"Name":"Latha","Location":"Bangalore"},

                        {"Id":3,"Name":"Mahima","Location":"Pune"},
                                        {"Id":4,"Name":"Sachin","Location":"Mumbai"},

                        {"Id":5,"Name":"Veena","Location":"Pune"}
                ]  | filter:{"Location":"Mumbai"}
        }}
        <!-- returns array
        [
                {"Id":1,"Name":"Anil","Location":"Mumbai"},
                {"Id":4,"Name":"Sachin","Location":"Mumbai"}
        ]
        -->
```

limitTo:
limitTo filter creates a new array or string that contains only the specified number of elements, either taken from the beginning or end, depending on whether the value is positive or negative.

```
{{ 'CAPGEMINI' | limitTo:5 }}
<!-- returns  first 5 characters: CAPGE -->

{{ 'CAPGEMINI INDIA LEARNING' | limitTo:-8}}
<!-- returns  last 8 characters: LEARNING' -->

{{
         ['Bangalore','Chennai','Hyderabad','Gandhinagar','Mumabai','Noida','Pune']
        | limitTo:2
}}
<!--  returns  first 2 array elements :  ["Bangalore","Chennai"] -->
```



3.2 Built-In filters

Built-In Filters

Filter	Description	Example
orderBy	The orderBy filter orders the specific array using an expression.	<div>ORDER BY OF THE PRODUCT</div> <ul **ng-repeat=*"product in products |orderBy:'id'"*>** <li>**{{product.id}}</li>** </ul></div> Return orderBy id

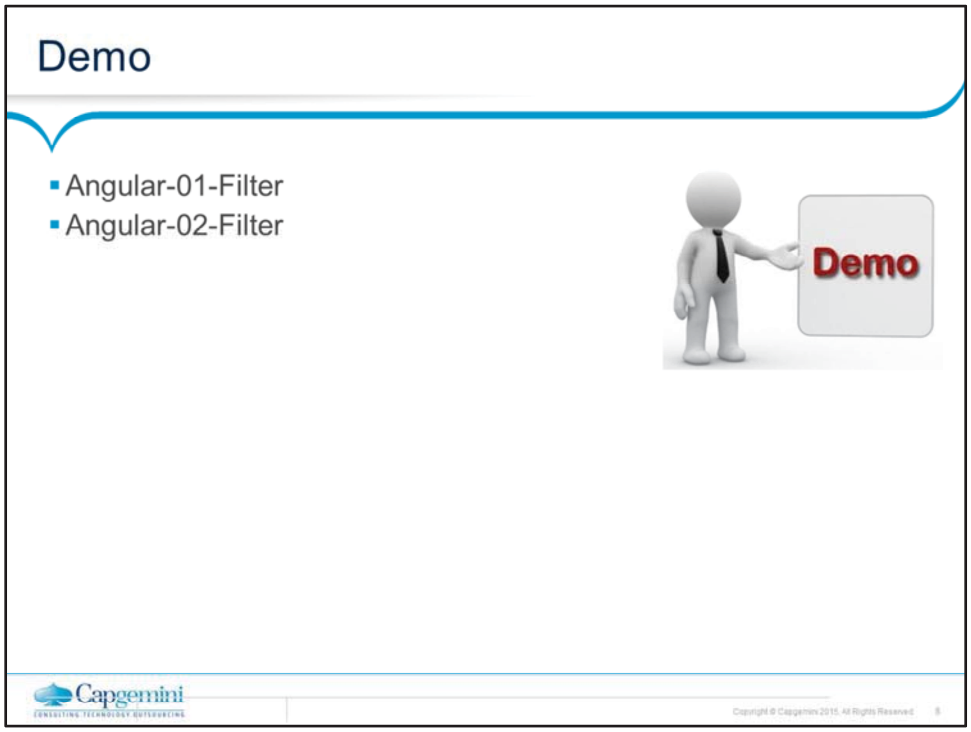
Capgemini

orderBy:

Orders alphabetically for strings and numerically for numbers.

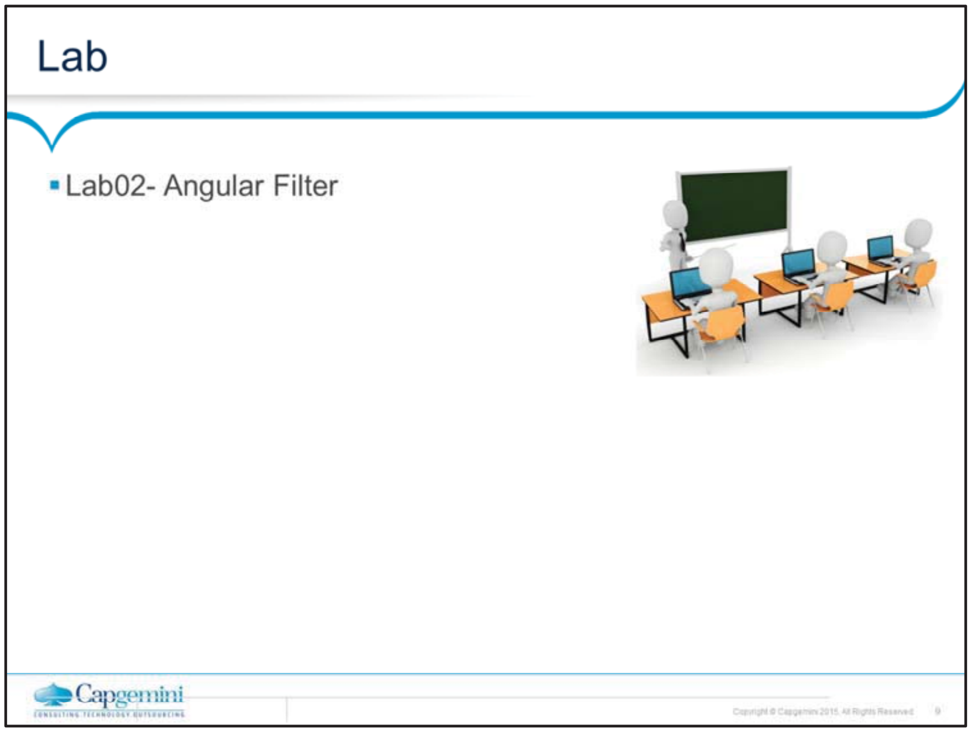
It takes 2 parameters. First parameter is the predicate used to determine the order of the sorted array. Second parameter(optional) is a boolean value, if it is true it will sort the data in reverse order.

```
{{['Chennai','Bangalore','Pune','Mumbai'] | orderBy:'toString()'}}
<!-- returns ["Bangalore","Chennai","Mumbai","Pune"] -->


{{[{'Id':1,'Location':'Bangalore'},{'Id':2,'Location':'Chennai'}] |
orderBy:'Id':true}}
<!-- returns
[{"Id":2,"Location":"Chennai"},{"Id":1,"Location":"Bangalore"}] -->
```



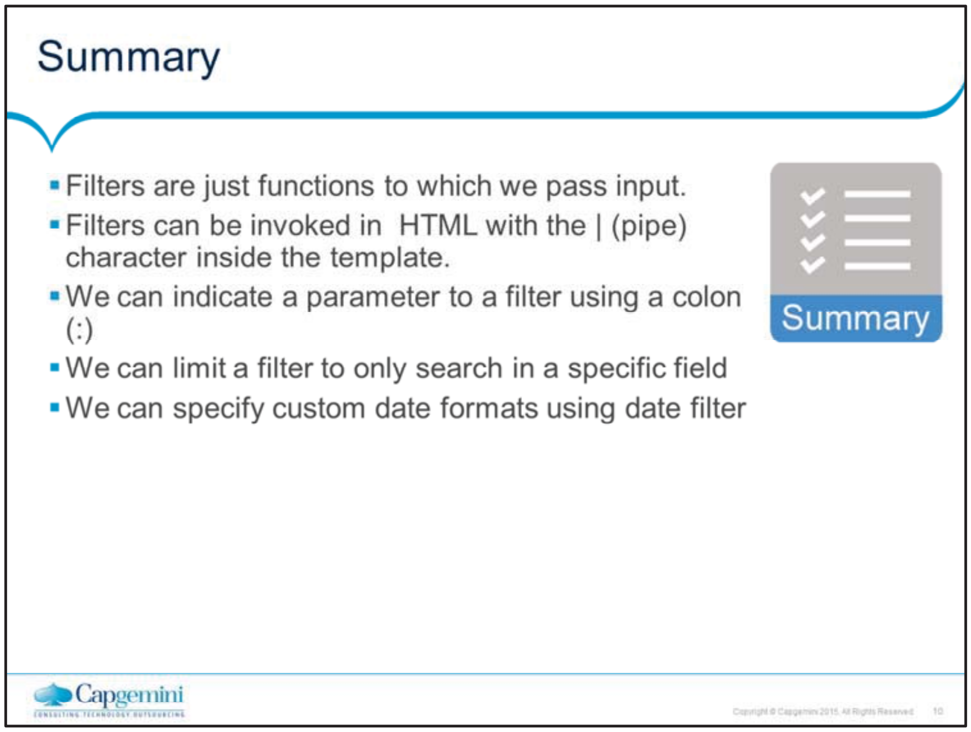
Demo

- Angular-01-Filter
- Angular-02-Filter



Lab

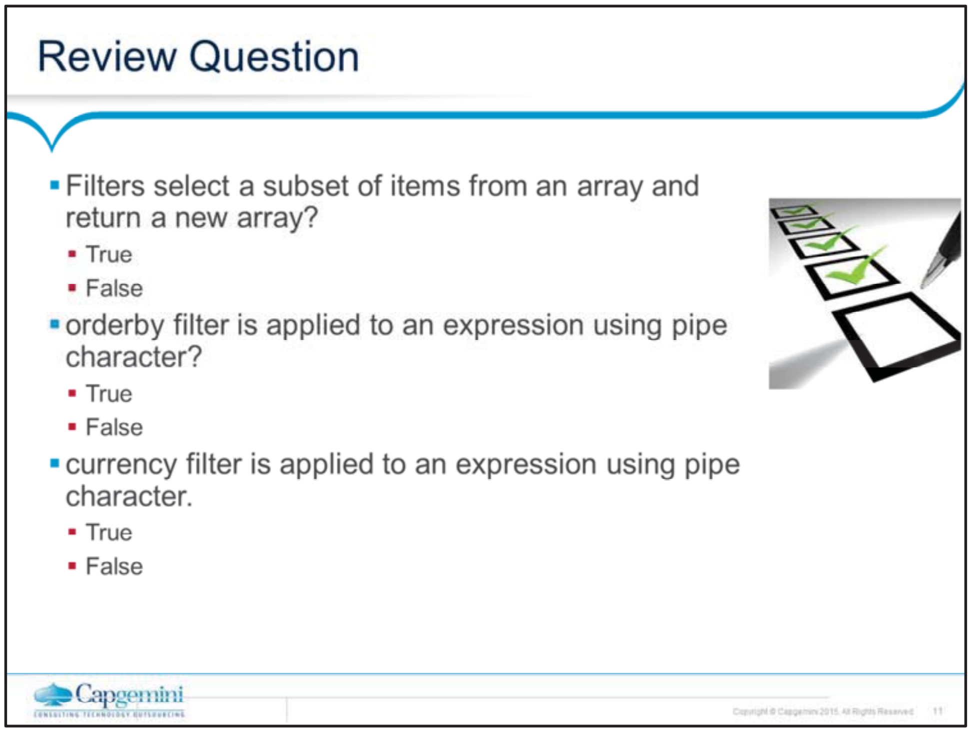
- Lab02- Angular Filter



Summary

- Filters are just functions to which we pass input.
- Filters can be invoked in HTML with the | (pipe) character inside the template.
- We can indicate a parameter to a filter using a colon (:)
- We can limit a filter to only search in a specific field
- We can specify custom date formats using date filter

Add the notes here.



Review Question

- Filters select a subset of items from an array and return a new array?
  - True
  - False
- orderby filter is applied to an expression using pipe character?
  - True
  - False
- currency filter is applied to an expression using pipe character.
  - True
  - False

Angular JS Lab Book

Document Revision History

Date	Revision No.	Author	Summary of Changes
May 2016	1.2	Rahul Vikash	Created new lab book as per revised course contents

Table of Contents

Getting Started

Overview

This lab book is a guided tour for learning Angular JS version 1.2 and above. It comprises 'To Do' assignments. Follow the steps provided to work out the 'To Do' assignments given.

Setup Checklist for AngularJS

Here's what is expected on your machine for the lab in order to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows XP, Windows 7 or Windows 8
- Memory: 2GB of RAM (1GB or more recommended)
- Google Chrome 36.0 or Mozilla Firefox 31.0 or Internet Explorer 10 or above

Please ensure that the following is done:

- A text editor like Notepad or Notepad++ or Eclipse Luna is installed.

Instructions

- Create a directory by your name in drive <drive>. In this directory, create a subdirectory angular_assgn. For each lab exercise create a directory as lab <lab number>.
- You may also look up the on-line help provided in https://docs.angularjs.org/api

Learning More

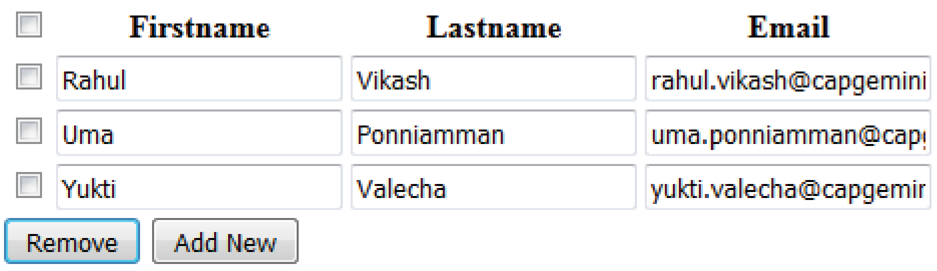
- O'REILLY AngularJS by Brad Green & Shyam Seshadri
- APress Pro AngularJS by Adam Freeman
- PACKT Publishing Instant AngularJS Starter by Dan Menard
- PACKT Publishing Dependency Injection with AngularJS by Alex Knol
- PACKT Publishing AngularJS Directives by Alex Vanston

Lab 1. Introduction to Angular JS & Directives

Goals	▪ Understand the process of creating an angular webpage and view in a browser window. ▪ Understanding the fundamentals of Angularjs (model, view, controller, modules, rootscope, Directives, events)
Time	60 minutes

1: Creating a webpage to work with Angular Basic & Directives

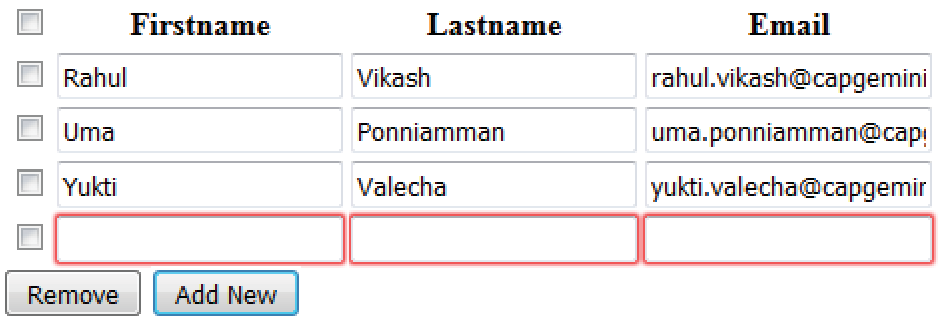
Create a web page and display the results shown below.



☐	Firstname	Lastname	Email
☐	Rahul	Vikash	rahul.vikash@capgemini
☐	Uma	Ponniamman	uma.ponniamman@cap
☐	Yukti	Valecha	yukti.valecha@capgemir

Figure 1

The rows could be added with help of 'Add New' as per the need. The rows are to be provided with the check boxes to facilitate the deletion process e.g., if any new row for the data entry is required, the Add New button is to be clicked on and the new row should appear.



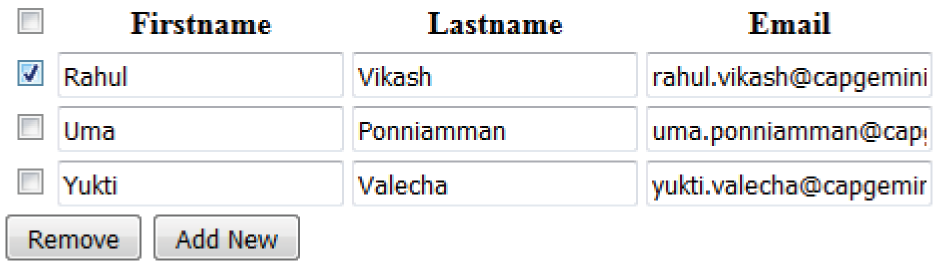
☐	Firstname	Lastname	Email
☐	Rahul	Vikash	rahul.vikash@capgemini
☐	Uma	Ponniamman	uma.ponniamman@cap
☐	Yukti	Valecha	yukti.valecha@capgemir
☐			

Figure 2

Note:

All fields are mandatory

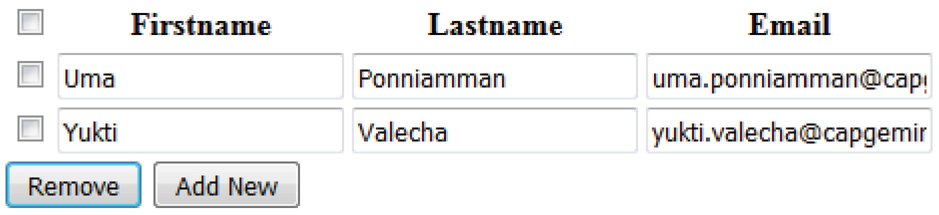
For deleting a row just check the check box and click on Remove button, and the row should get deleted. If we select topmost checkbox all the rows will be selected & when Remove button is clicked, all the rows will get deleted.



☐	Firstname	Lastname	Email
✓	Rahul	Vikash	rahul.vikash@capgemini
☐	Uma	Ponniamman	uma.ponniamman@cap
☐	Yukti	Valecha	yukti.valecha@capgemir

Figure 3

Removed Row



☐	Firstname	Lastname	Email
☐	Uma	Ponniamman	uma.ponniamman@cap
☐	Yukti	Valecha	yukti.valecha@capgemir

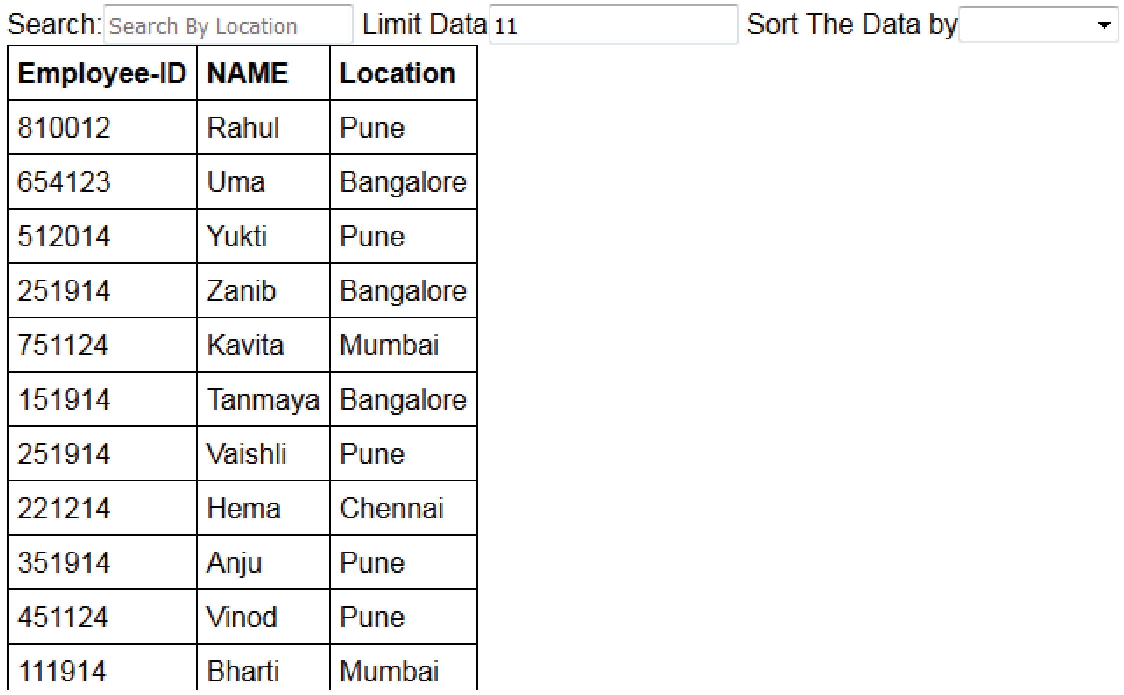
Figure 4

Lab 2. Angular Filters

Goals	▪ Working with inbuilt filters & creating Custom filter
Time	30 minutes

2. Creating a webpage to work with angular inbuilt-filters

Create a web page and display the results shown below



Search: Search By Location Limit Data 11 Sort The Data by

Employee-ID	NAME	Location
810012	Rahul	Pune
654123	Uma	Bangalore
512014	Yukti	Pune
251914	Zanib	Bangalore
751124	Kavita	Mumbai
151914	Tanmaya	Bangalore
251914	Vaishli	Pune
221214	Hema	Chennai
351914	Anju	Pune
451124	Vinod	Pune
111914	Bharti	Mumbai

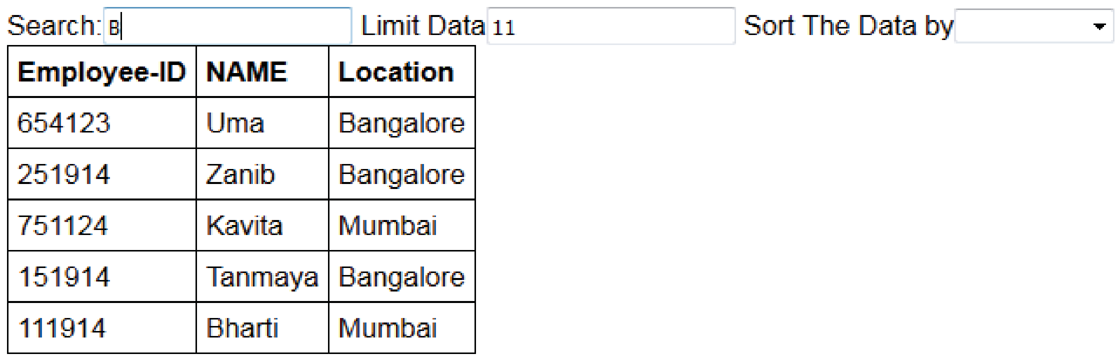
Figure 5

The data consists of 11 rows and 3 columns with the fields employee-ID, Name and Location.

Refer Below txt File

Lab02.txt

In the **Search field** we can search by Location such as, having letter 'B' anywhere in the string Location field. Result should appear as e.g, **'B'**angalore and Mum**'b'**ai.

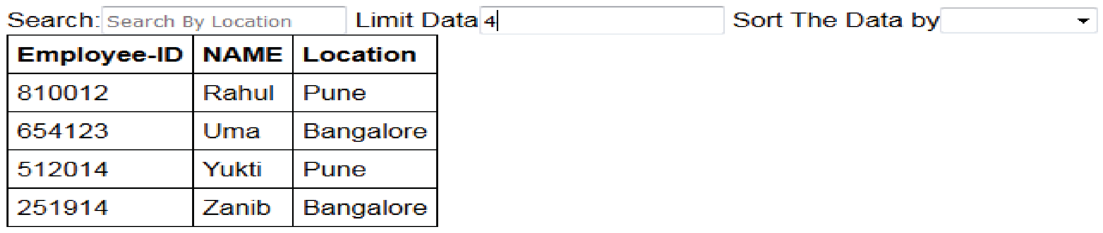


Search: B Limit Data 11 Sort The Data by

Employee-ID	NAME	Location
654123	Uma	Bangalore
251914	Zanib	Bangalore
751124	Kavita	Mumbai
151914	Tanmaya	Bangalore
111914	Bharti	Mumbai

Figure 6

In the **Limit field** we can limit the data eg how many number of rows we want .If we put 4 only first 4 rows will be displayed.

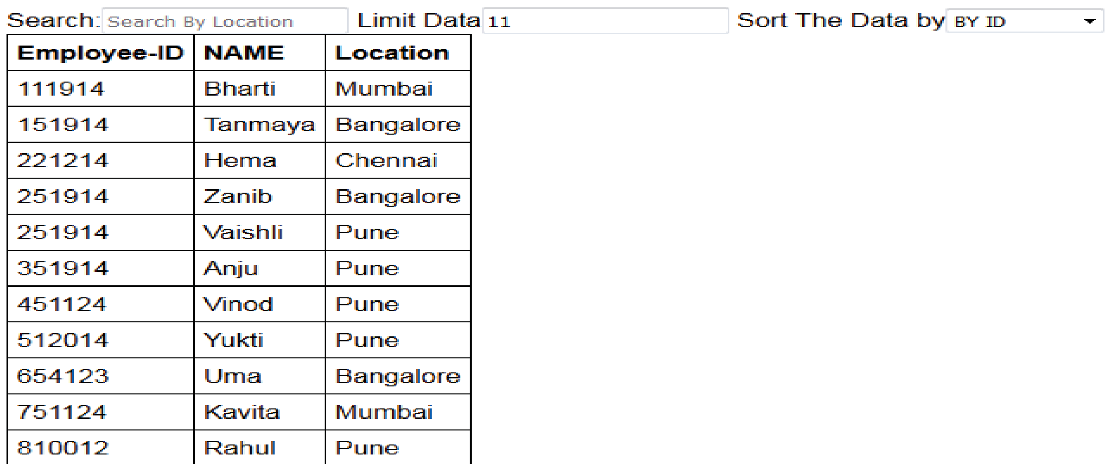


Search: Search By Location Limit Data 4 Sort The Data by

Employee-ID	NAME	Location
810012	Rahul	Pune
654123	Uma	Bangalore
512014	Yukti	Pune
251914	Zanib	Bangalore

Figure 7

In the **Sort the Data by field** there are 3 options "**By employee-id**" ,"**By Name**", "**By Location**", We can sort the data in ascending order by employeeid or by name or location eg sort by employee Id



Search: Search By Location Limit Data 11 Sort The Data by BY ID

Employee-ID	NAME	Location
111914	Bharti	Mumbai
151914	Tanmaya	Bangalore
221214	Hema	Chennai
251914	Zanib	Bangalore
251914	Vaishli	Pune
351914	Anju	Pune
451124	Vinod	Pune
512014	Yukti	Pune
654123	Uma	Bangalore
751124	Kavita	Mumbai
810012	Rahul	Pune

Figure 8

Appendix A: Table of Figures