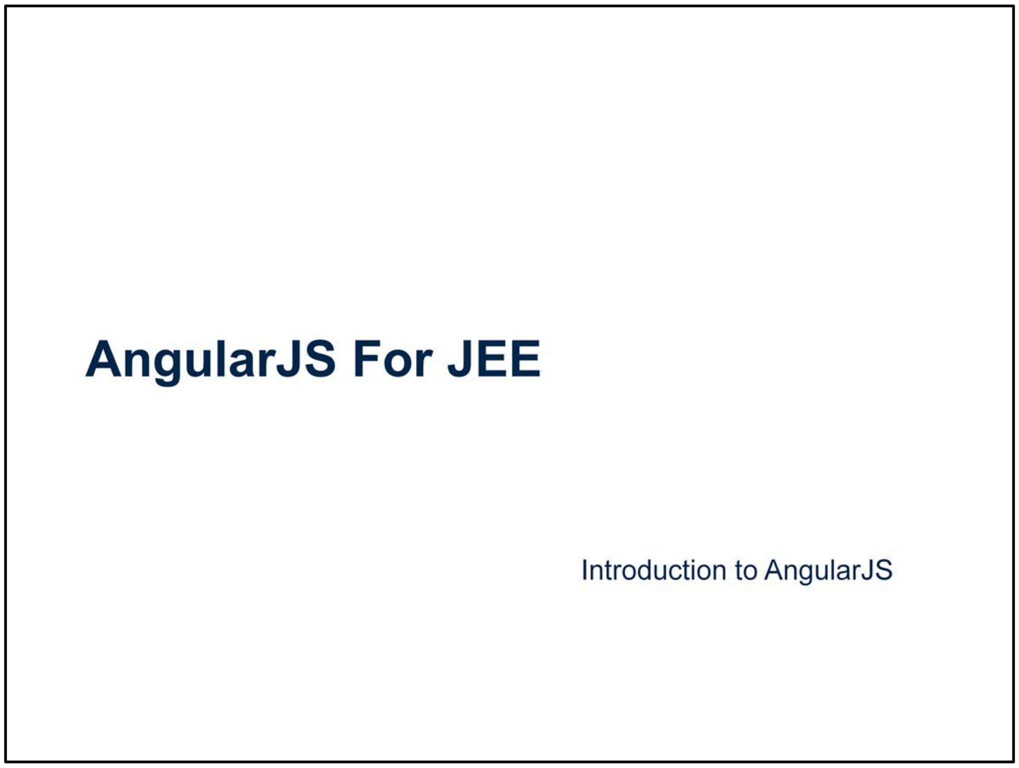
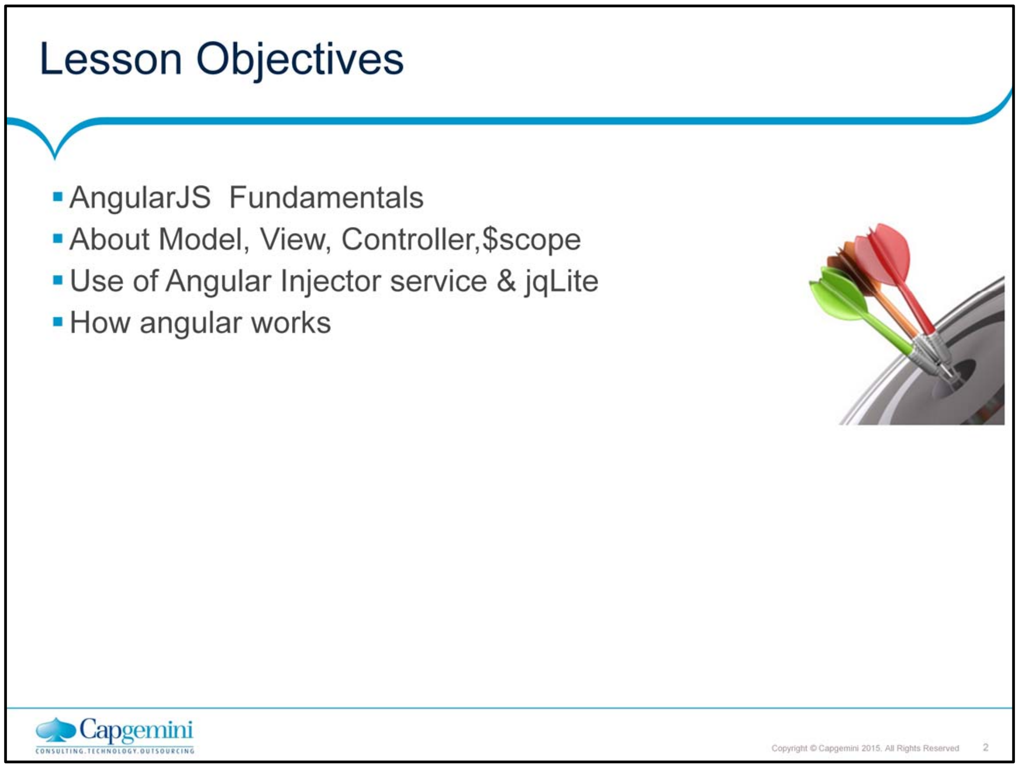


AngularJS For JEE

Introduction to AngularJS

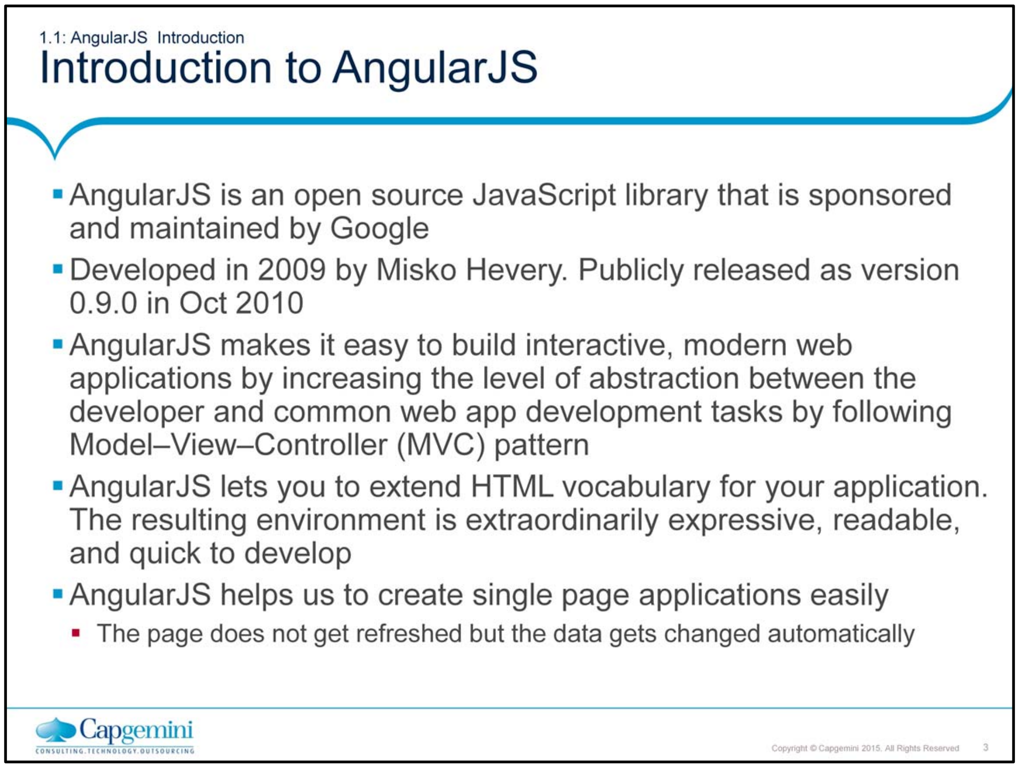


Lesson Objectives

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

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

2



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 CONSULTING.TECHNOLOGY.OUTSOURCING

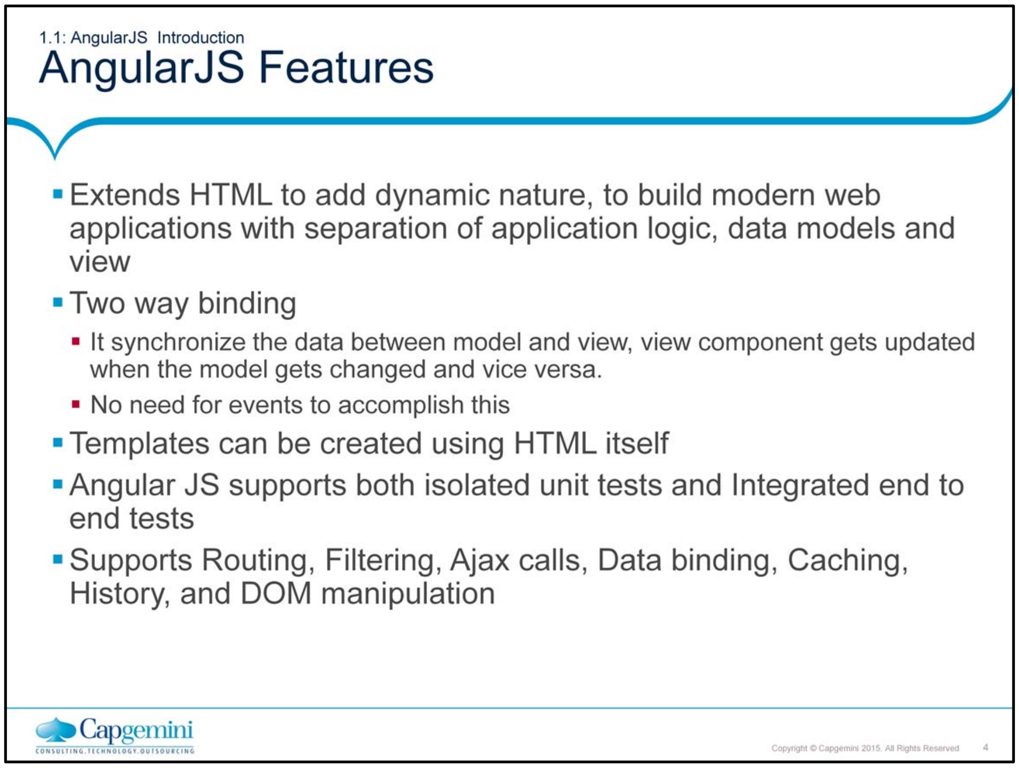
AngularJS is a client-side technology, written entirely in JavaScript.

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

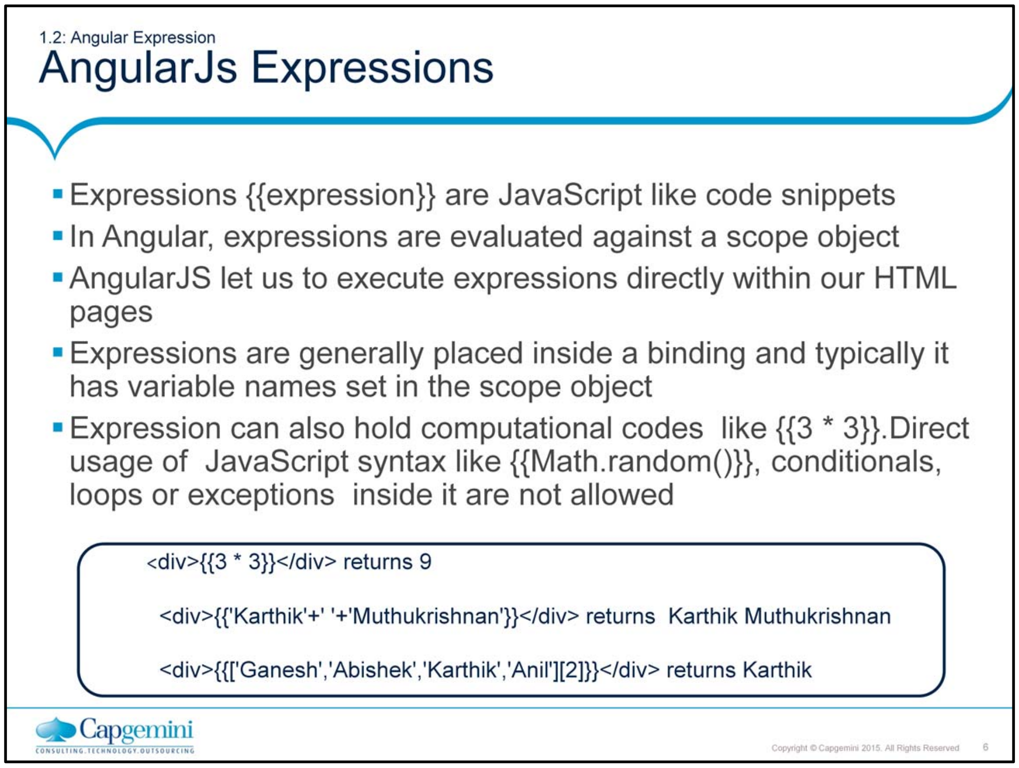
HTML is designed for static webpages, but developing a dynamic site with HTML we need to do many tricks to achieve what we want. With AngularJS extending the HTML it is really simple to make a dynamic site in a proper MVC structure.
With AngularJS we can achieve
Create a template and reuse it in application 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.

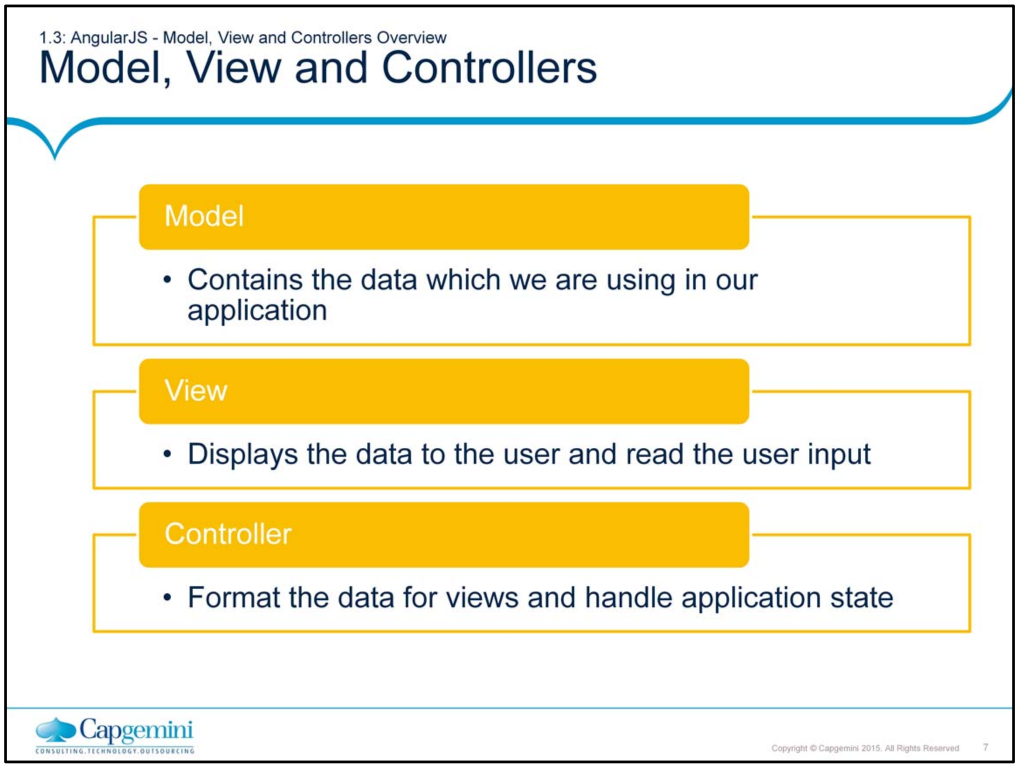


1.2: Angular Expression

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

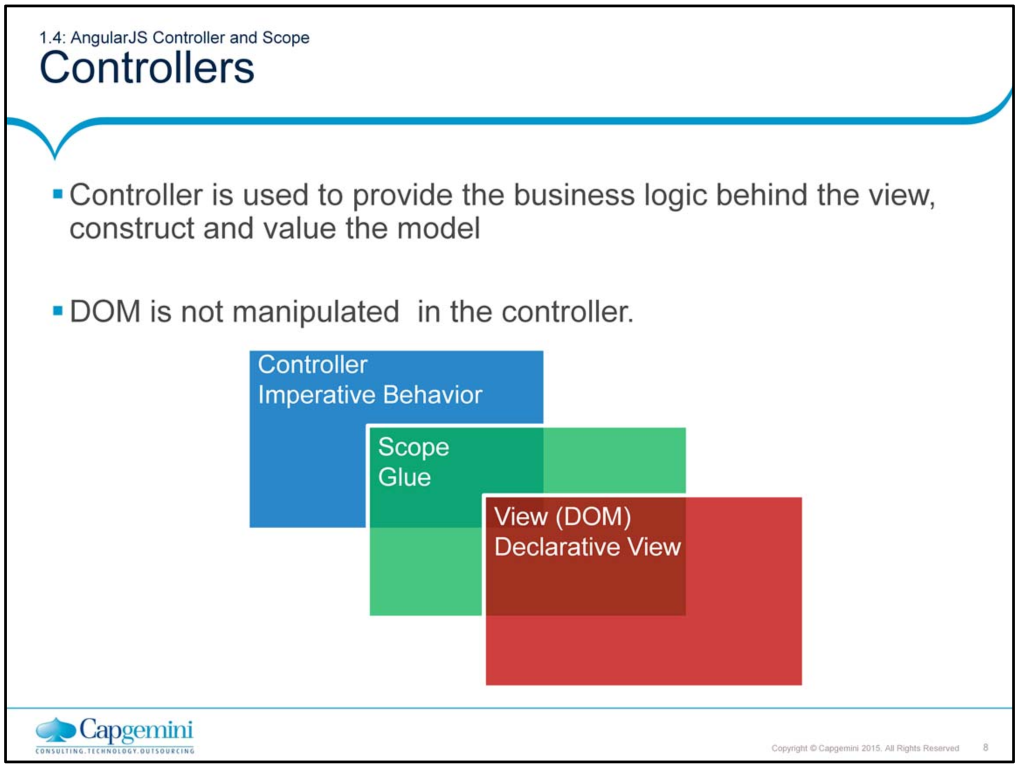
View

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

Controller

- Format the data for views and handle application state

7

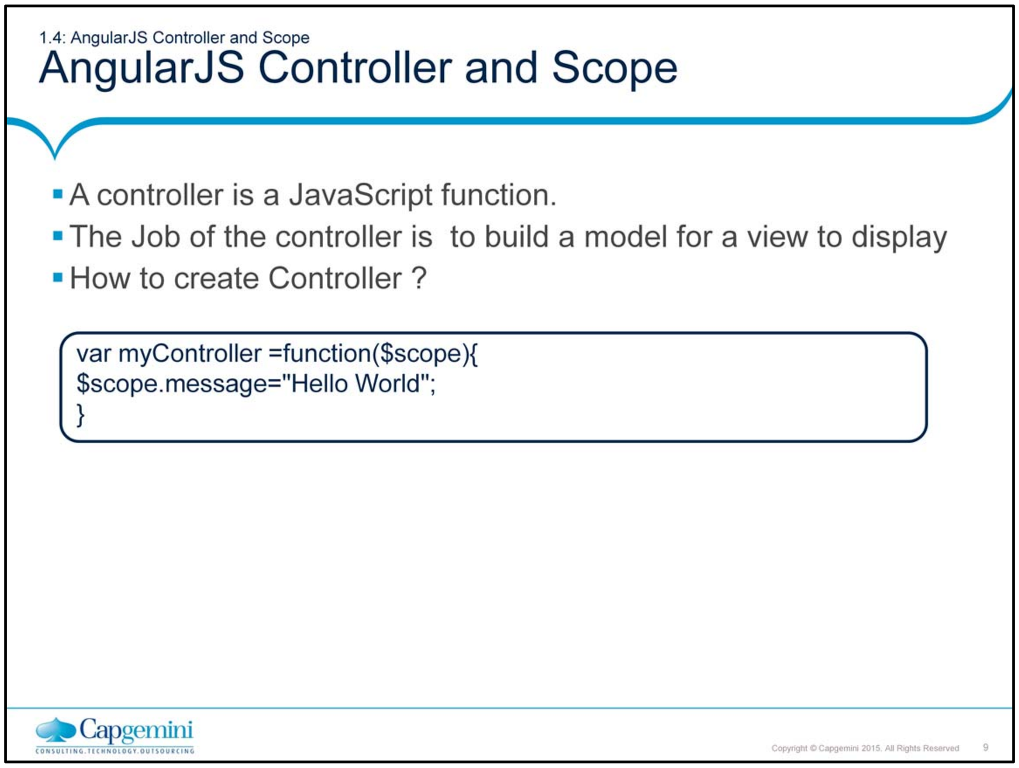


1.4: AngularJS Controller and Scope

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

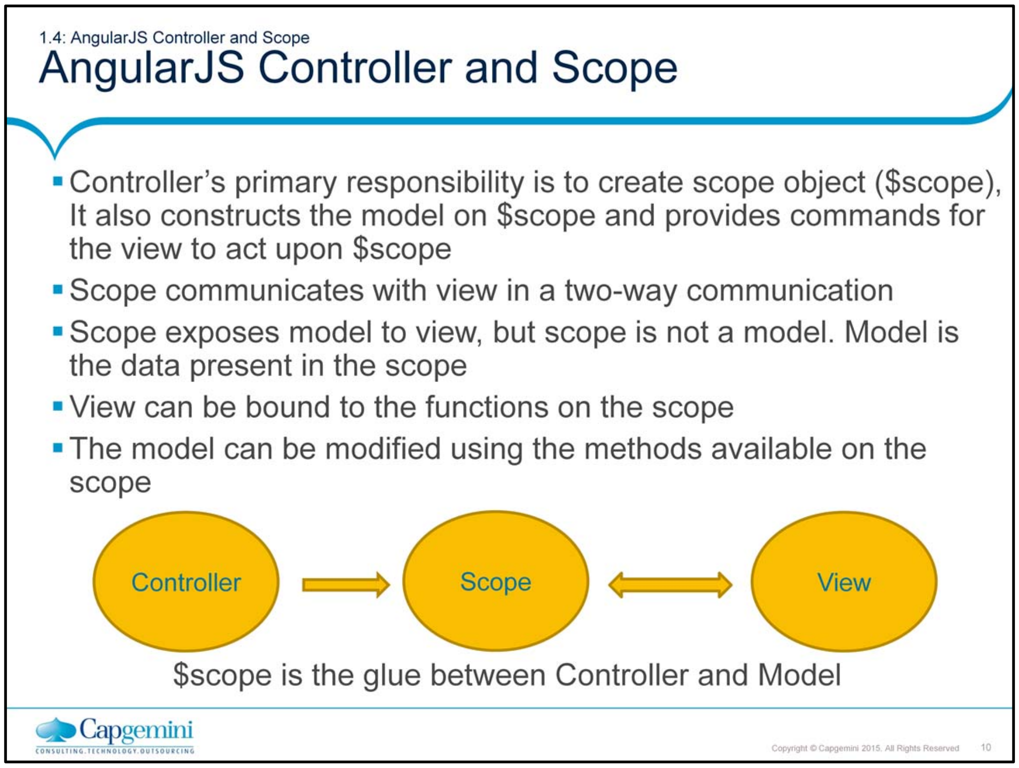


1.4: AngularJS Controller and Scope

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

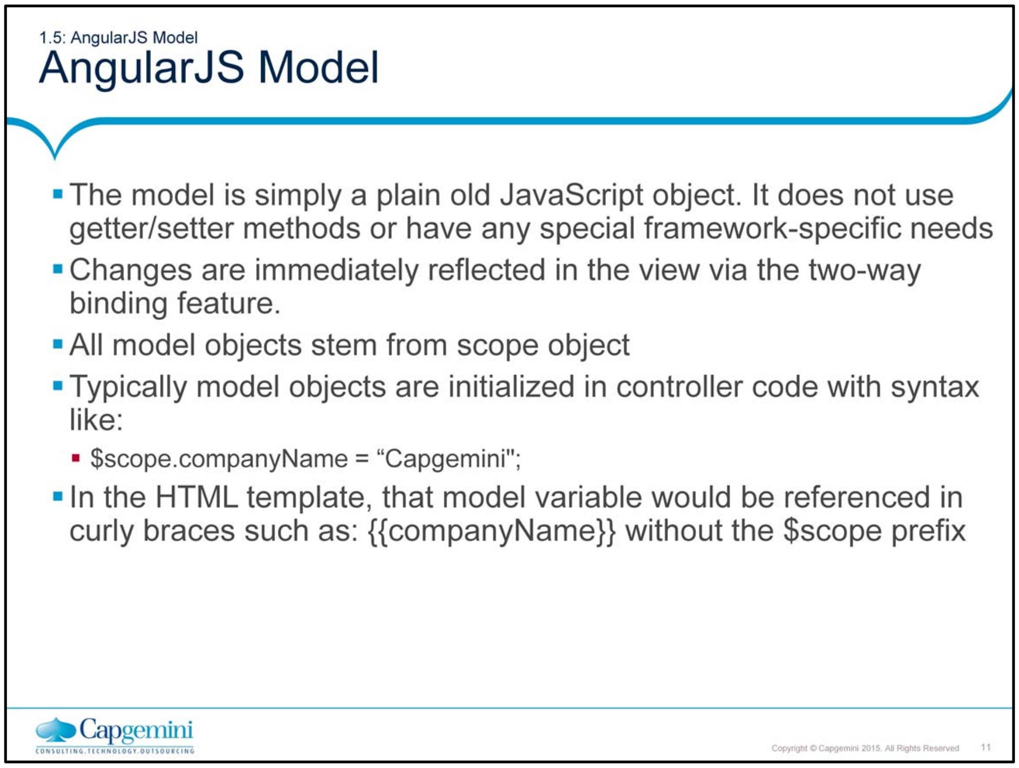
$scope is the glue between Controller and Model

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.

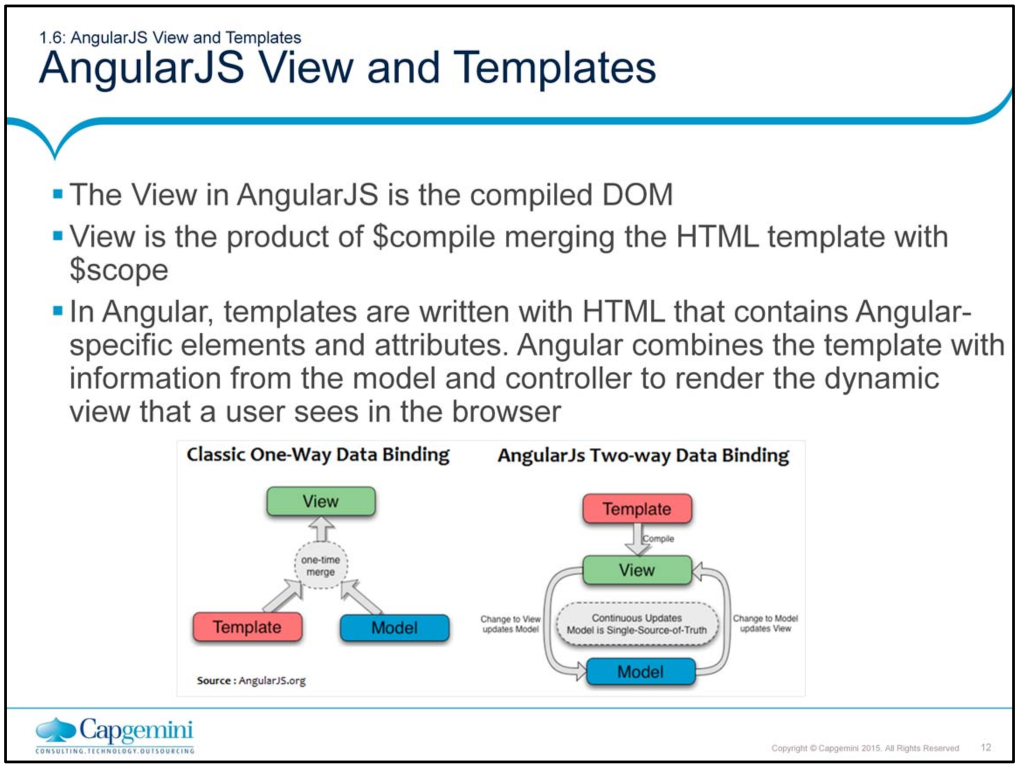


1.5: AngularJS Model

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

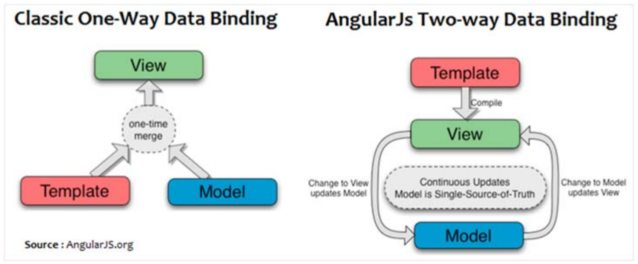
11



1.6: AngularJS View and Templates

AngularJS View and Templates

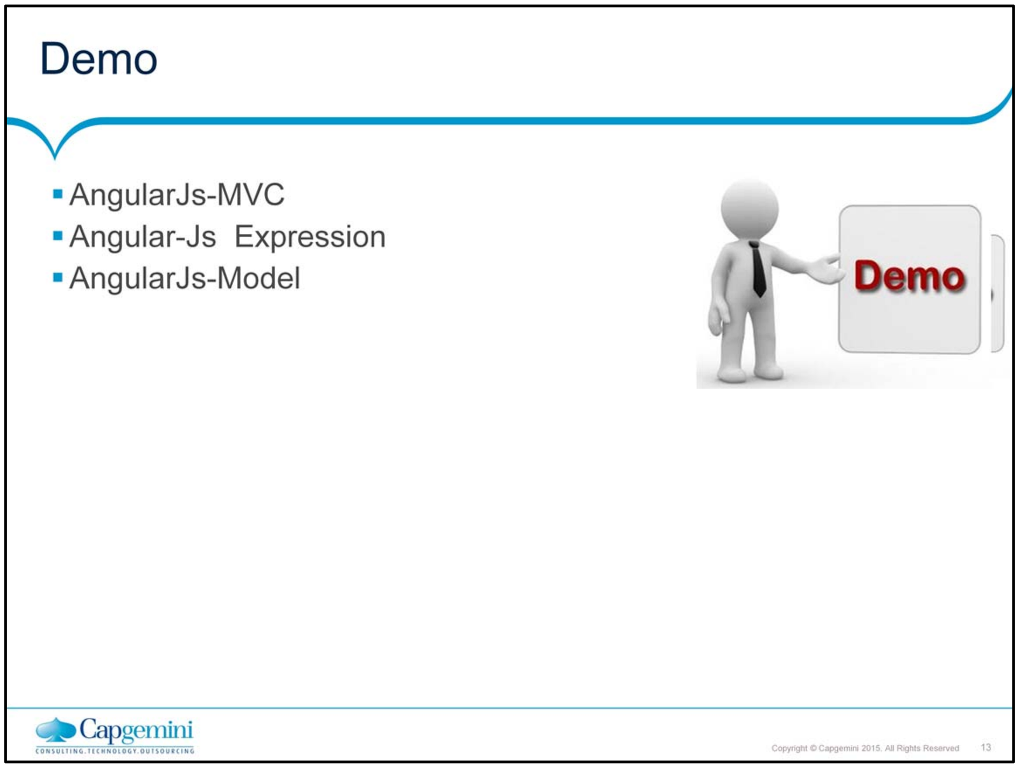
- The View in AngularJS is the compiled DOM
- View is the product of $compile merging the HTML template with $scope
- In Angular, templates are written with HTML that contains Angular-specific elements and attributes. Angular combines the template with information from the model and controller to render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


Classic One-Way Data Binding
View
one-time merge
Template
Model
Source : AngularJS.org
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

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.

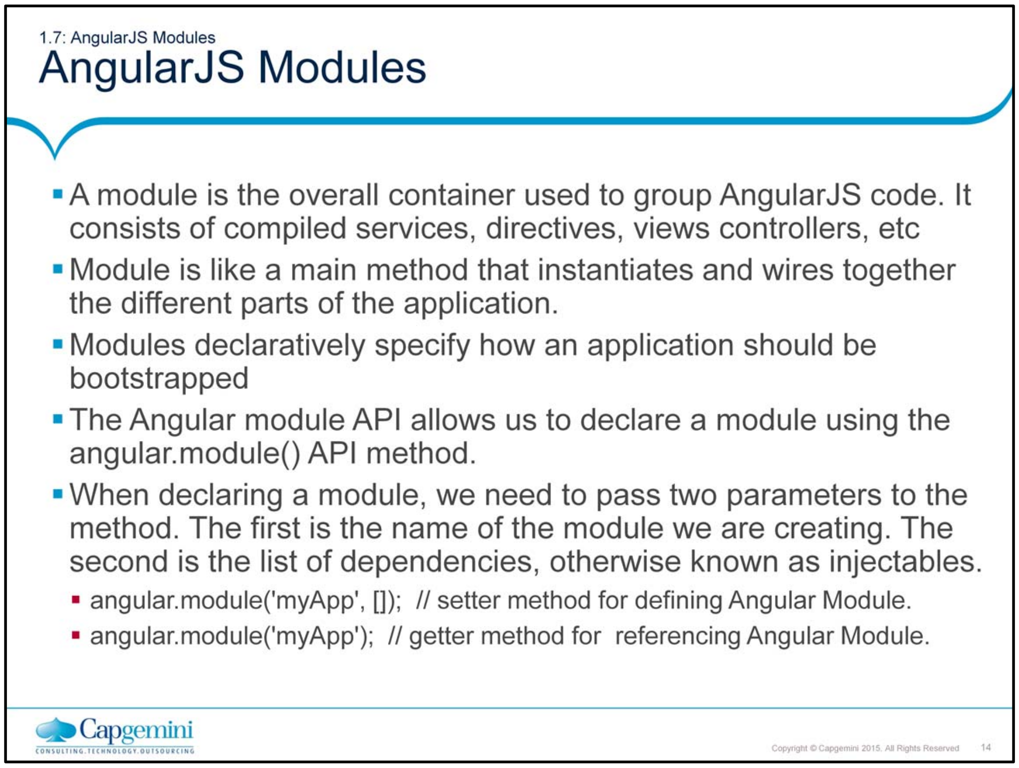
Angular is different. The Angular compiler consumes the DOM, not string templates. The result is a linking function, which when combined with a scope model results in a live view. The view and scope model bindings are transparent. The developer does not need to make any special calls to update the view. 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.

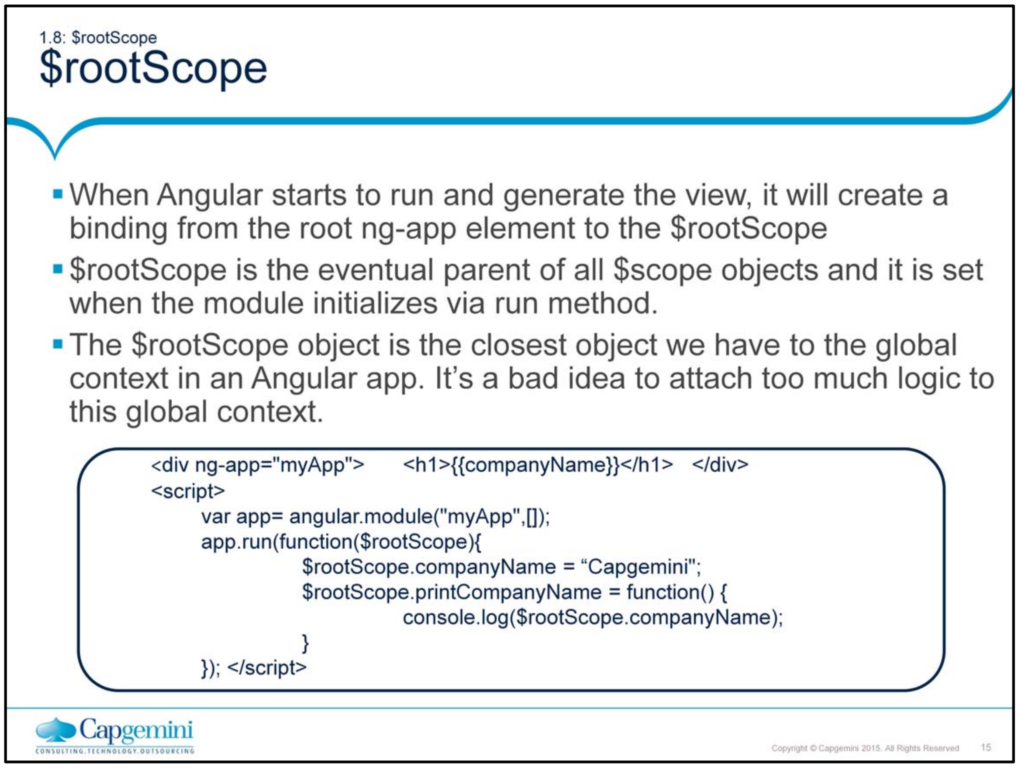
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

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.

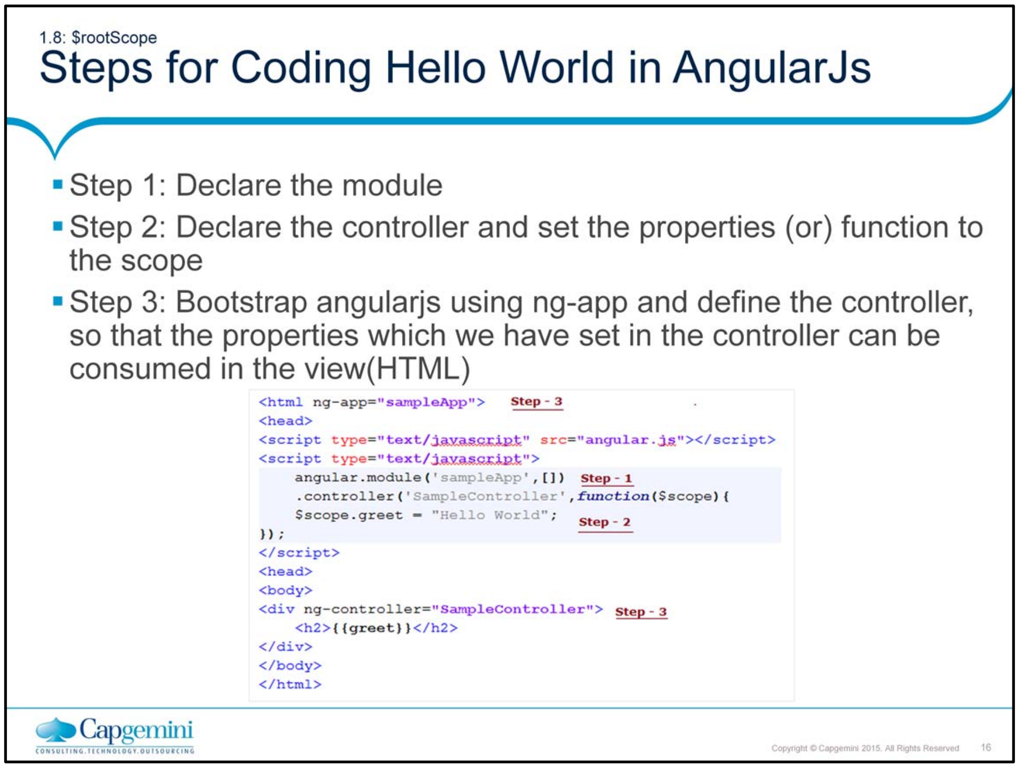
```
<div ng-app="myApp">      <h1>{{companyName}}</h1>   </div>
<script>
        var app= angular.module("myApp",[]);
        app.run(function($rootScope){
                        $rootScope.companyName = “Capgemini";
                        $rootScope.printCompanyName = function() {
               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

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

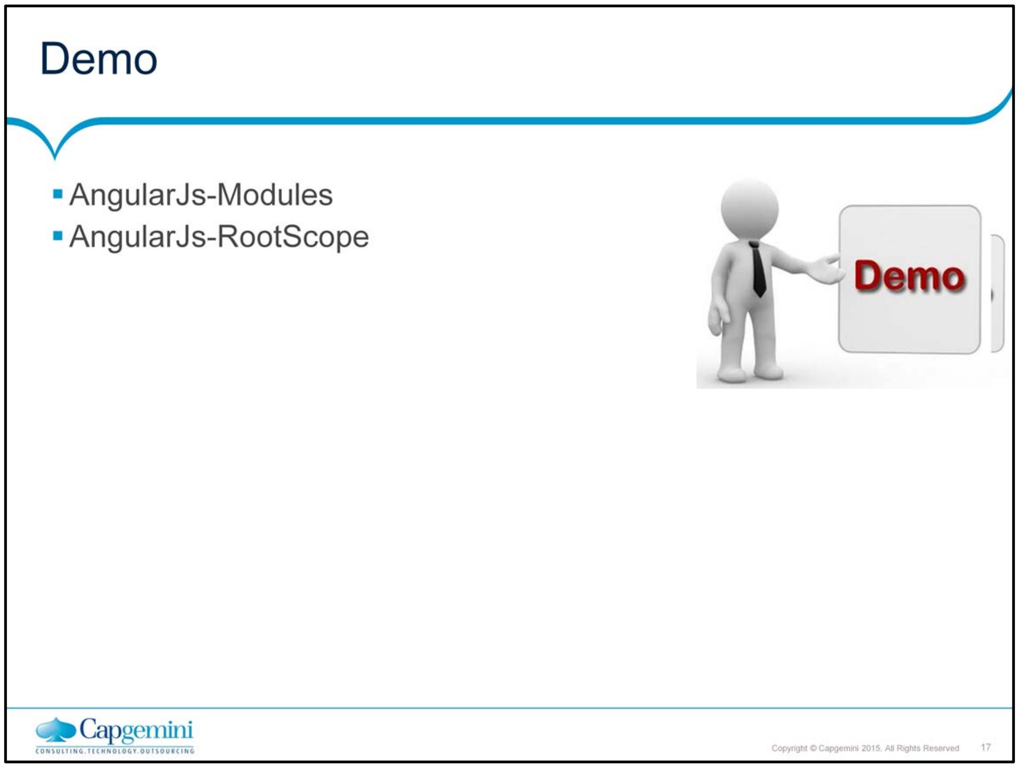
Steps for Coding Hello World in AngularJs

- 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
CONSULTING.TECHNOLOGY.OUTSOURCING

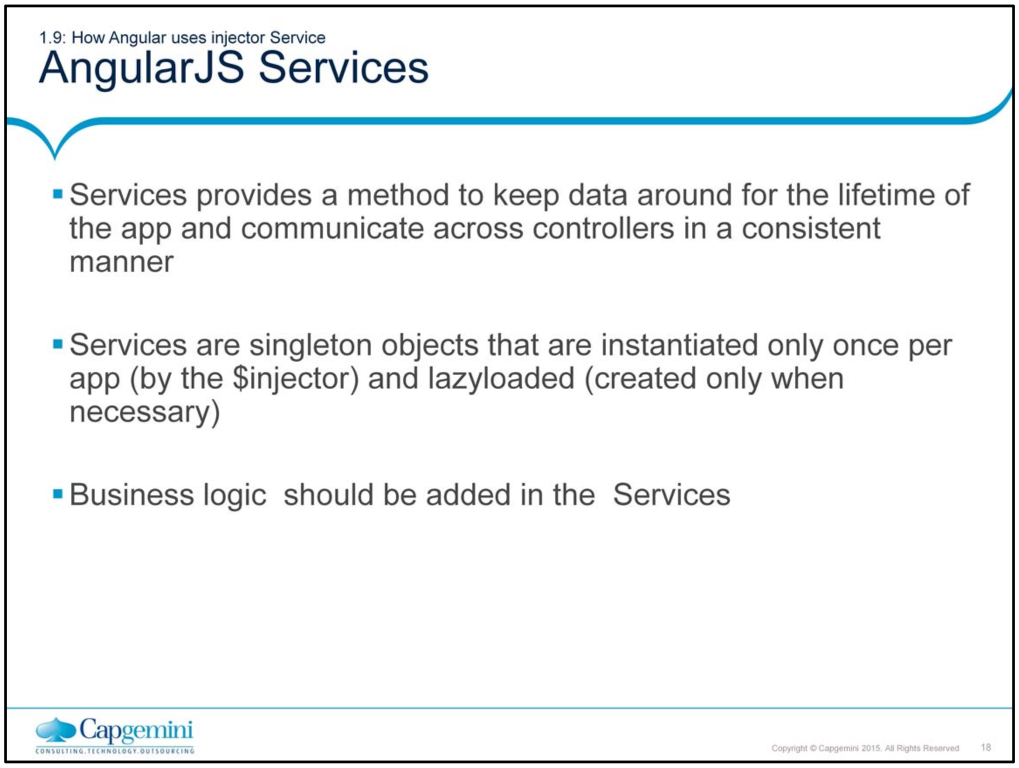
Step 1 Declare a Module by using angular.module ,‘sampleApp’ this sampleApp have same name as ng-app
Step 2 Attach with controller ‘SampleController’ have same name what we write in ng-controller



Demo

- AngularJs-Modules
- AngularJs-RootScope

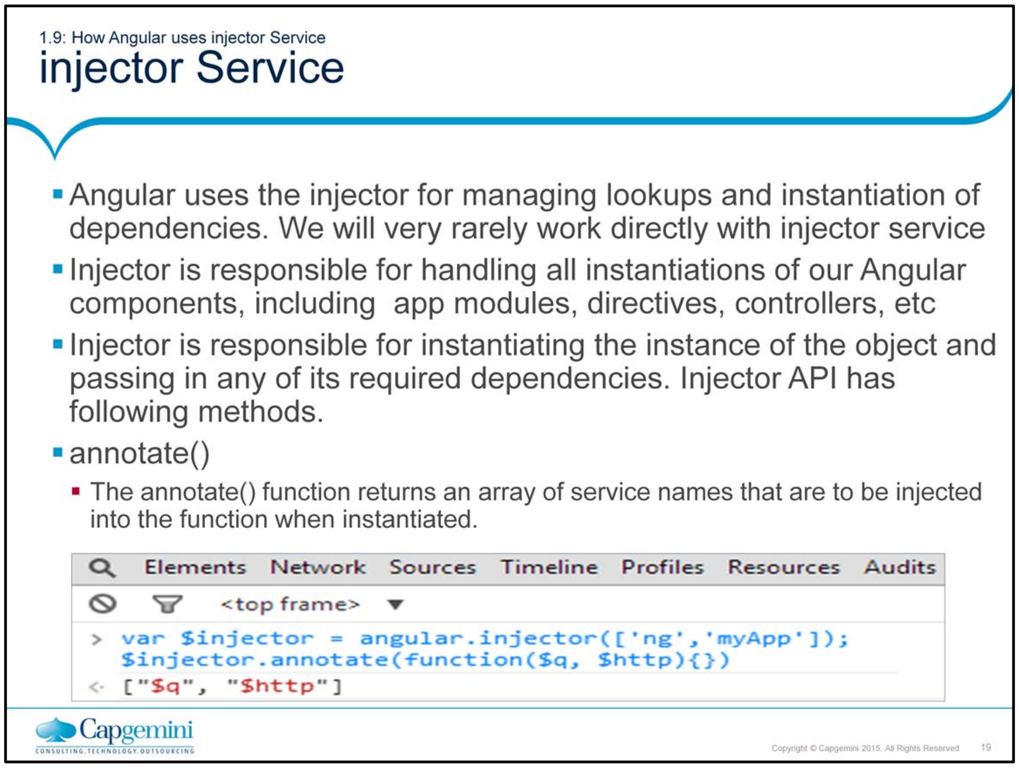
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



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



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"]
```

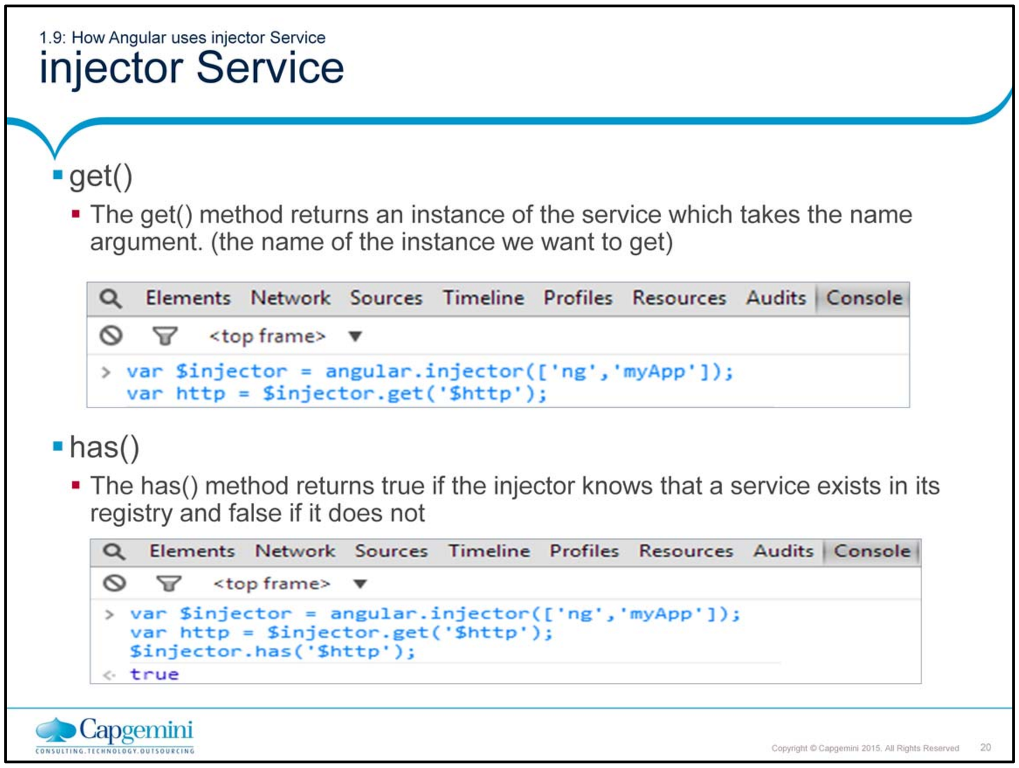
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

$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>

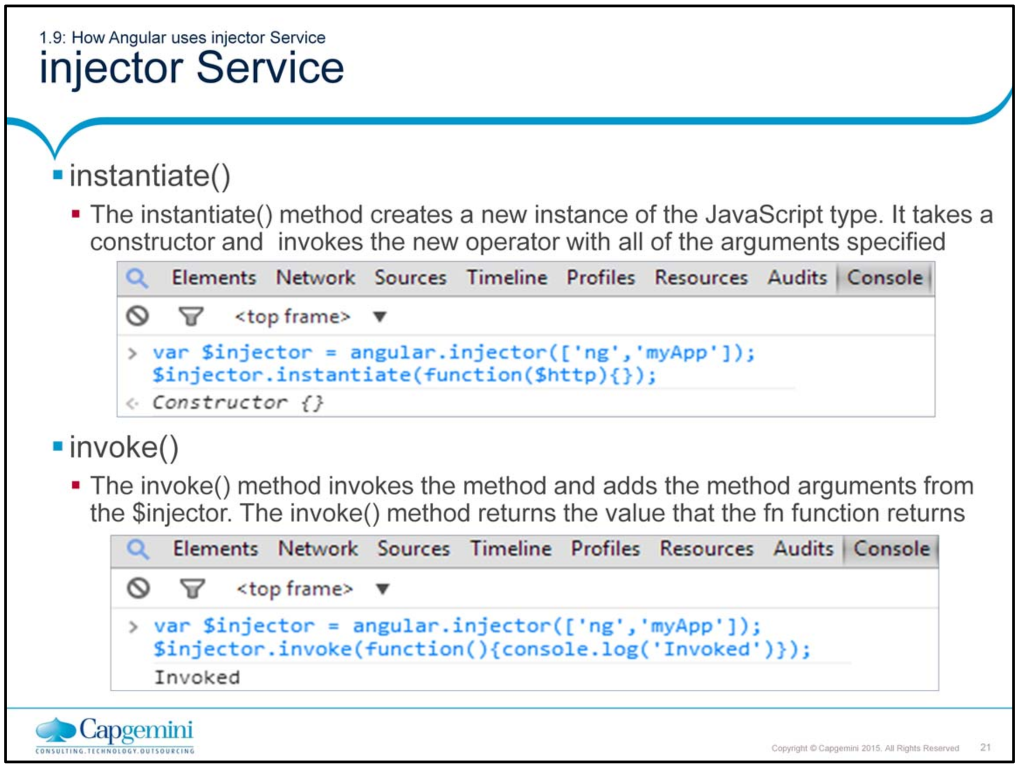
```
> var $injector = angular.injector(['ng','myApp']);
  var http = $injector.get('$http');
  $injector.has('$http');
<· true
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

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>

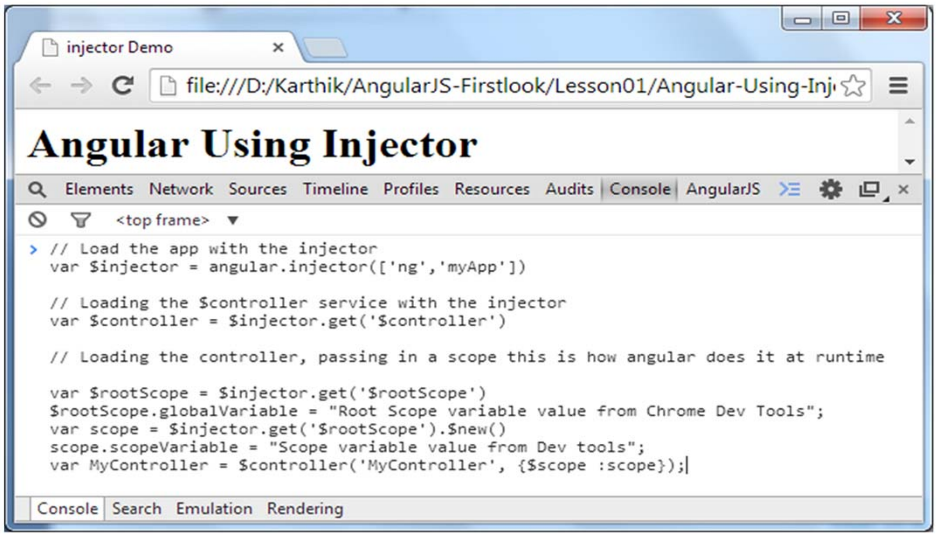
```
> var $injector = angular.injector(['ng','myApp']);
  $injector.invoke(function(){console.log('Invoked')});
  Invoked
```

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

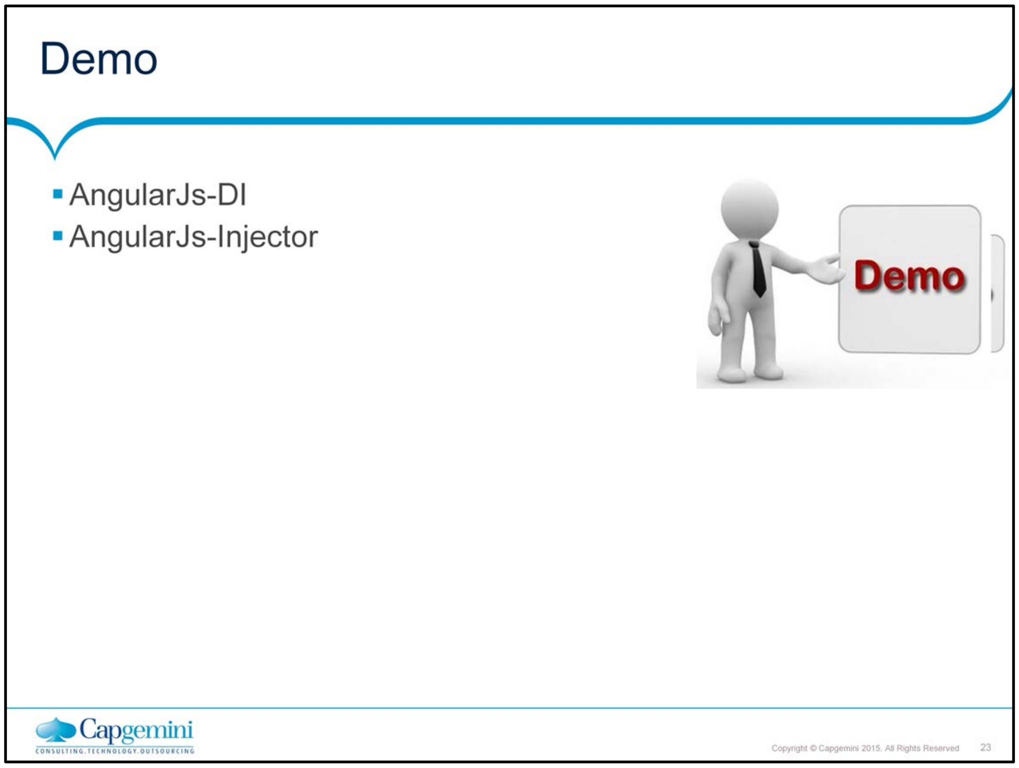
How Angular uses injector Service


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

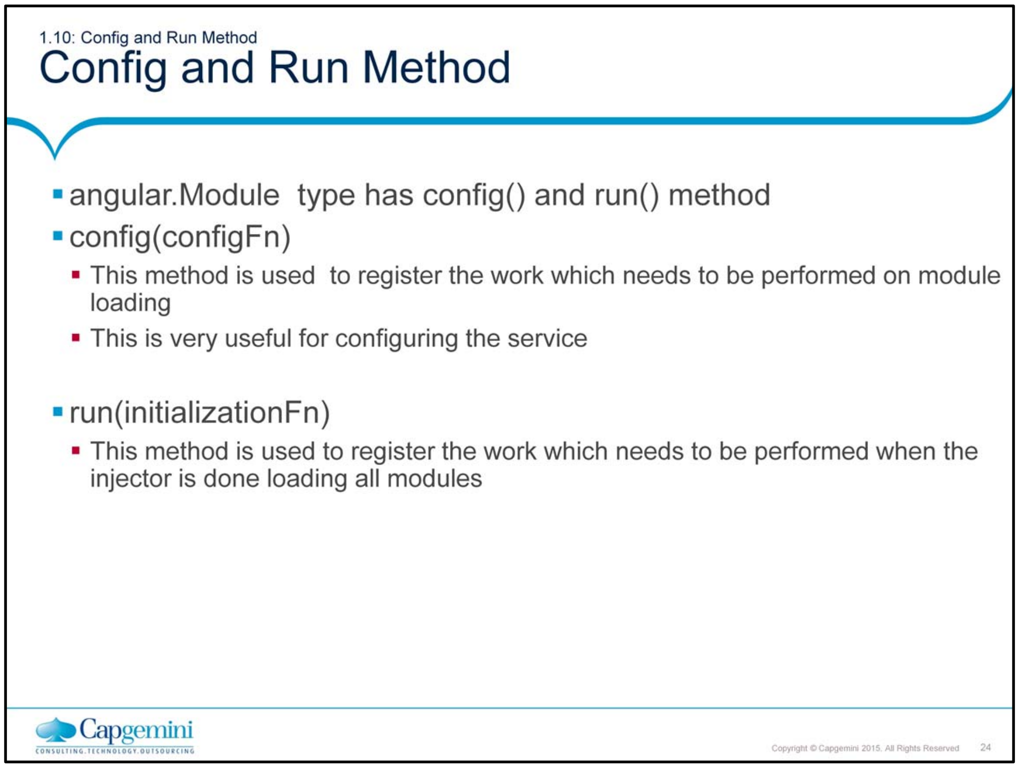
```
// Load the app with the injector
var $injector = angular.injector(['ng','myApp'])

// 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

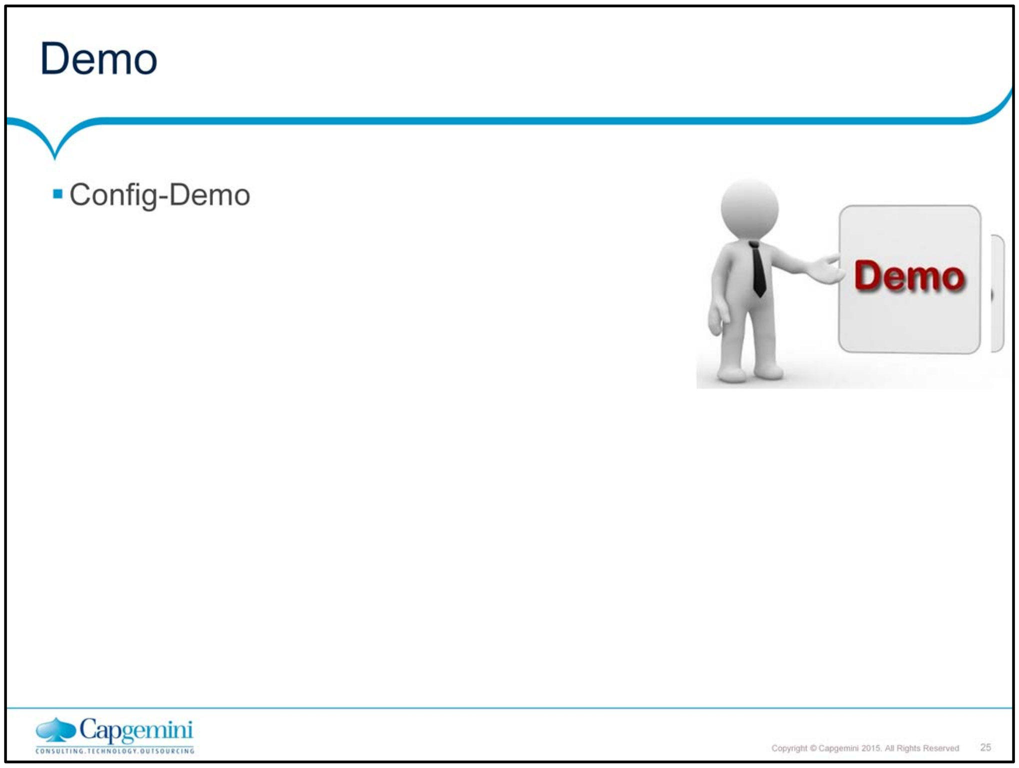


1.10: Config and Run Method

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

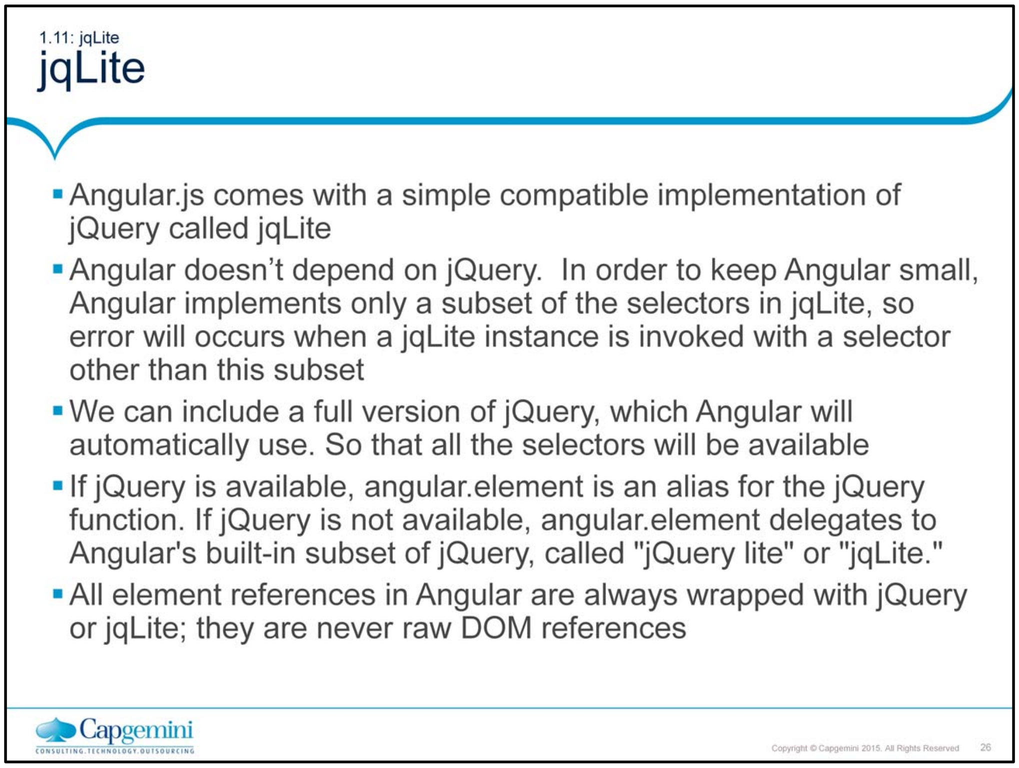
A module is a collection of configuration and run blocks which get applied to the application during the bootstrap process. In its simplest form the module consists of a collection of two kinds of blocks:
Configuration blocks - get executed during the provider registrations and configuration phase. Only 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

25



1.11: jqLite

jqLite

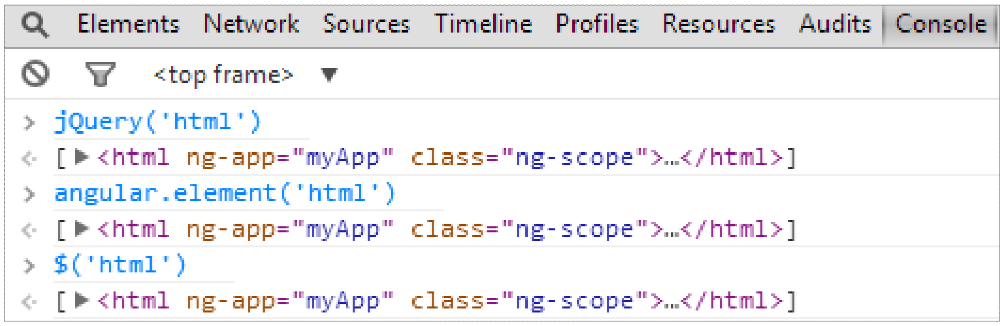
- Angular.js comes with a simple compatible implementation of jQuery called jqLite
- Angular doesn't depend on jQuery. In order to keep Angular small, Angular implements only a subset of the selectors in jqLite, so error will occurs when a jqLite instance is invoked with a selector other than this subset
- 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

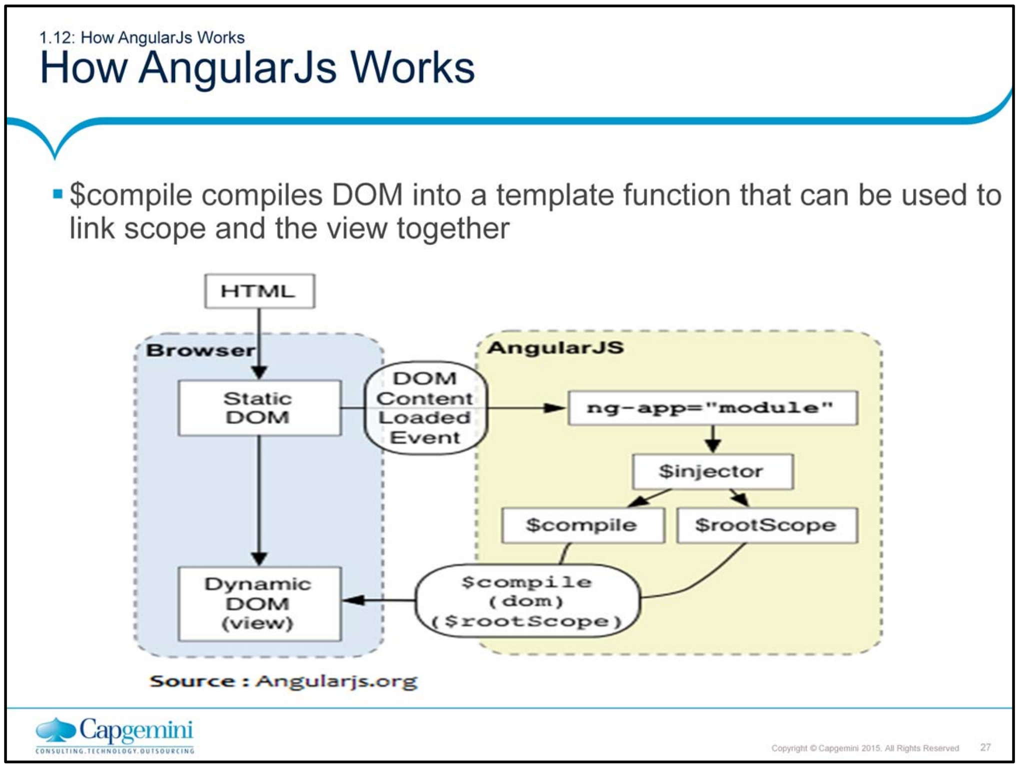
jqLite provides only the following jQuery methods:

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.



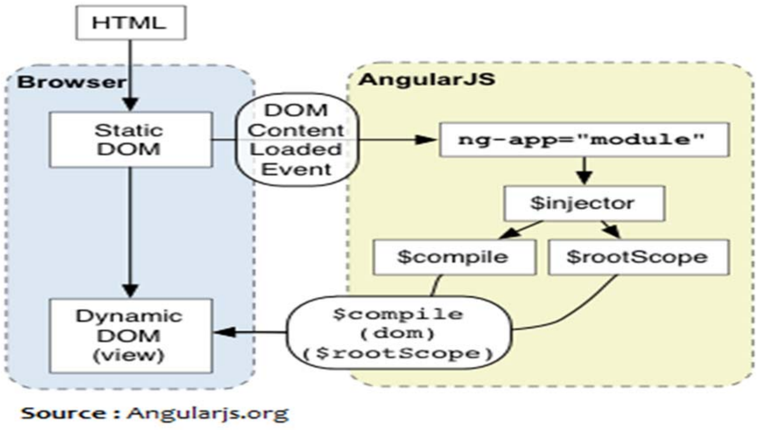
```
> jQuery('html')
< [▶<html ng-app="myApp" class="ng-scope">…</html>]
> angular.element('html')
< [▶<html ng-app="myApp" class="ng-scope">…</html>]
> $('html')
< [▶<html ng-app="myApp" class="ng-scope">…</html>]
```



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



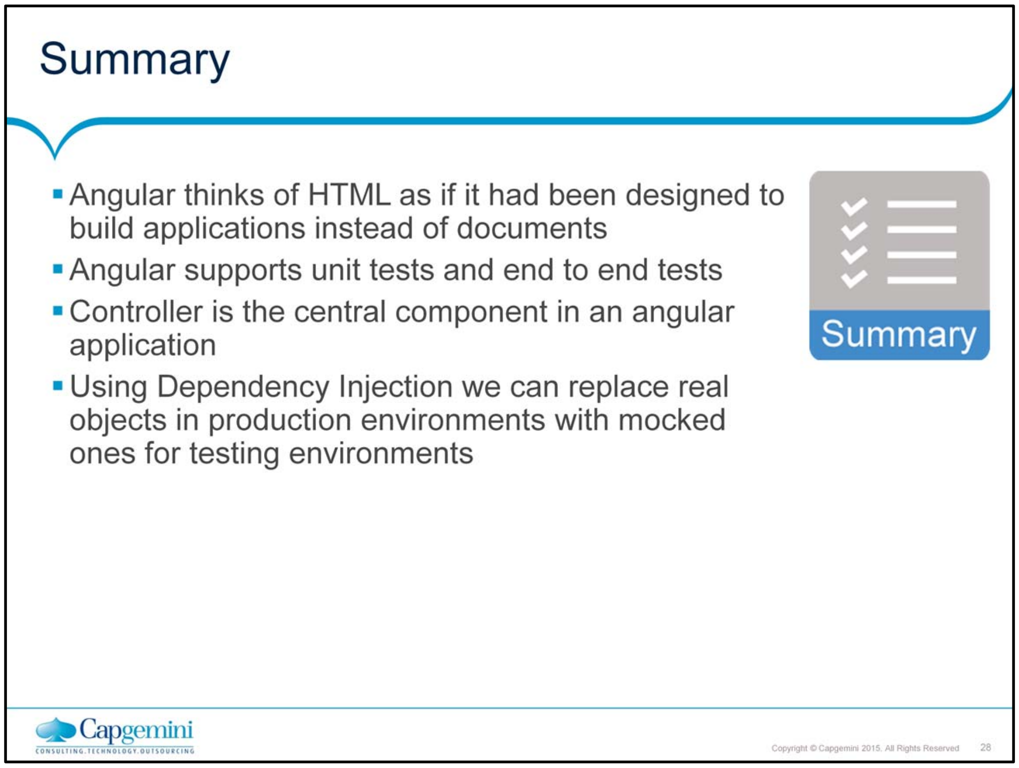
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

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.



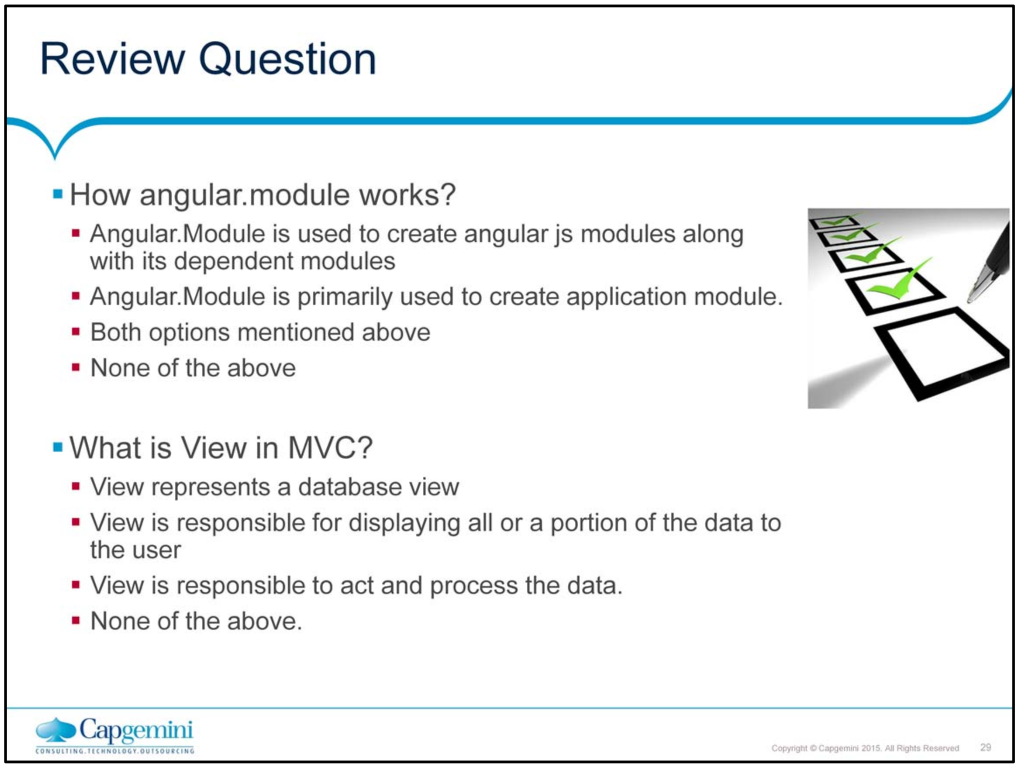
Summary

- 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

Summary

28

Add the notes here.



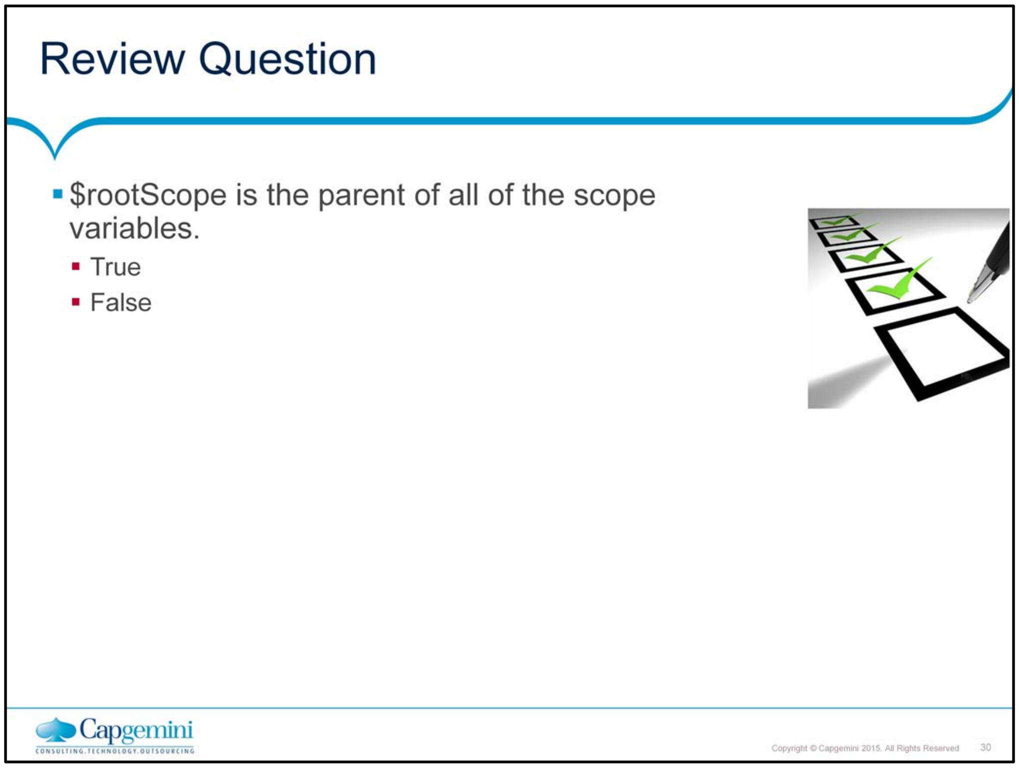
Review Question

- 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.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

29



Review Question

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