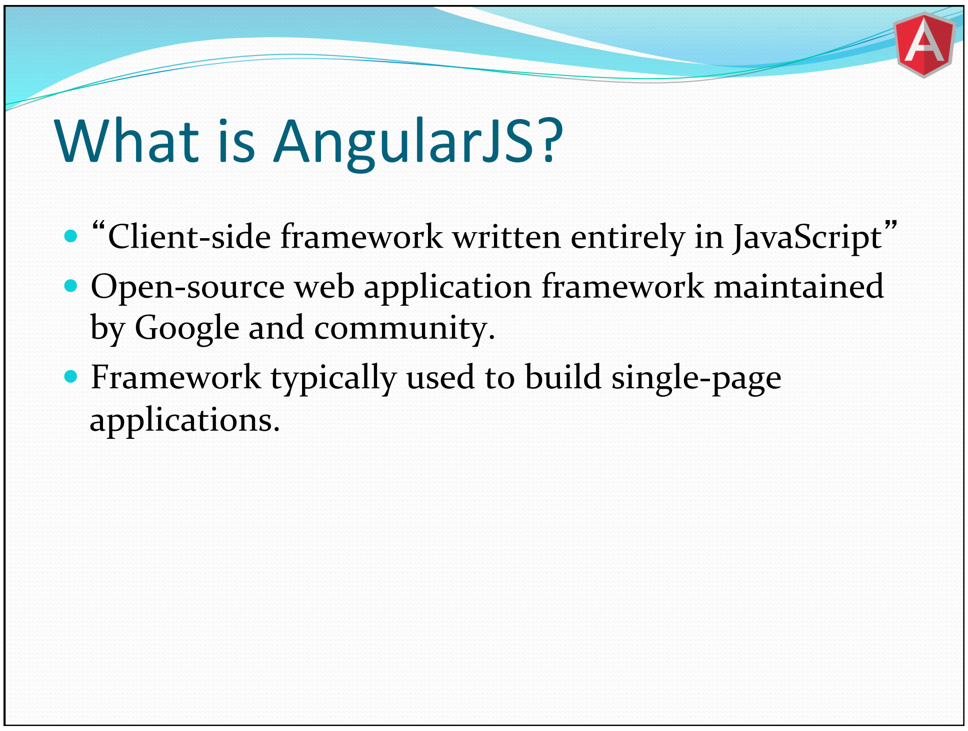
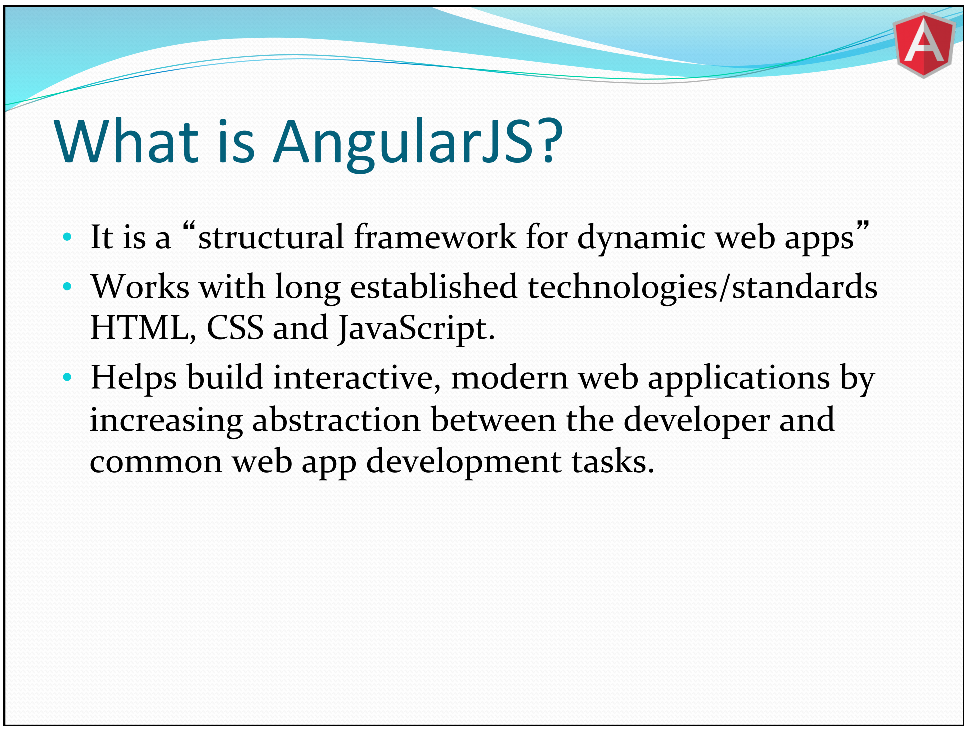
AngularJS

Compiled by Lakshman M N
Tech Consultant / Mentor
Lakshman.mn@gmail.com



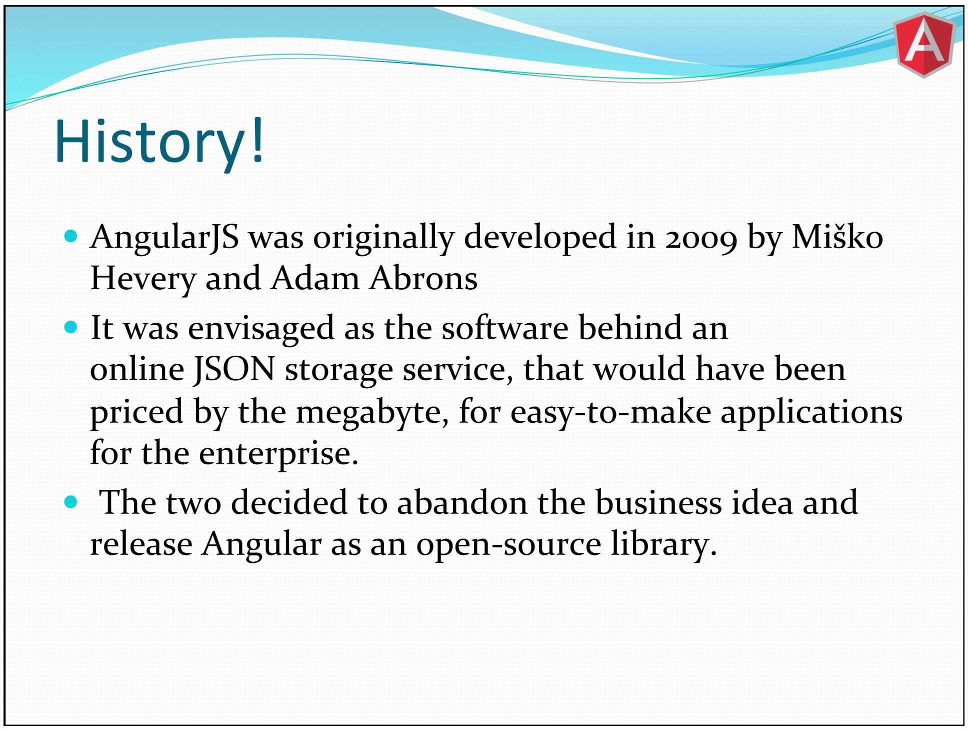
What is AngularJS?

- “Client-side framework written entirely in JavaScript”
- Open-source web application framework maintained by Google and community.
- Framework typically used to build single-page applications.



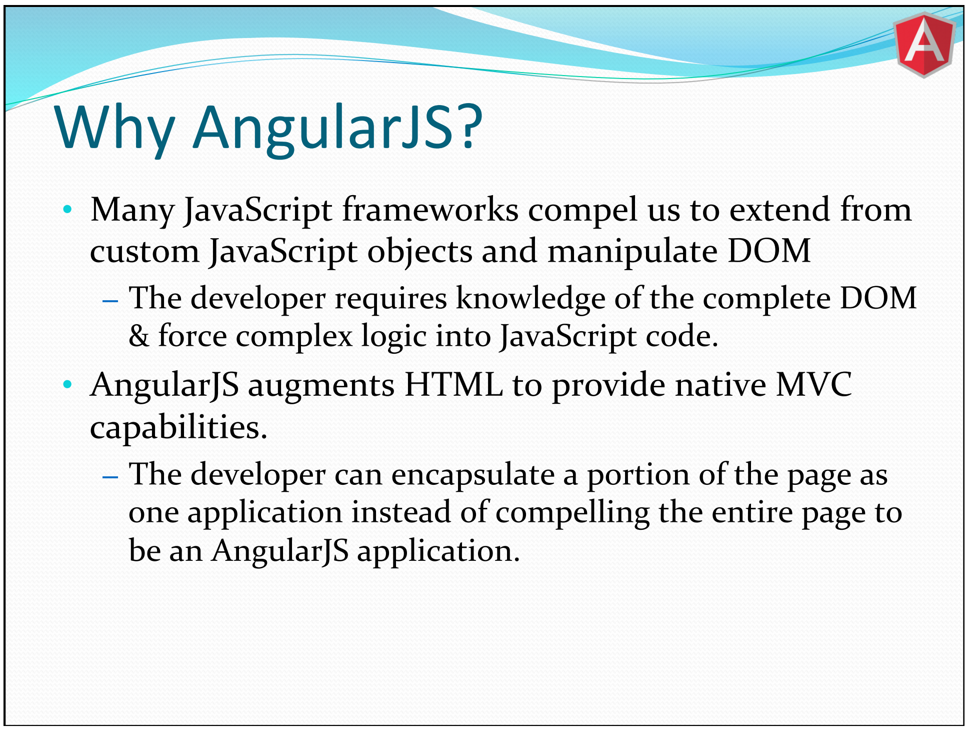
What is AngularJS?

- It is a “structural framework for dynamic web apps”
- Works with long established technologies/standards HTML, CSS and JavaScript.
- Helps build interactive, modern web applications by increasing abstraction between the developer and common web app development tasks.



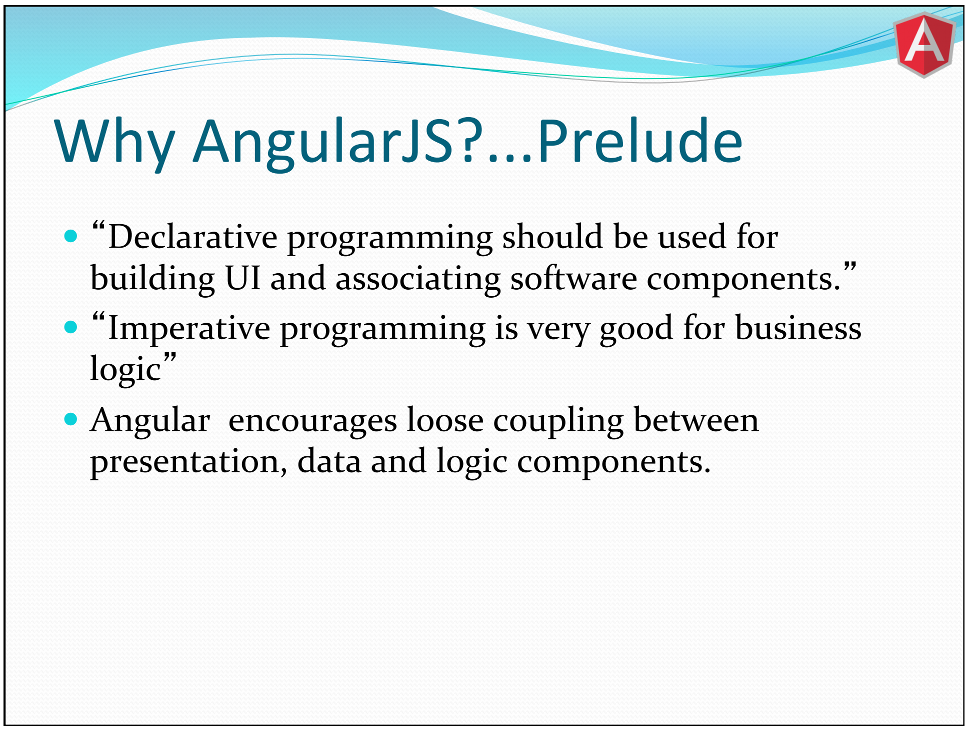
History!

- AngularJS was originally developed in 2009 by Miško Hevery and Adam Abrons
- It was envisaged as the software behind an online JSON storage service, that would have been priced by the megabyte, for easy-to-make applications for the enterprise.
- The two decided to abandon the business idea and release Angular as an open-source library.



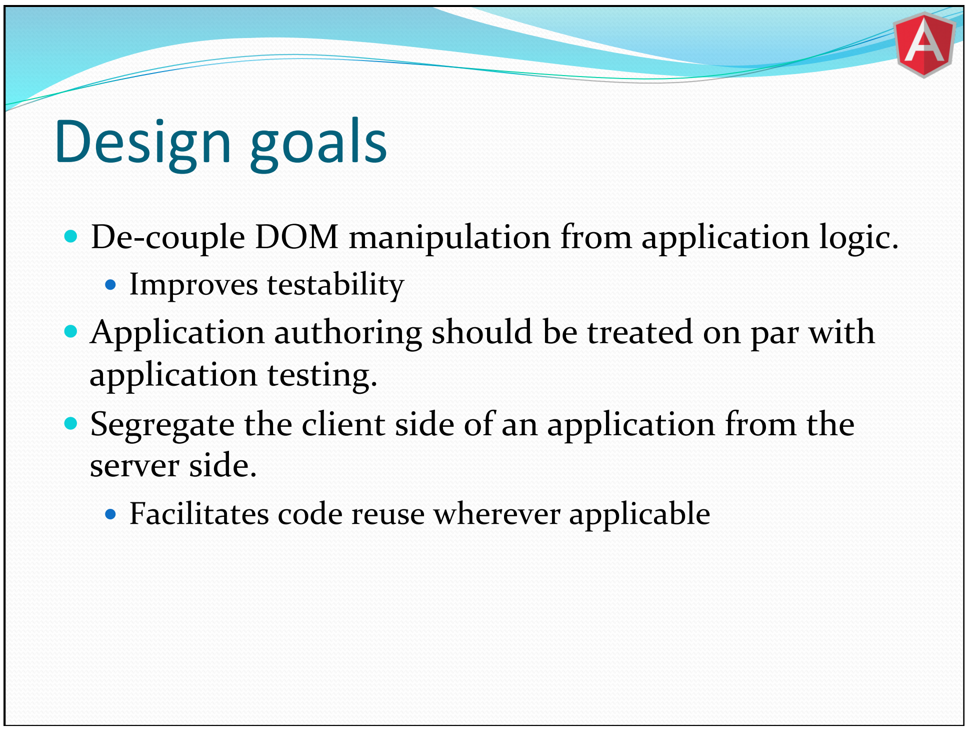
Why AngularJS?

- Many JavaScript frameworks compel us to extend from custom JavaScript objects and manipulate DOM
  - The developer requires knowledge of the complete DOM & force complex logic into JavaScript code.
- AngularJS augments HTML to provide native MVC capabilities.
  - The developer can encapsulate a portion of the page as one application instead of compelling the entire page to be an AngularJS application.



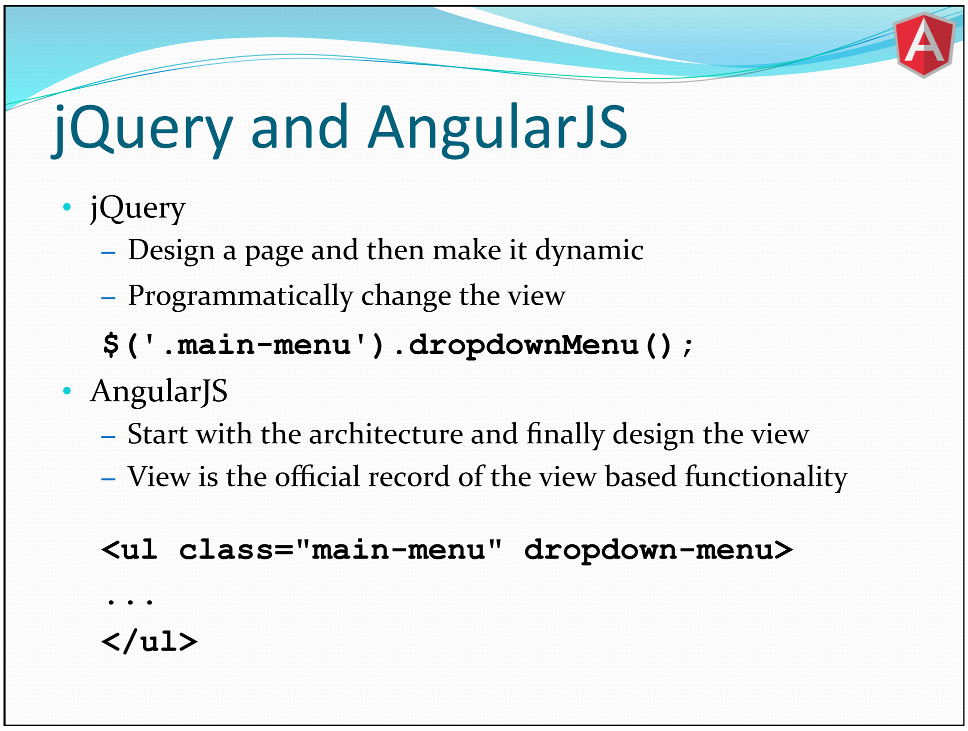
Why AngularJS?...Prelude

- “Declarative programming should be used for building UI and associating software components.”
- “Imperative programming is very good for business logic”
- Angular encourages loose coupling between presentation, data and logic components.



Design goals

- De-couple DOM manipulation from application logic.
  - Improves testability
- Application authoring should be treated on par with application testing.
- Segregate the client side of an application from the server side.
  - Facilitates code reuse wherever applicable

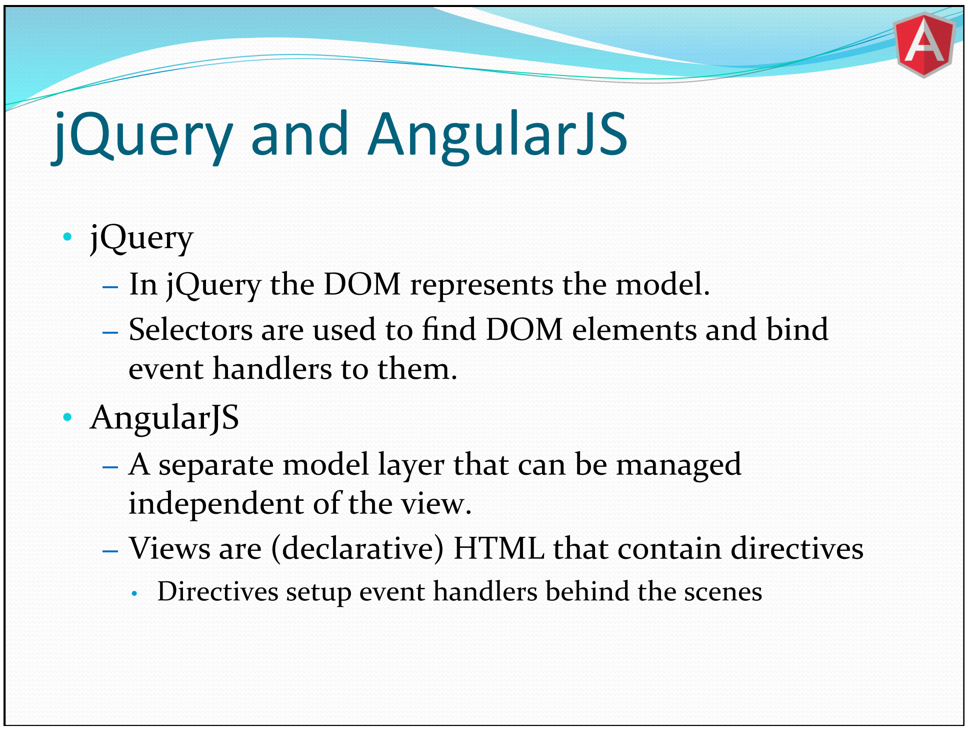


jQuery and AngularJS

- jQuery
  - Design a page and then make it dynamic
  - Programmatically change the view

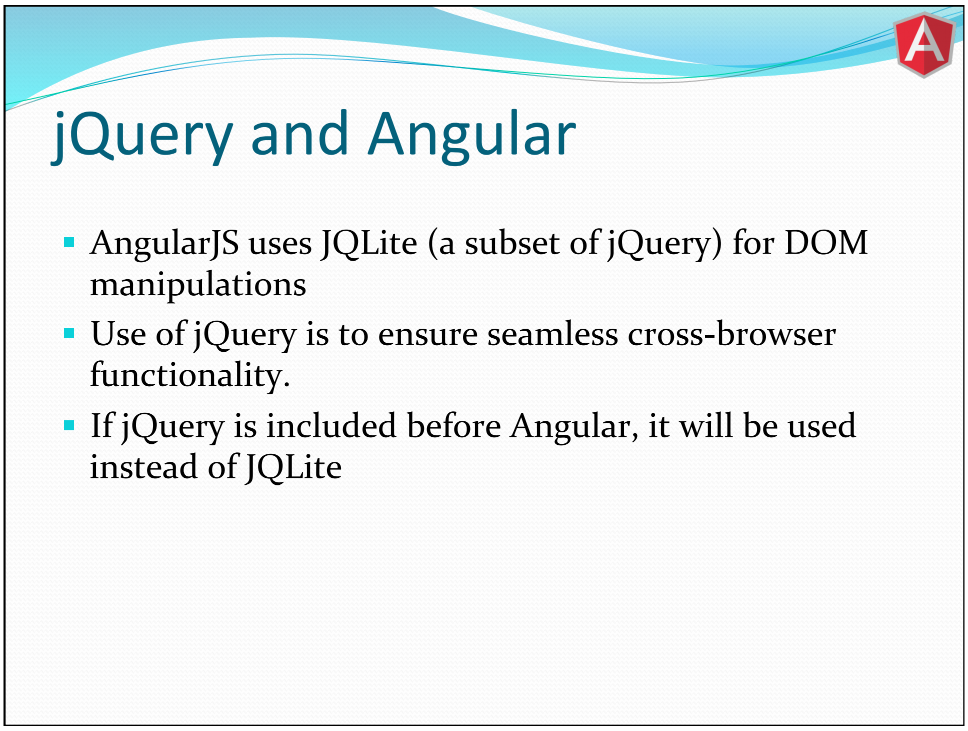
  ```
  $('.main-menu').dropdownMenu();
  ```
- AngularJS
  - Start with the architecture and finally design the view
  - View is the official record of the view based functionality

  ```
  <ul class="main-menu" dropdown-menu>
  ...
  </ul>
  ```



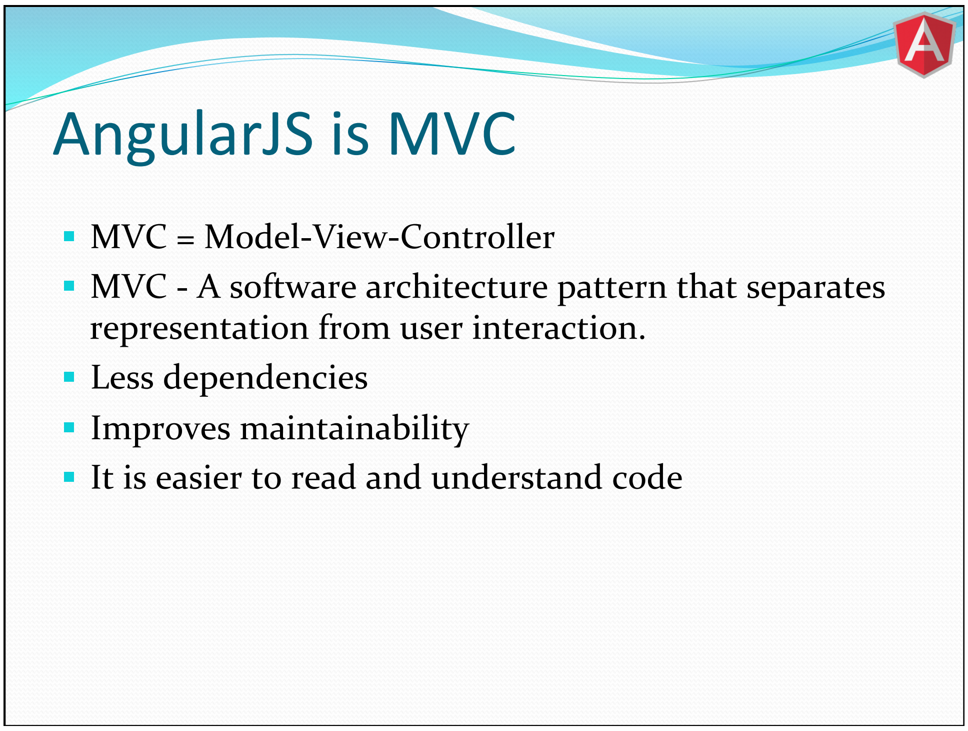
jQuery and AngularJS

- jQuery
  - In jQuery the DOM represents the model.
  - Selectors are used to find DOM elements and bind event handlers to them.
- AngularJS
  - A separate model layer that can be managed independent of the view.
  - Views are (declarative) HTML that contain directives
    - Directives setup event handlers behind the scenes



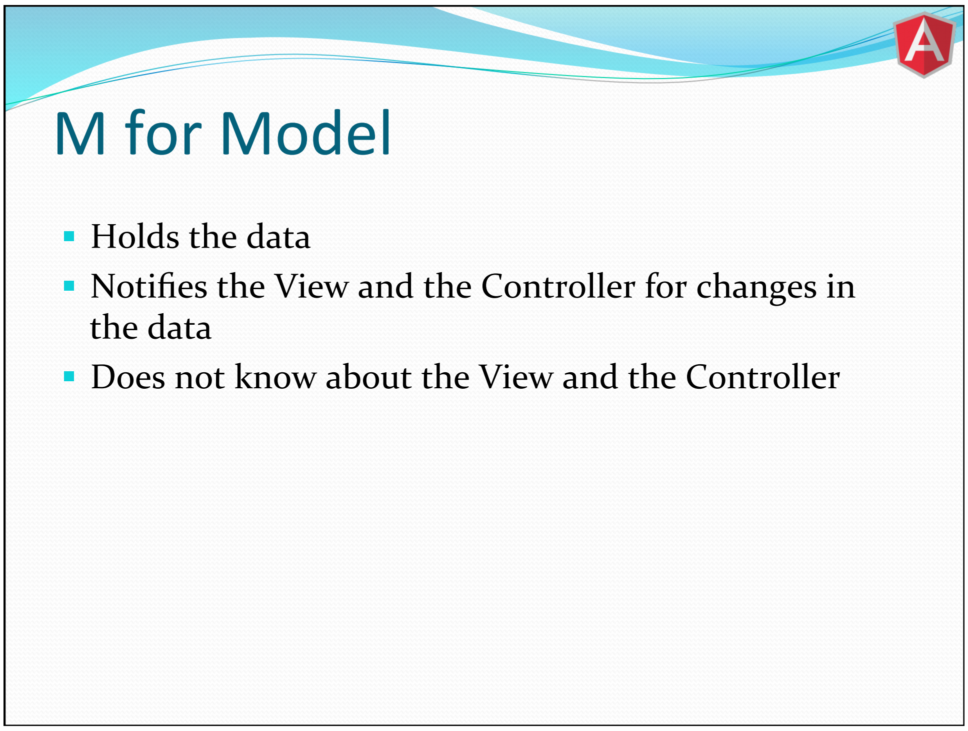
jQuery and Angular

- AngularJS uses JQLite (a subset of jQuery) for DOM manipulations
- Use of jQuery is to ensure seamless cross-browser functionality.
- If jQuery is included before Angular, it will be used instead of JQLite



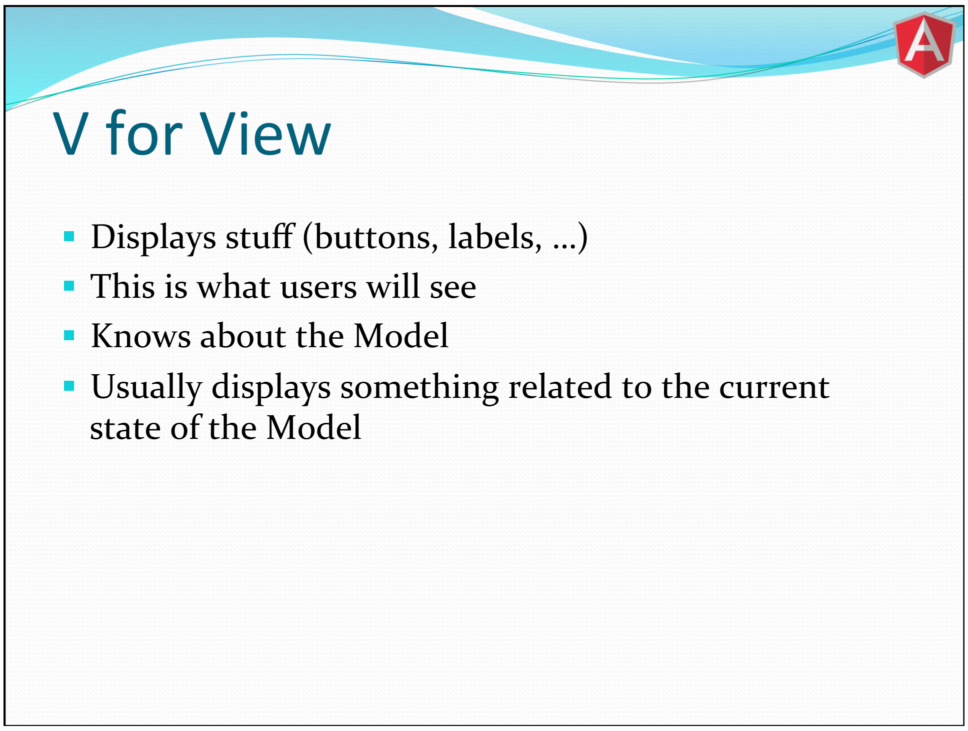
AngularJS is MVC

- MVC = Model-View-Controller
- MVC - A software architecture pattern that separates representation from user interaction.
- Less dependencies
- Improves maintainability
- It is easier to read and understand code



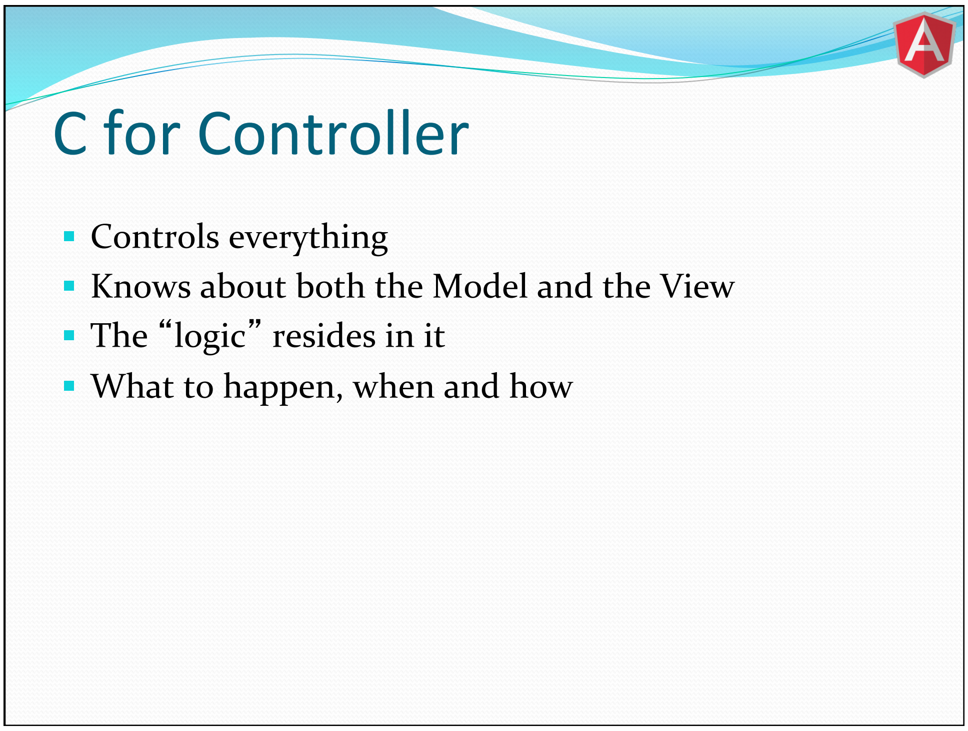
M for Model

- Holds the data
- Notifies the View and the Controller for changes in the data
- Does not know about the View and the Controller



V for View

- Displays stuff (buttons, labels, …)
- This is what users will see
- Knows about the Model
- Usually displays something related to the current state of the Model



C for Controller

- Controls everything
- Knows about both the Model and the View
- The “logic” resides in it
- What to happen, when and how

An AngularJS application

```
<!DOCTYPE html>
<html ng-app>
<head>
<title>Simple app</title>
<script src="angular.js">
</script>
</head>
<body>
<input ng-model="name" type="text"
  placeholder="Your name">
<h1>Hello <span ng-bind="name"></span></h1>
</body>
</html>
```

Demo01.htm

An AngularJS application…

- The **ng-app** attribute is set on an element of DOM.
- The **ng-app** attribute declares that everything inside of it belongs to this Angular app.
  - This approach helps nest an Angular app in a web app.
- The only components that are impacted by AngularJS are the DOM elements that are declared inside the element with the **ng-app** attribute.
- Instead of merging data into a template and replacing a DOM element, AngularJS creates live templates as a view.

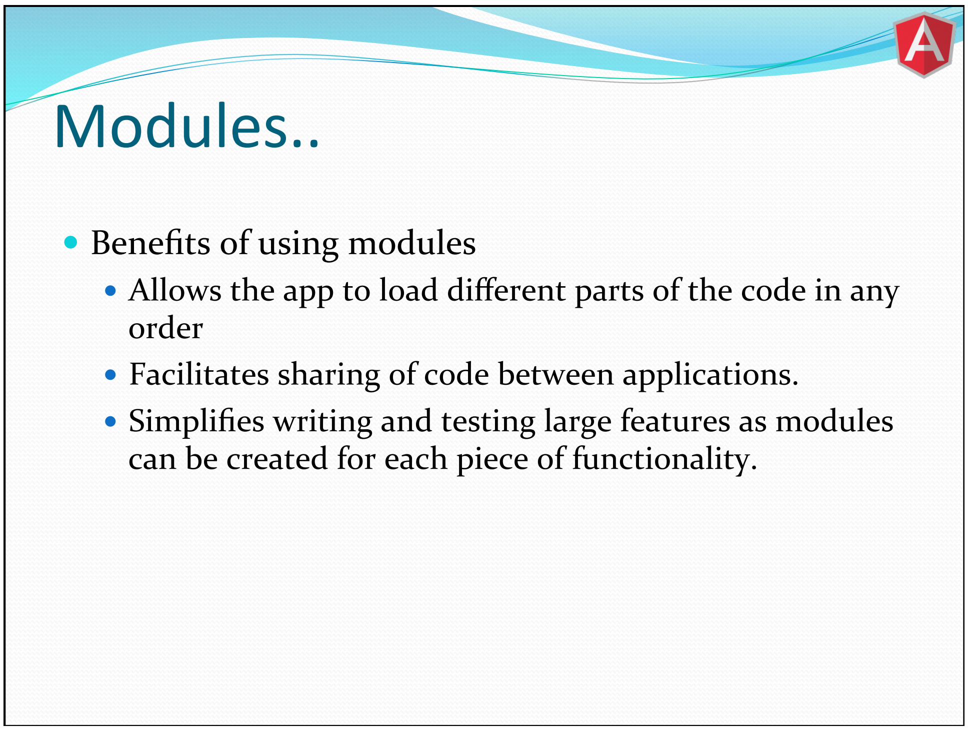
Data Binding

- The **`city`** attribute is bound to the input field using the **`ng-model`** directive.
- The **`ng-init`** is used for initialization.
- The value placed in the input field will be reflected in the **`model`** object.
- The change in the **`model`** in turn updates the **`view`**.

Demo02.htm

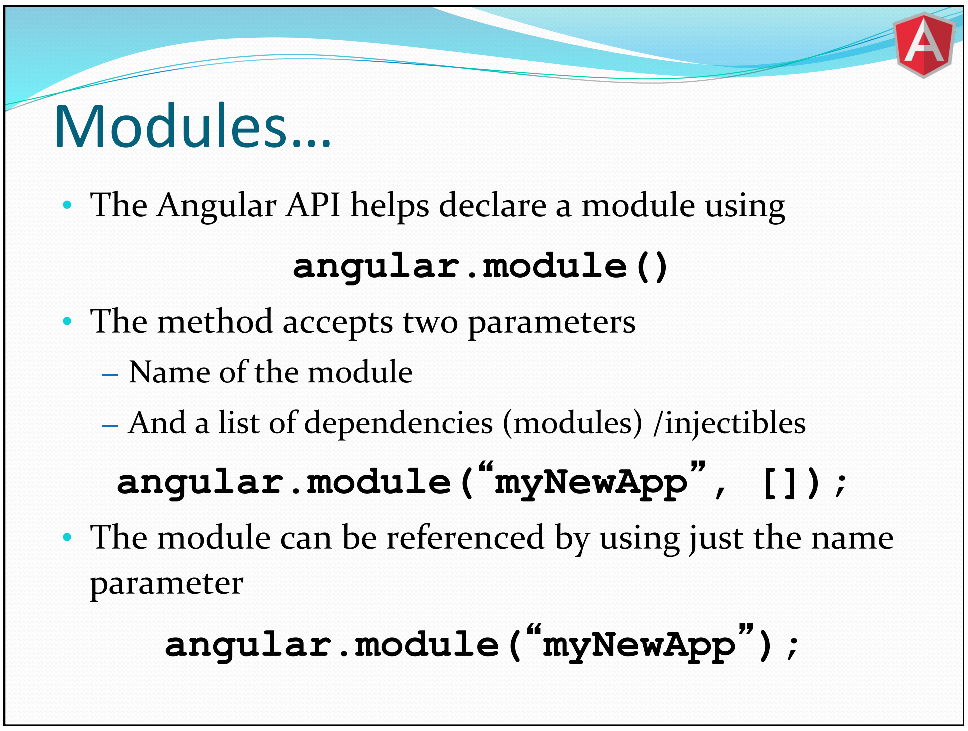
Modules

- Module is the main approach to define an app in AngularJS.
- Modules are logical entities that an application can be divided into.
- Module of an app contains all the application code.
- An app can contain several modules.
- Each module can contain code for a specific functionality.



Modules..

- Benefits of using modules
  - Allows the app to load different parts of the code in any order
  - Facilitates sharing of code between applications.
  - Simplifies writing and testing large features as modules can be created for each piece of functionality.



Modules…

- The Angular API helps declare a module using

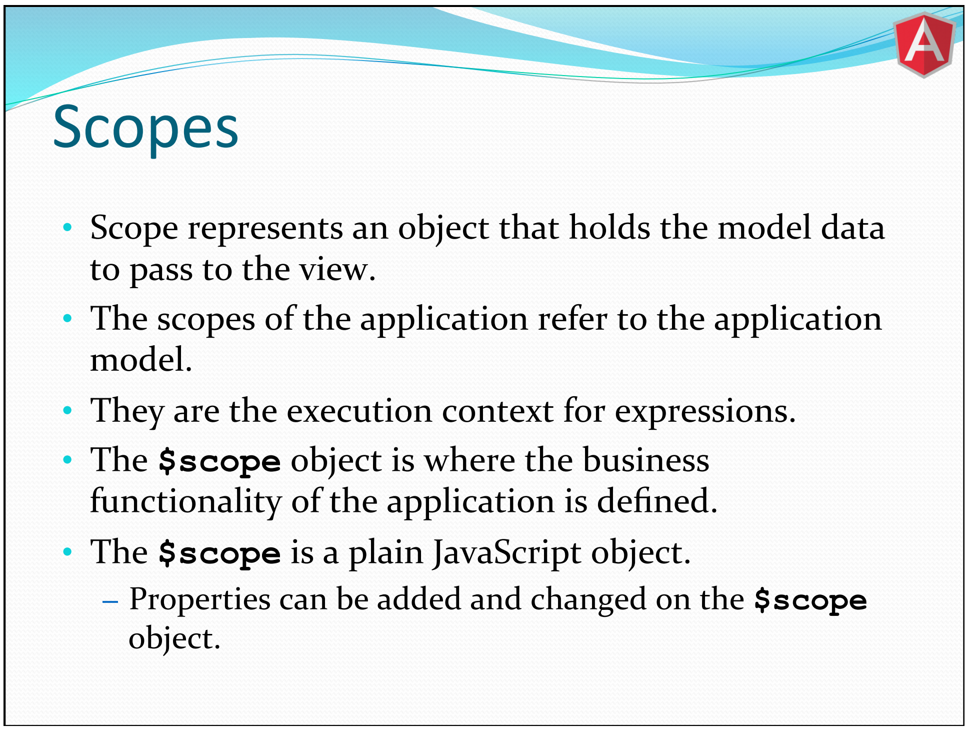
```
angular.module()
```

- The method accepts two parameters
  - Name of the module
  - And a list of dependencies (modules) /injectibles

```
angular.module("myNewApp", []);
```

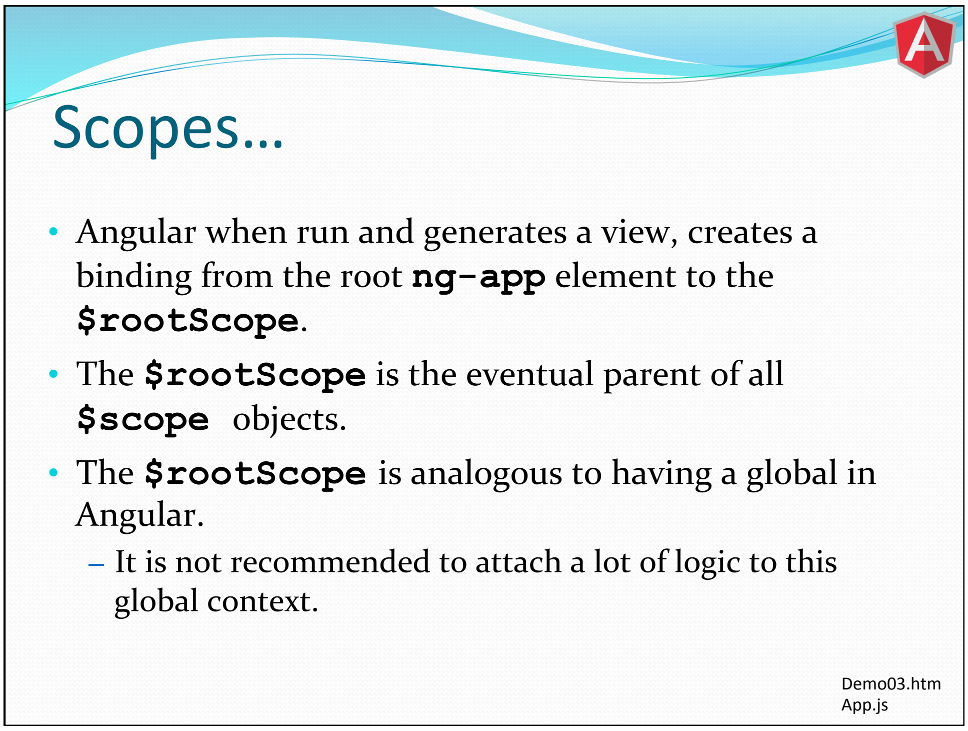
- The module can be referenced by using just the name parameter

```
angular.module("myNewApp");
```



Scopes

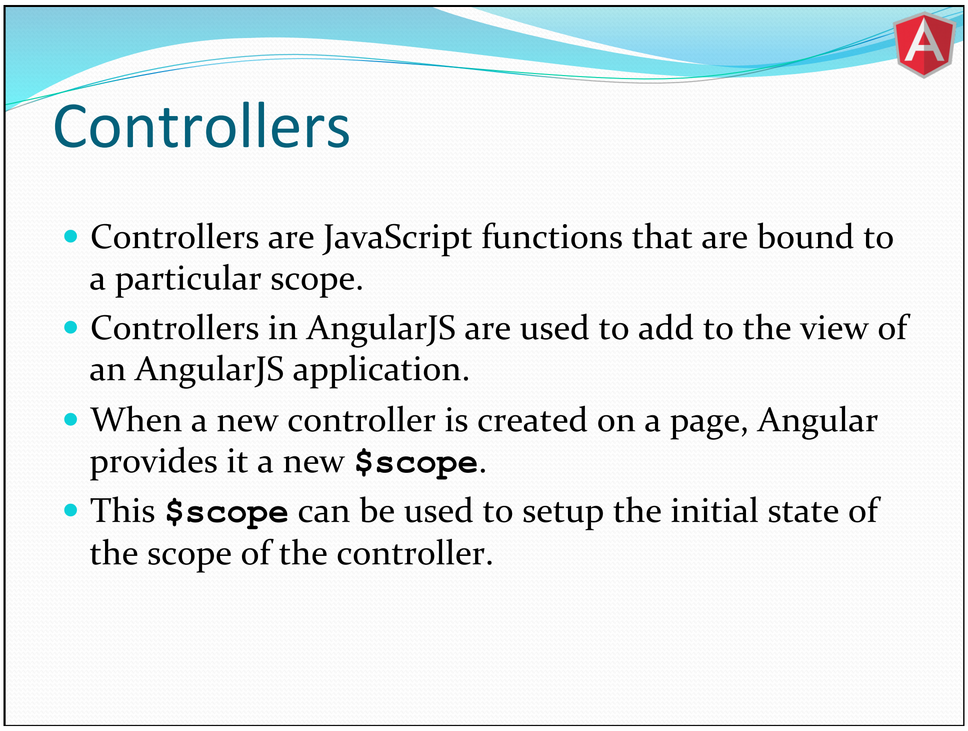
- Scope represents an object that holds the model data to pass to the view.
- The scopes of the application refer to the application model.
- They are the execution context for expressions.
- The **`$scope`** object is where the business functionality of the application is defined.
- The **`$scope`** is a plain JavaScript object.
  - Properties can be added and changed on the **`$scope`** object.



Scopes…

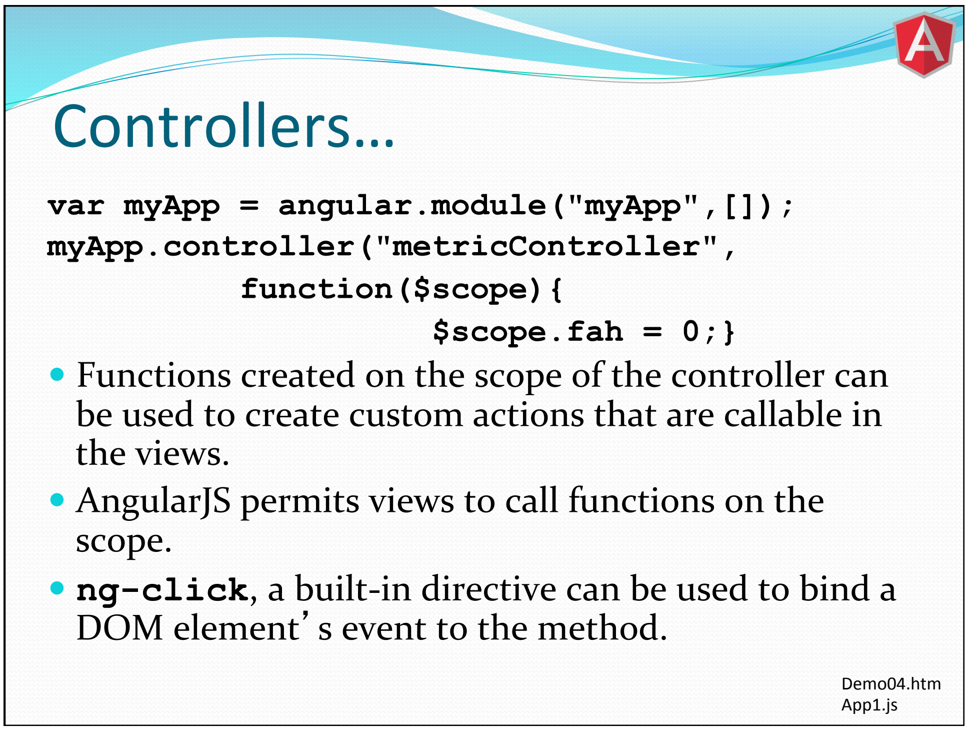
- Angular when run and generates a view, creates a binding from the root **`ng-app`** element to the **`$rootScope`**.
- The **`$rootScope`** is the eventual parent of all **`$scope`** objects.
- The **`$rootScope`** is analogous to having a global in Angular.
  - It is not recommended to attach a lot of logic to this global context.

Demo03.htm
App.js



Controllers

- Controllers are JavaScript functions that are bound to a particular scope.
- Controllers in AngularJS are used to add to the view of an AngularJS application.
- When a new controller is created on a page, Angular provides it a new **`$scope`**.
- This **`$scope`** can be used to setup the initial state of the scope of the controller.

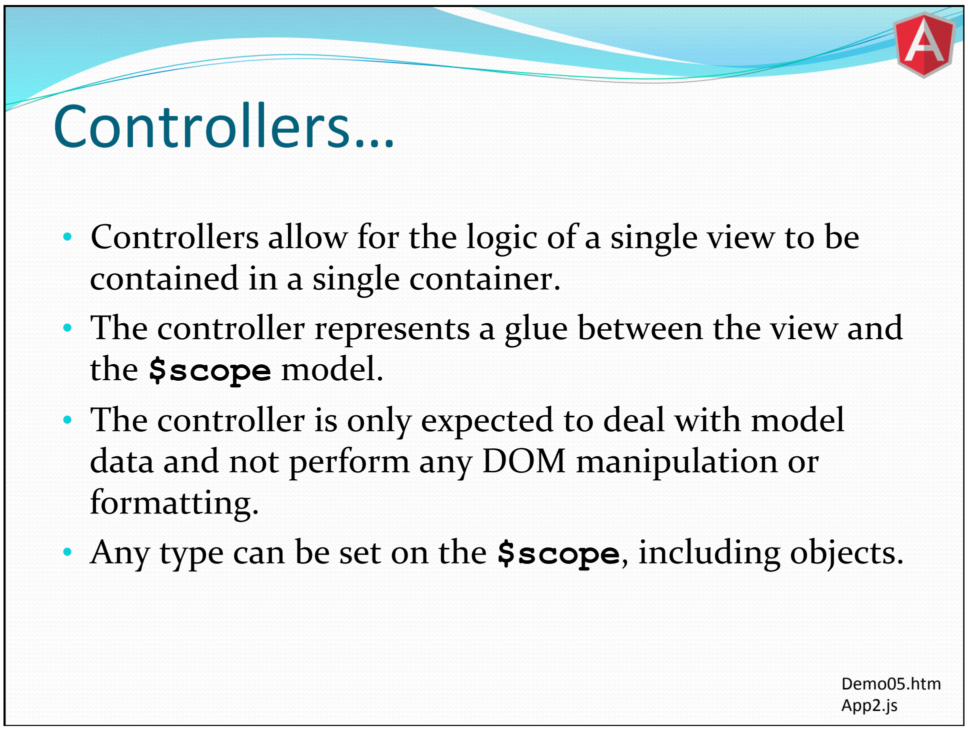


Controllers…

```
var myApp = angular.module("myApp",[]);
myApp.controller("metricController",
          function($scope){
                    $scope.fah = 0;}
```

- Functions created on the scope of the controller can be used to create custom actions that are callable in the views.
- AngularJS permits views to call functions on the scope.
- **`ng-click`**, a built-in directive can be used to bind a DOM element' s event to the method.

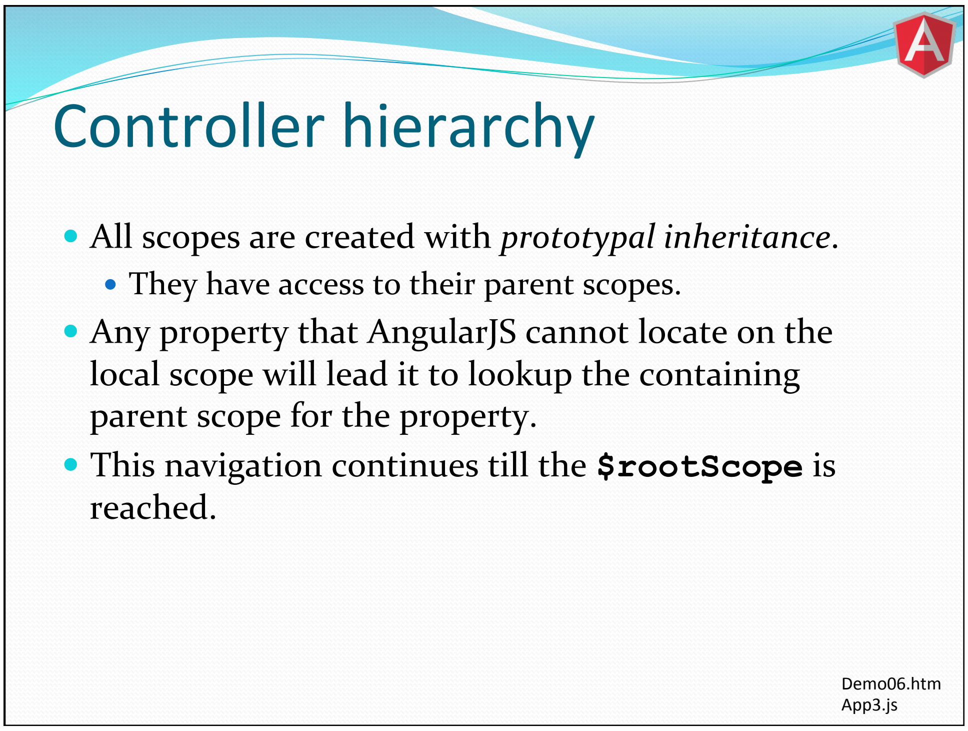
Demo04.htm
App1.js



Controllers…

- Controllers allow for the logic of a single view to be contained in a single container.
- The controller represents a glue between the view and the **`$scope`** model.
- The controller is only expected to deal with model data and not perform any DOM manipulation or formatting.
- Any type can be set on the **`$scope`**, including objects.

Demo05.htm
App2.js



Controller hierarchy

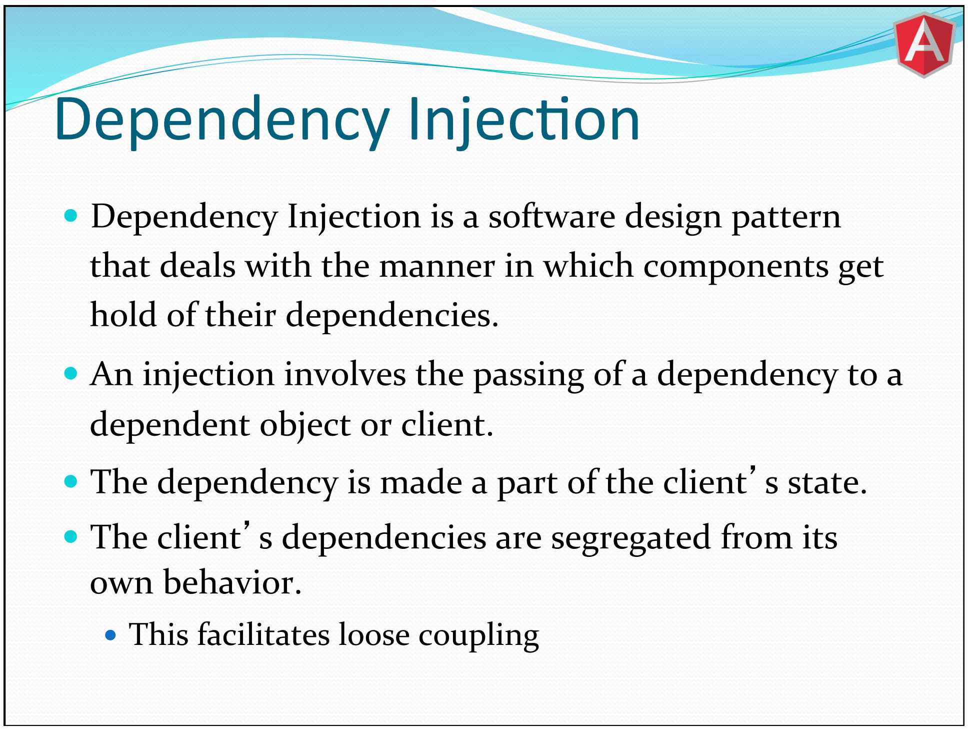
- All scopes are created with *prototypal inheritance*.
  - They have access to their parent scopes.
- Any property that AngularJS cannot locate on the local scope will lead it to lookup the containing parent scope for the property.
- This navigation continues till the **`$rootScope`** is reached.

Demo06.htm
App3.js

Expressions

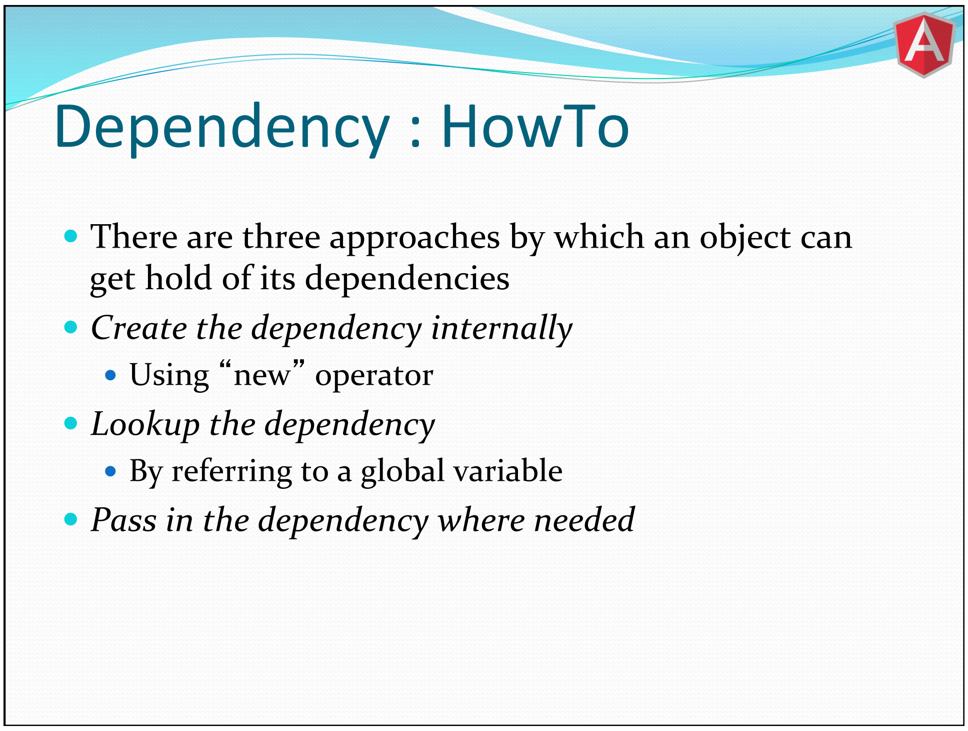
- Expressions are used extensively by AngularJS.
- The `{{ }}` notation for depicting a variable attached to a `$scope` is an expression.
- Expressions are loosely similar to the result of an `eval()` in JavaScript
- All expressions are executed in the context of the scope and have access to local `$scope` variables.
- Expressions do not allow control flow functions (if/ else etc)

Dependency Injection



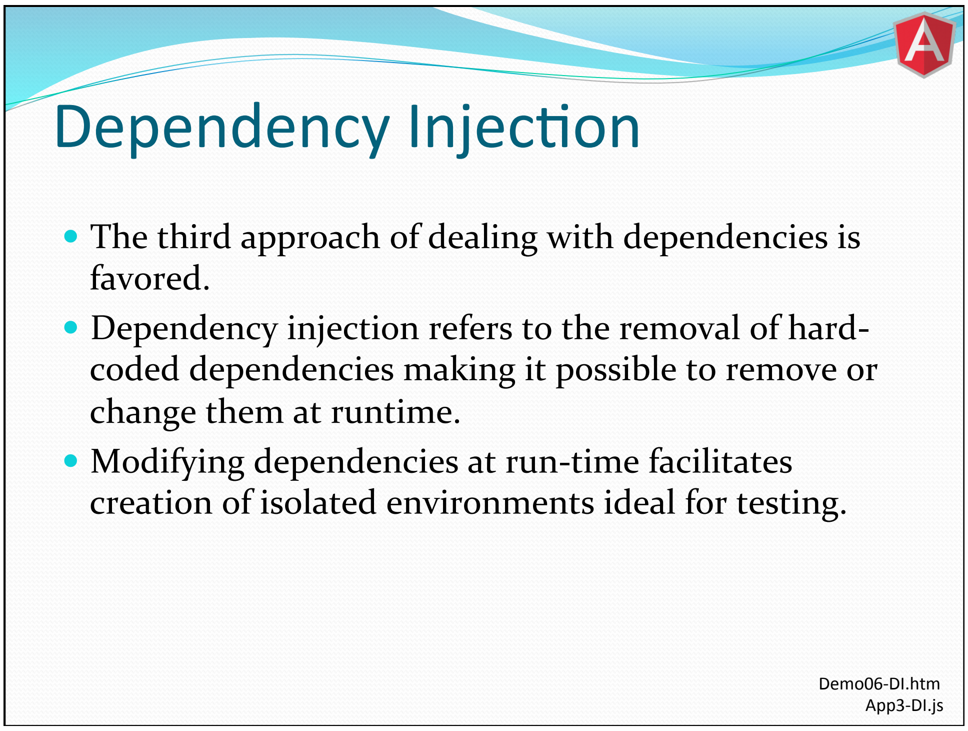
Dependency Injection

- Dependency Injection is a software design pattern that deals with the manner in which components get hold of their dependencies.
- An injection involves the passing of a dependency to a dependent object or client.
- The dependency is made a part of the client's state.
- The client's dependencies are segregated from its own behavior.
  - This facilitates loose coupling



Dependency : HowTo

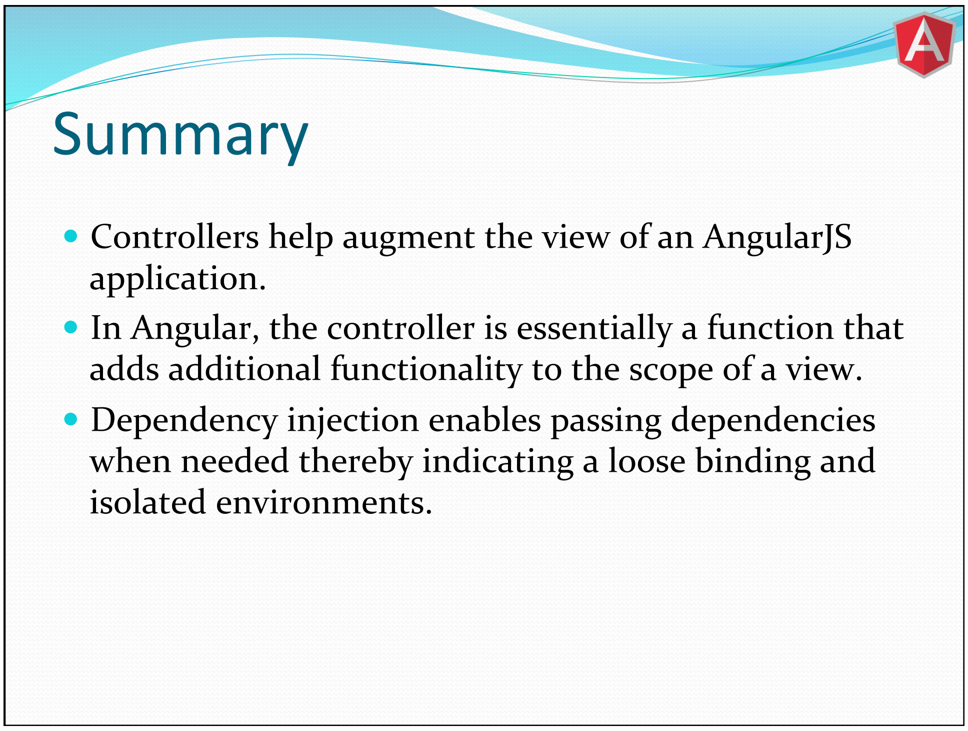
- There are three approaches by which an object can get hold of its dependencies
- *Create the dependency internally*
  - Using "new" operator
- *Lookup the dependency*
  - By referring to a global variable
- *Pass in the dependency where needed*



Dependency Injection

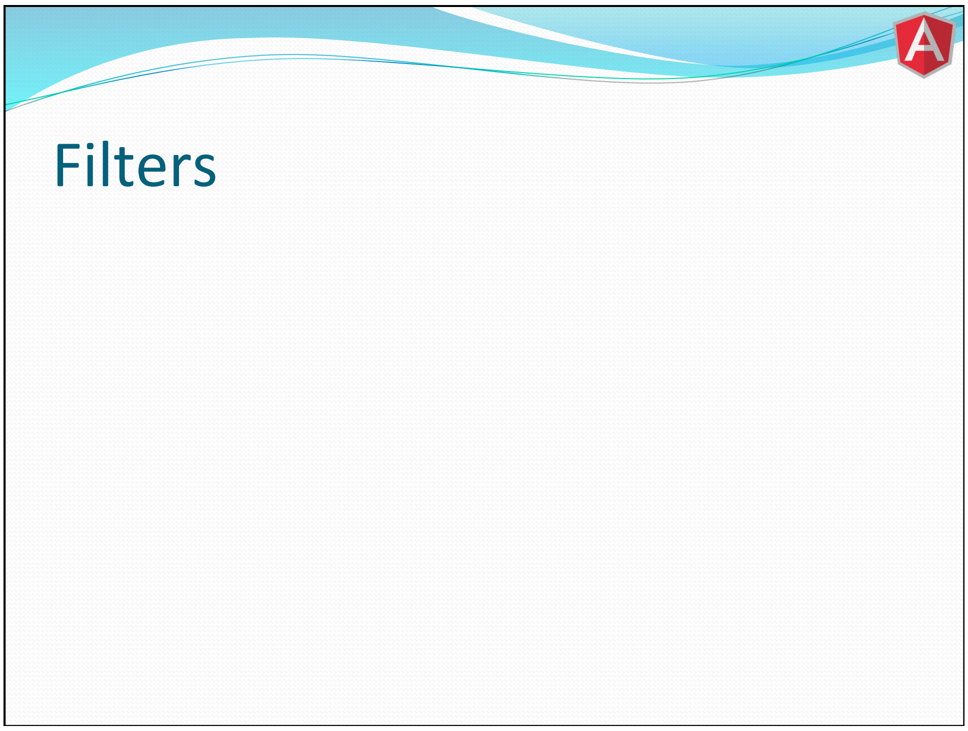
- The third approach of dealing with dependencies is favored.
- Dependency injection refers to the removal of hard-coded dependencies making it possible to remove or change them at runtime.
- Modifying dependencies at run-time facilitates creation of isolated environments ideal for testing.

Demo06-DI.htm
App3-DI.js

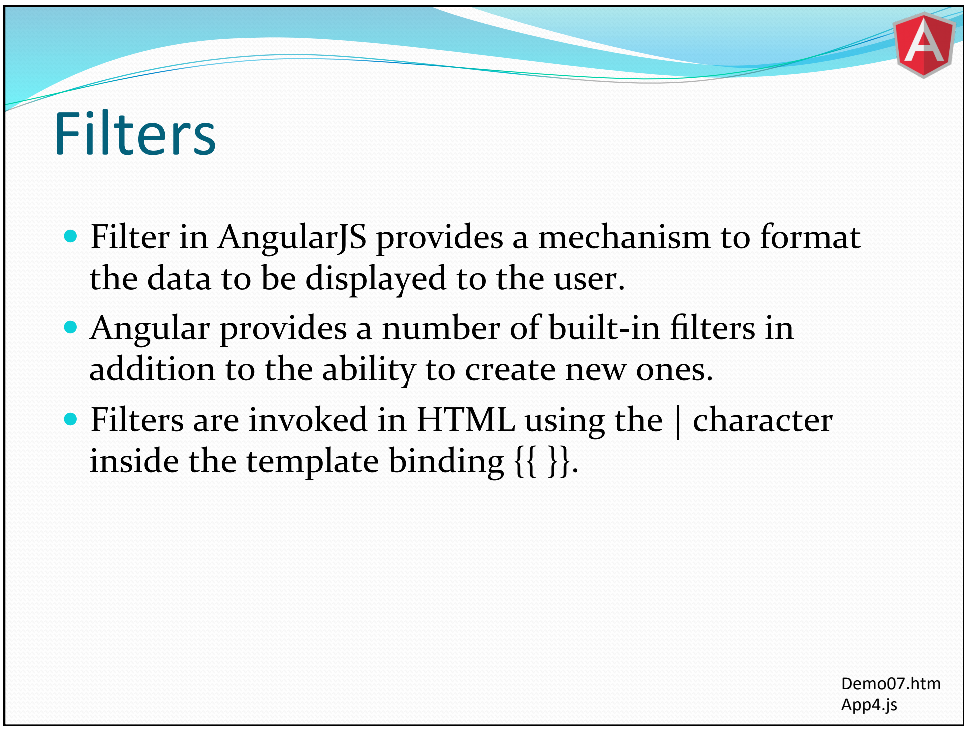


Summary

- Controllers help augment the view of an AngularJS application.
- In Angular, the controller is essentially a function that adds additional functionality to the scope of a view.
- Dependency injection enables passing dependencies when needed thereby indicating a loose binding and isolated environments.



Filters



Filters

- Filter in AngularJS provides a mechanism to format the data to be displayed to the user.
- Angular provides a number of built-in filters in addition to the ability to create new ones.
- Filters are invoked in HTML using the | character inside the template binding {{ }}.

Demo07.htm
App4.js

Filters - Currency

- The currency filter is used to format number as currency.
- It provides us the option of displaying a symbol or an identifier.
- The default currency option is that of the current locale.

```
{{ amount | currency:"&#8377" }}
```

Demo08.htm

Filters - Date

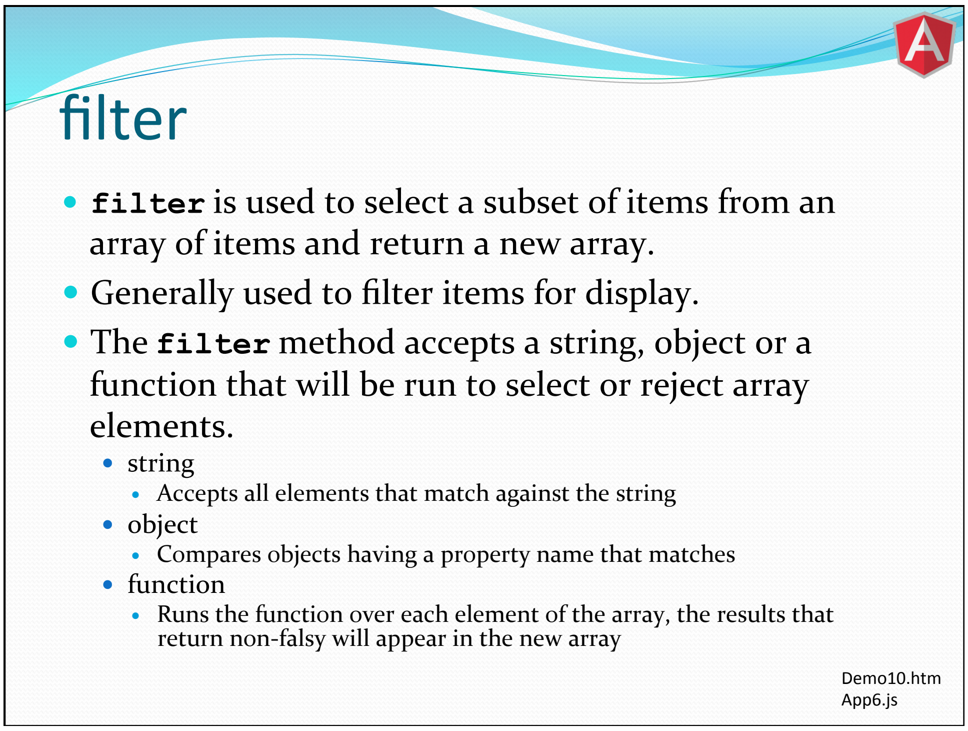
- The date filter allows to format the date in a required format style.
- It provides several built-in options.

```
{{ today | date:'medium'}}
```

- The date formatter enables customization of the date format.

```
{{ today | date:'EEEE, MMMM d, yyyy'}}
```

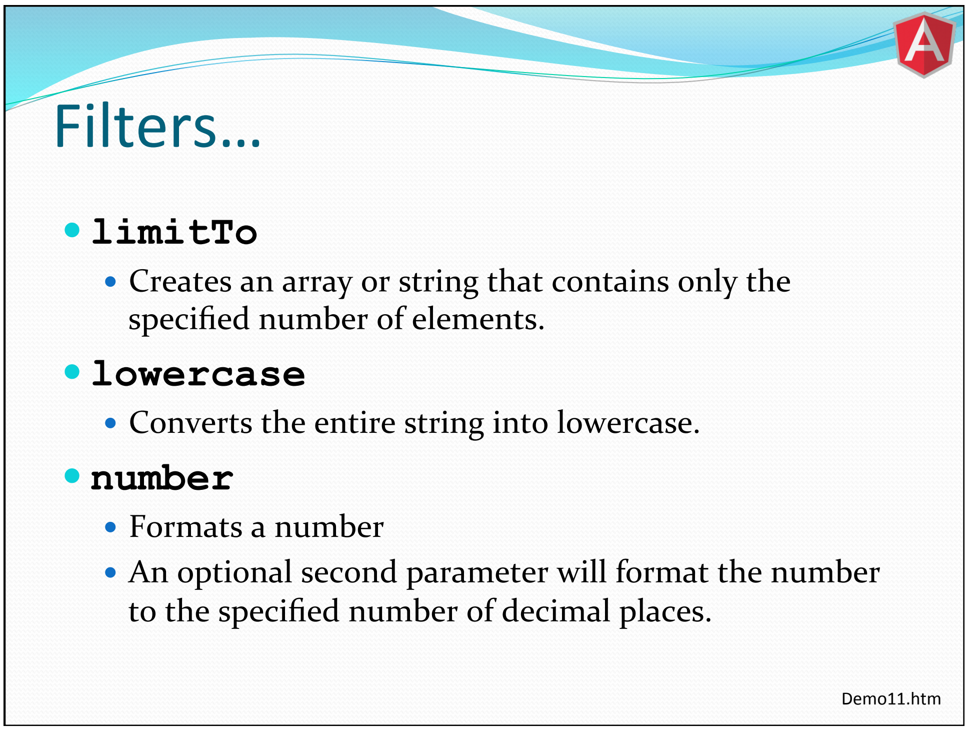
Demo09.htm
App5.js



filter

- `filter` is used to select a subset of items from an array of items and return a new array.
- Generally used to filter items for display.
- The `filter` method accepts a string, object or a function that will be run to select or reject array elements.
  - string
    - Accepts all elements that match against the string
  - object
    - Compares objects having a property name that matches
  - function
    - Runs the function over each element of the array, the results that return non-falsy will appear in the new array

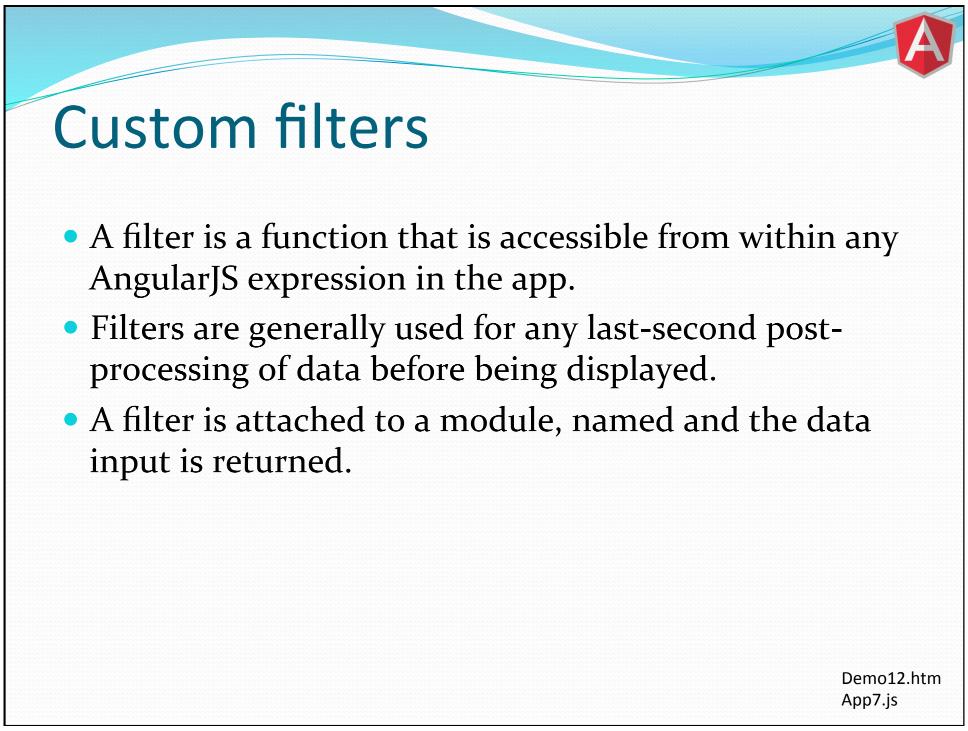
Demo10.htm
App6.js



Filters…

- **`limitTo`**
  - Creates an array or string that contains only the specified number of elements.
- **`lowercase`**
  - Converts the entire string into lowercase.
- **`number`**
  - Formats a number
  - An optional second parameter will format the number to the specified number of decimal places.

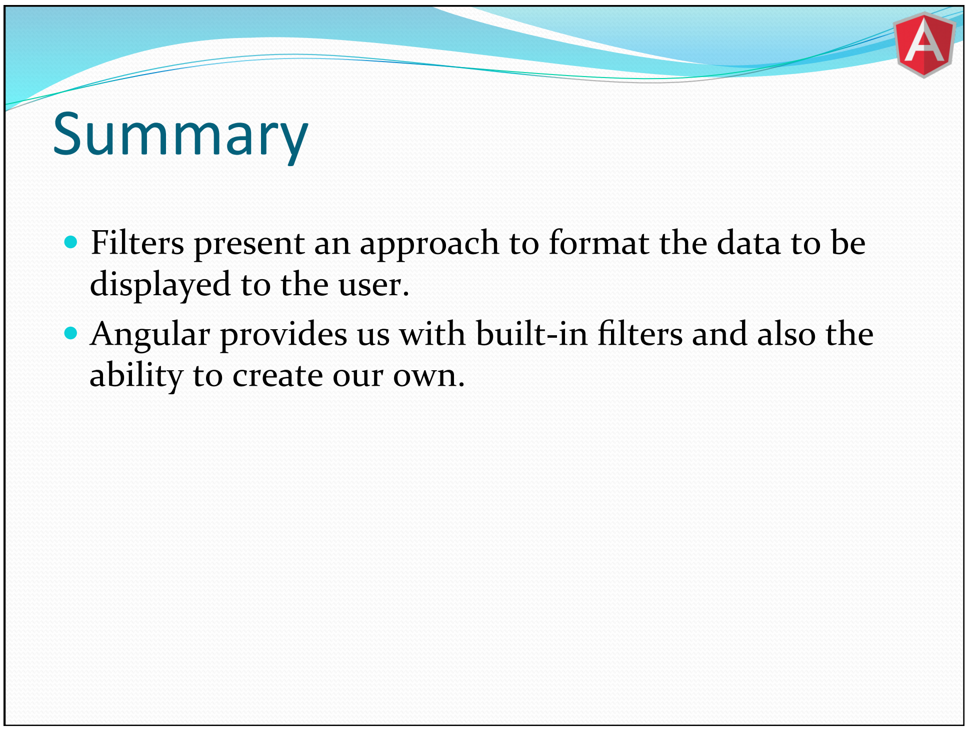
Demo11.htm



Custom filters

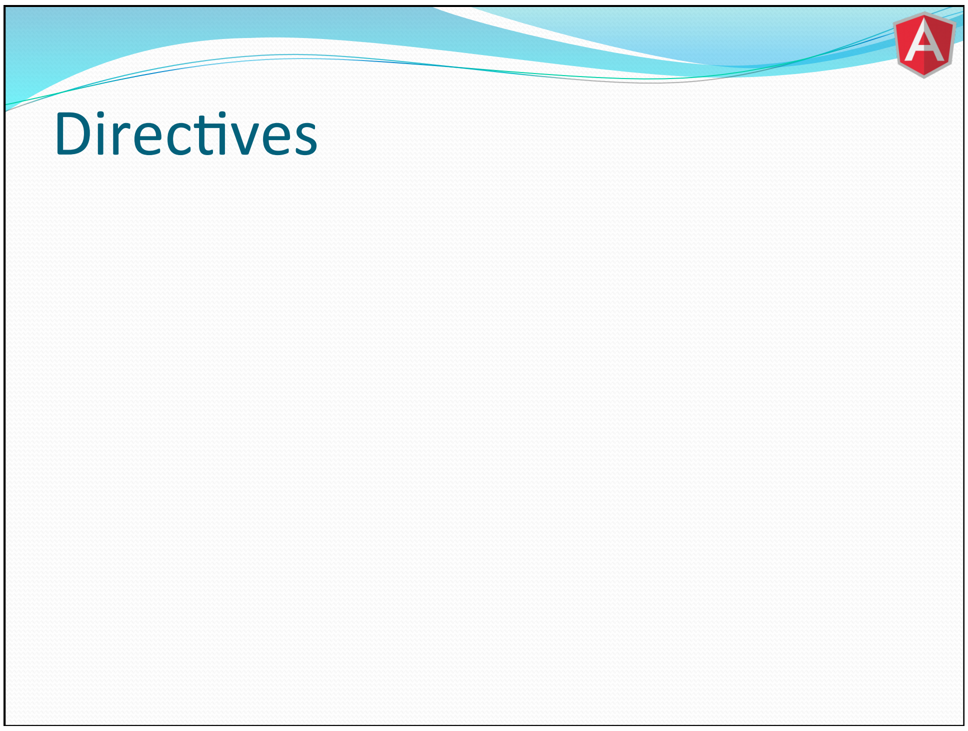
- A filter is a function that is accessible from within any AngularJS expression in the app.
- Filters are generally used for any last-second post-processing of data before being displayed.
- A filter is attached to a module, named and the data input is returned.

Demo12.htm
App7.js

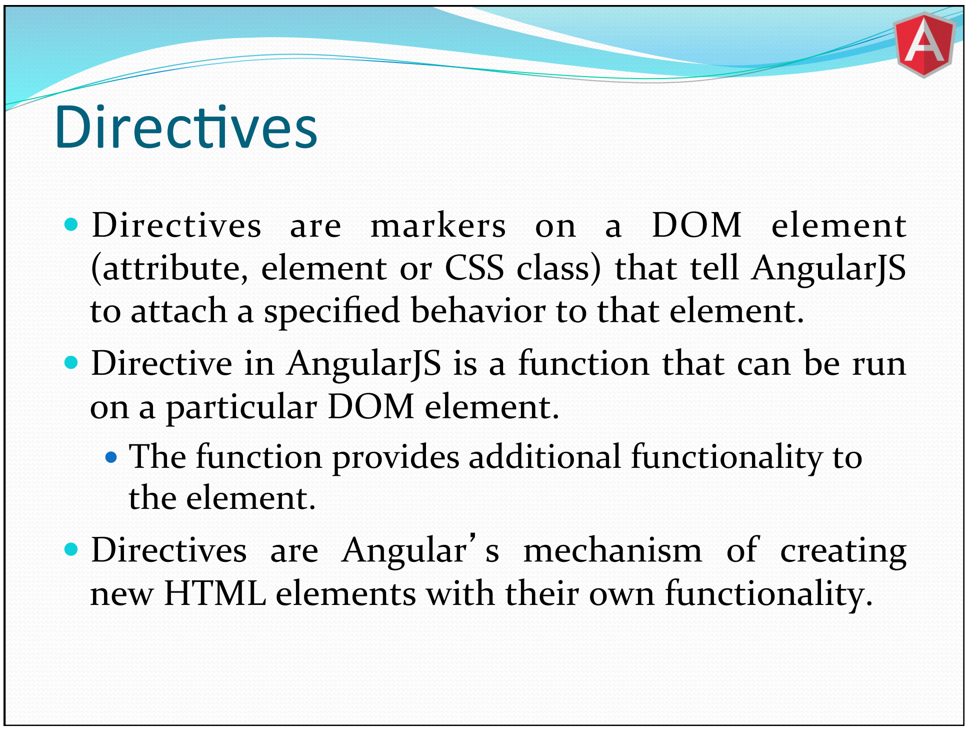


Summary

- Filters present an approach to format the data to be displayed to the user.
- Angular provides us with built-in filters and also the ability to create our own.

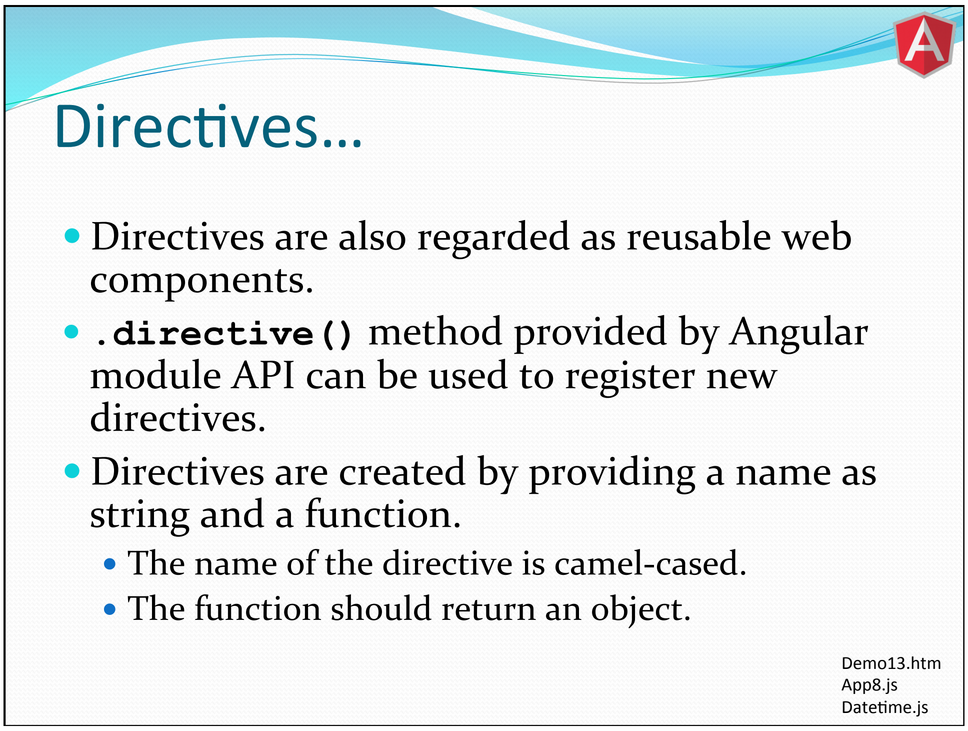


Directives



Directives

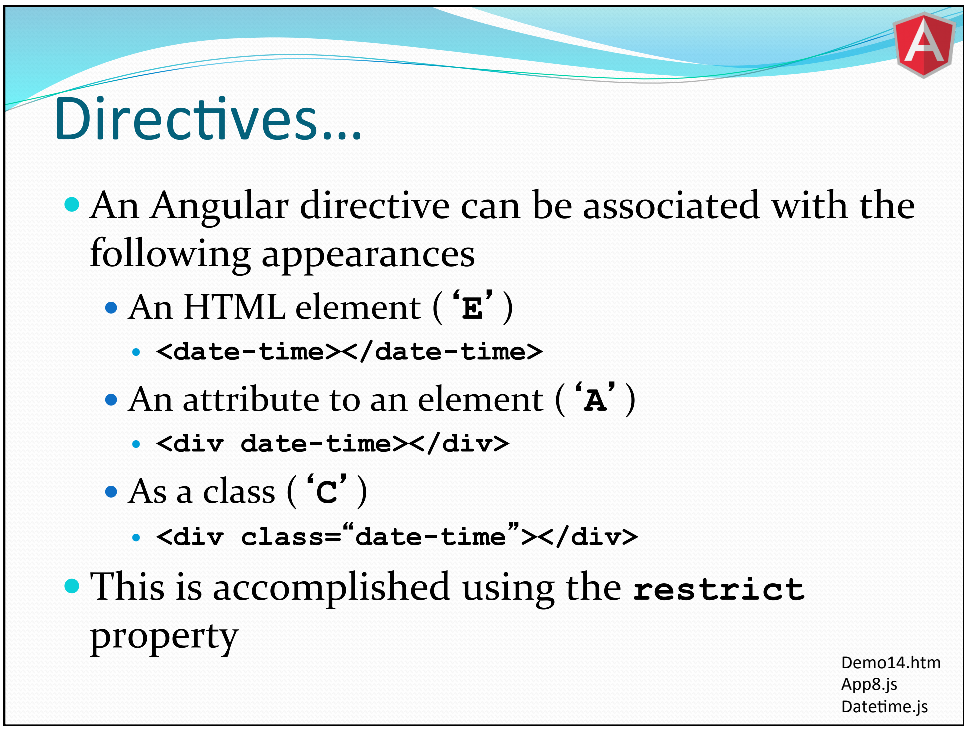
- Directives are markers on a DOM element (attribute, element or CSS class) that tell AngularJS to attach a specified behavior to that element.
- Directive in AngularJS is a function that can be run on a particular DOM element.
  - The function provides additional functionality to the element.
- Directives are Angular's mechanism of creating new HTML elements with their own functionality.



Directives…

- Directives are also regarded as reusable web components.
- **`.directive()`** method provided by Angular module API can be used to register new directives.
- Directives are created by providing a name as string and a function.
  - The name of the directive is camel-cased.
  - The function should return an object.

Demo13.htm
App8.js
Datetime.js



Directives…

- An Angular directive can be associated with the following appearances
  - An HTML element (**'E'**)
    - **`<date-time></date-time>`**
  - An attribute to an element (**'A'**)
    - **`<div date-time></div>`**
  - As a class (**'C'**)
    - **`<div class="date-time"></div>`**
- This is accomplished using the **`restrict`** property

Demo14.htm
App8.js
Datetime.js

Directives…

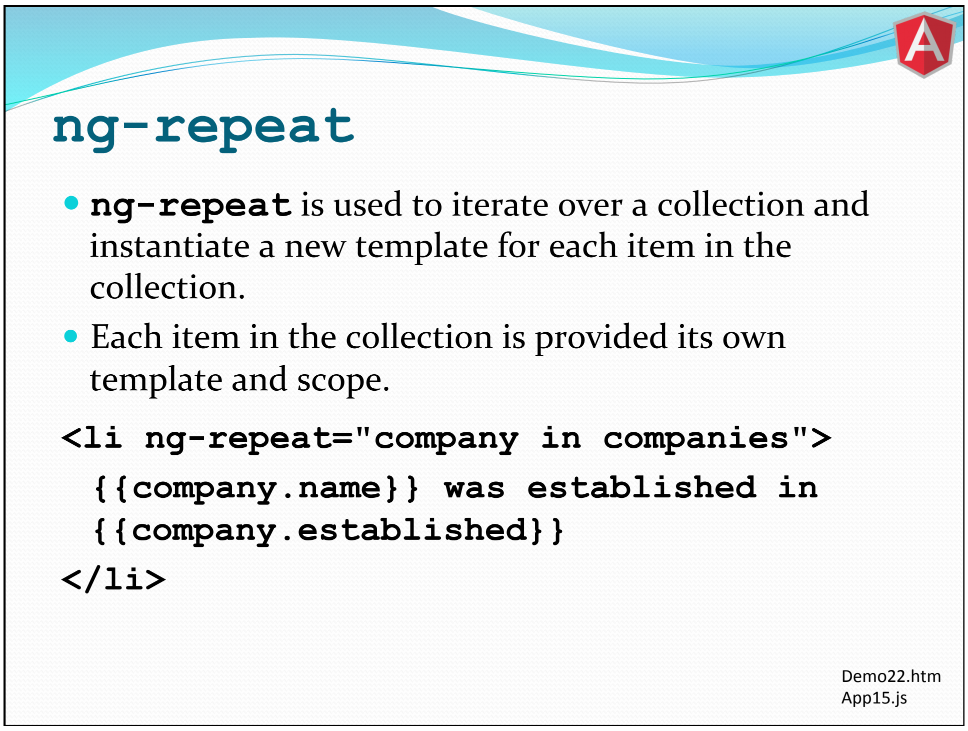
- The default behavior of Angular is to nest the HTML provided by the **`template`** property inside the custom tag.
- The custom element can be removed from the generated DOM with the help of **`replace`** property with a value set to **`true`**.

```
.directive("dateTime",function(){
      return {
            restrict : 'EAC',
            replace : true,
            template : getDateTime
      }
```

Demo15.htm
App8.js
Datetime1.js

Basic **ng** Attribute directives

- **`ng-disabled`**
- **`ng-readonly`**
- **`ng-checked`**
- **`ng-selected`**
- **`ng-class`**
- **`ng-style`**
- **`ng-src`**
- **`ng-href`**

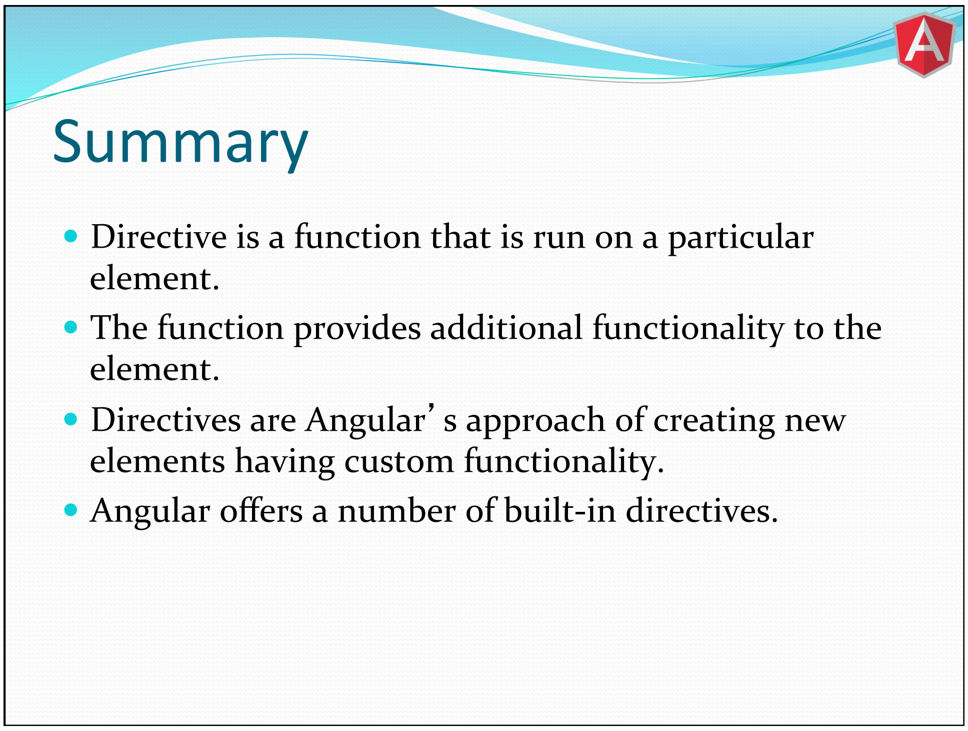


ng-repeat

- **ng-repeat** is used to iterate over a collection and instantiate a new template for each item in the collection.
- Each item in the collection is provided its own template and scope.

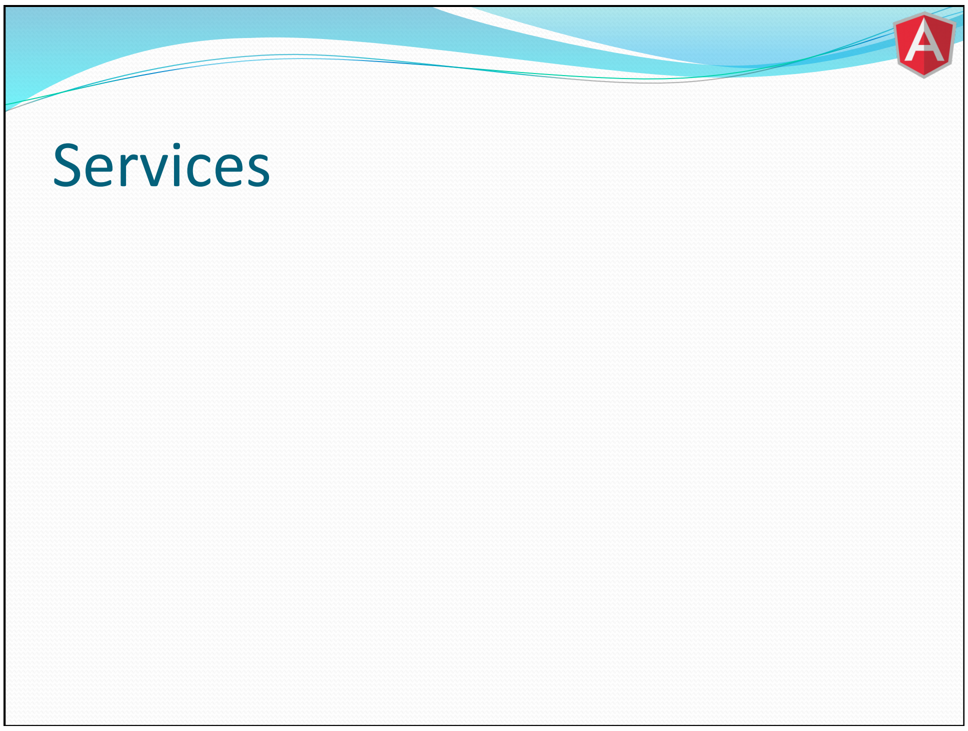
```
<li ng-repeat="company in companies">
  {{company.name}} was established in
  {{company.established}}
</li>
```

Demo22.htm
App15.js

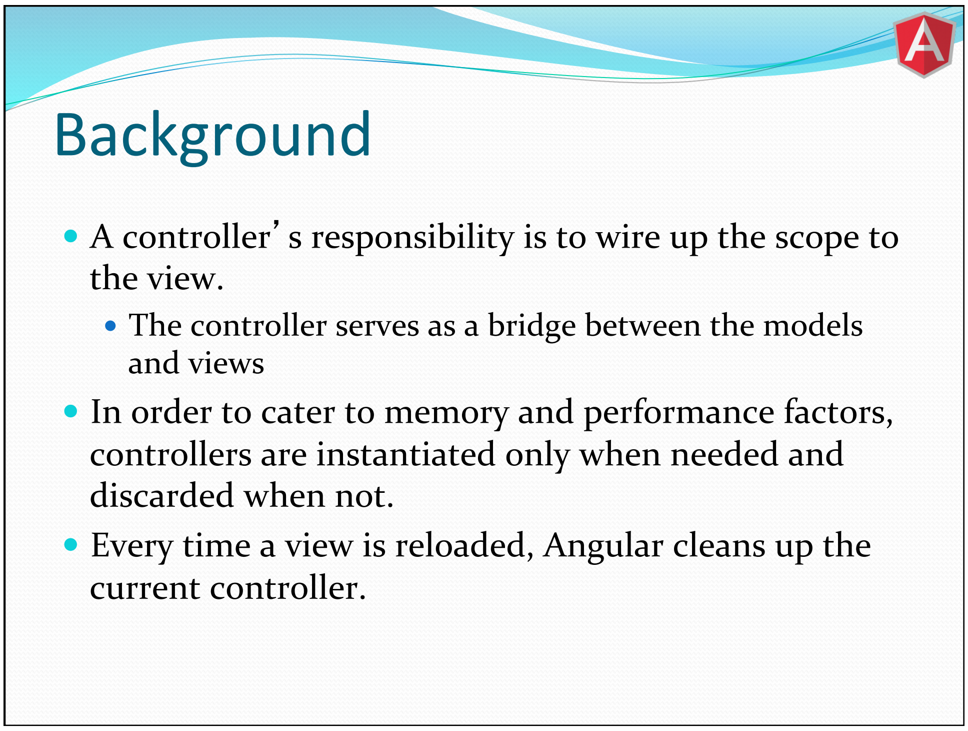


Summary

- Directive is a function that is run on a particular element.
- The function provides additional functionality to the element.
- Directives are Angular's approach of creating new elements having custom functionality.
- Angular offers a number of built-in directives.

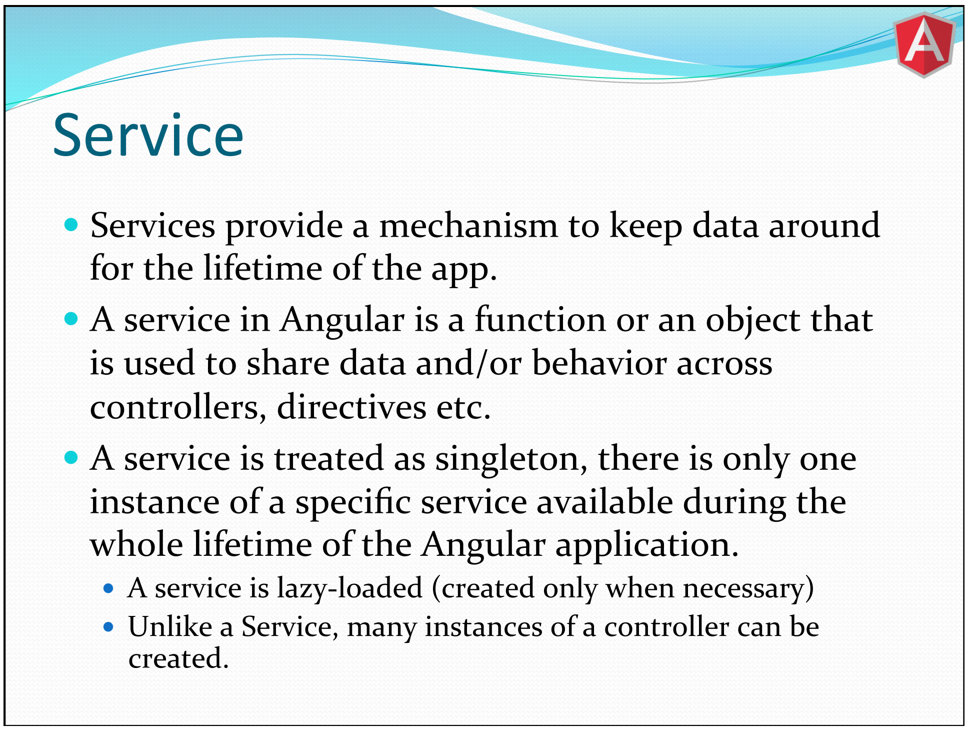


Services



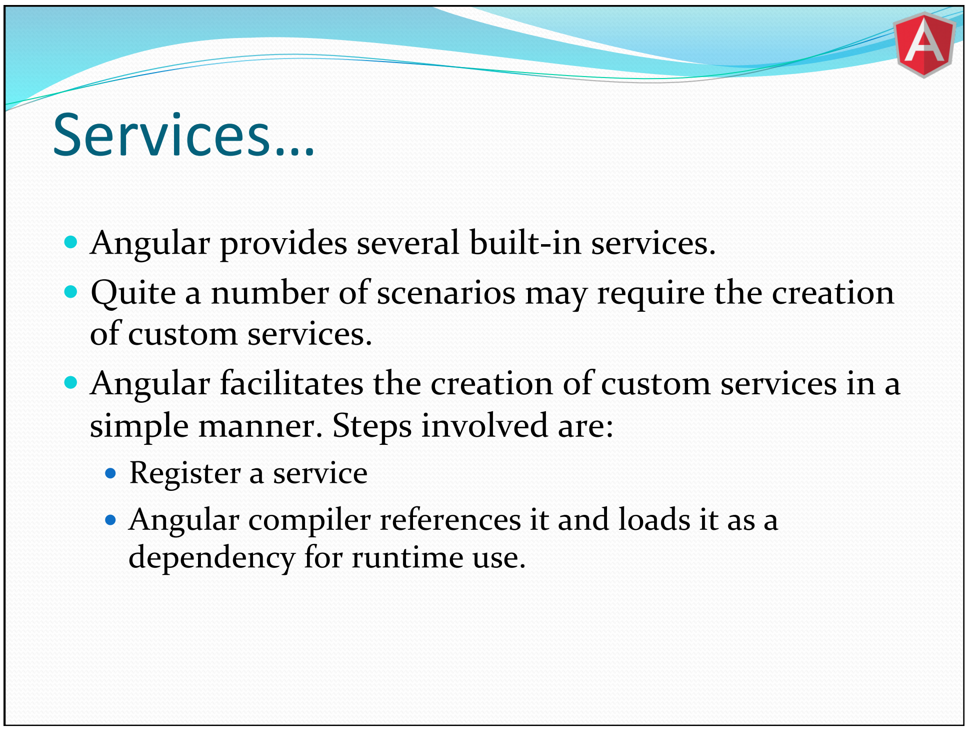
Background

- A controller's responsibility is to wire up the scope to the view.
  - The controller serves as a bridge between the models and views
- In order to cater to memory and performance factors, controllers are instantiated only when needed and discarded when not.
- Every time a view is reloaded, Angular cleans up the current controller.



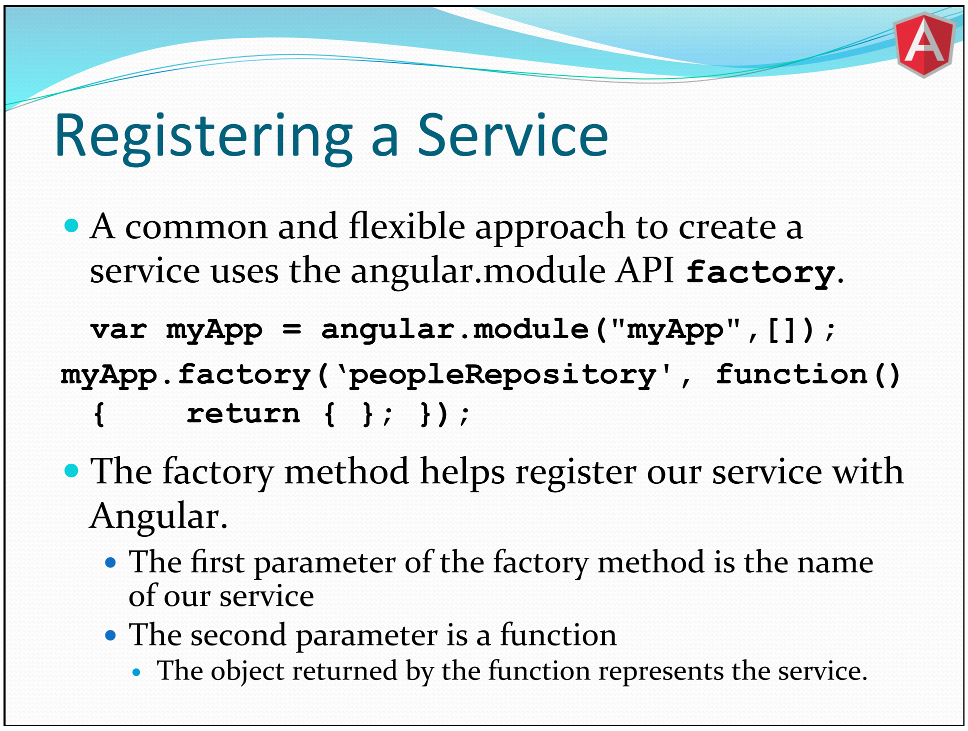
Service

- Services provide a mechanism to keep data around for the lifetime of the app.
- A service in Angular is a function or an object that is used to share data and/or behavior across controllers, directives etc.
- A service is treated as singleton, there is only one instance of a specific service available during the whole lifetime of the Angular application.
  - A service is lazy-loaded (created only when necessary)
  - Unlike a Service, many instances of a controller can be created.



Services…

- Angular provides several built-in services.
- Quite a number of scenarios may require the creation of custom services.
- Angular facilitates the creation of custom services in a simple manner. Steps involved are:
  - Register a service
  - Angular compiler references it and loads it as a dependency for runtime use.

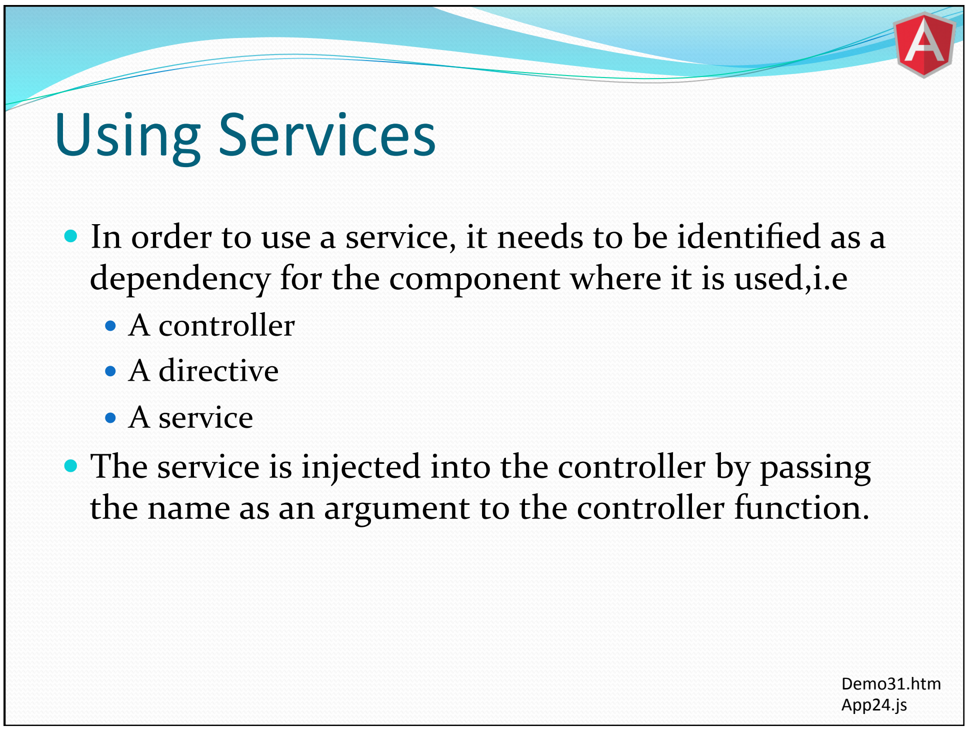


Registering a Service

- A common and flexible approach to create a service uses the angular.module API **`factory`**.

```
  var myApp = angular.module("myApp",[]);
myApp.factory(`peopleRepository', function()
  {     return { }; });
```

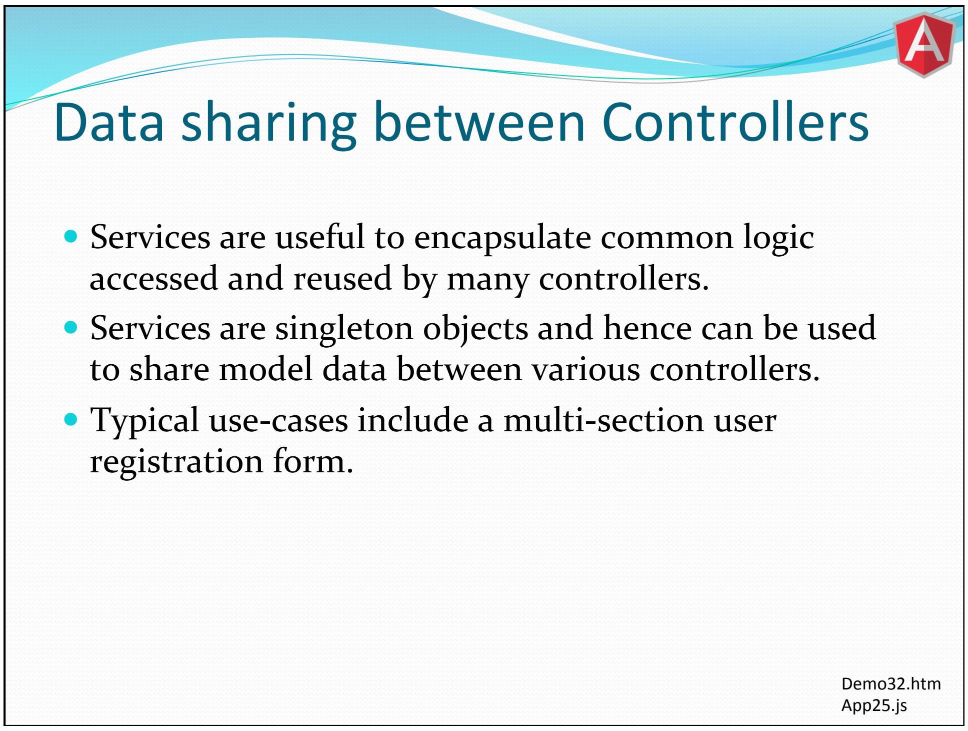
- The factory method helps register our service with Angular.
  - The first parameter of the factory method is the name of our service
  - The second parameter is a function
    - The object returned by the function represents the service.



Using Services

- In order to use a service, it needs to be identified as a dependency for the component where it is used,i.e
  - A controller
  - A directive
  - A service
- The service is injected into the controller by passing the name as an argument to the controller function.

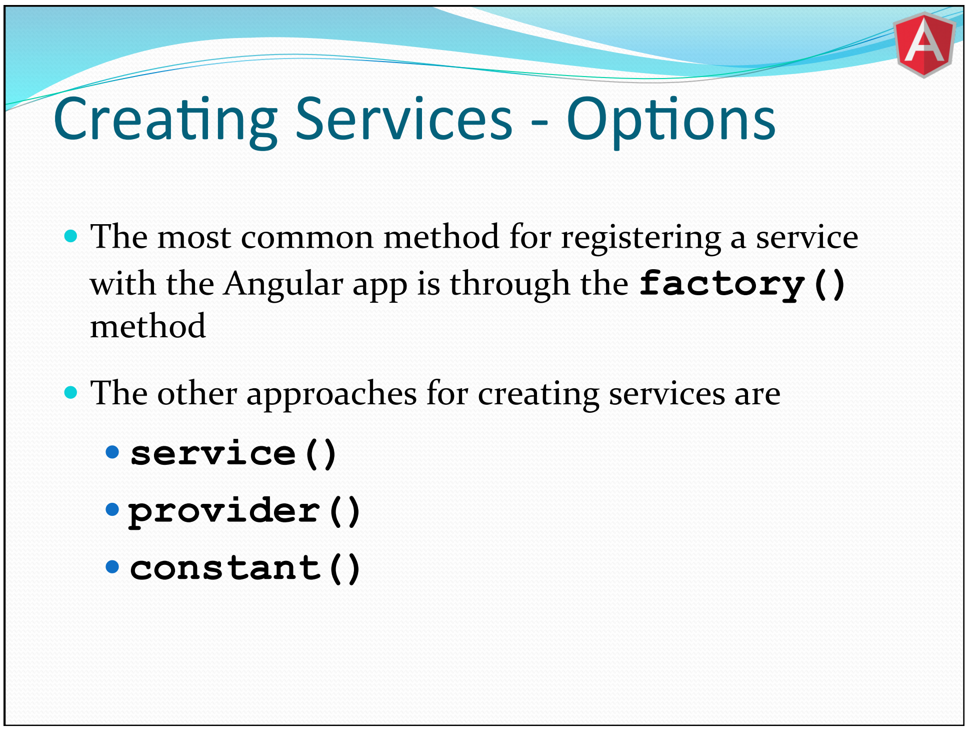
Demo31.htm
App24.js



Data sharing between Controllers

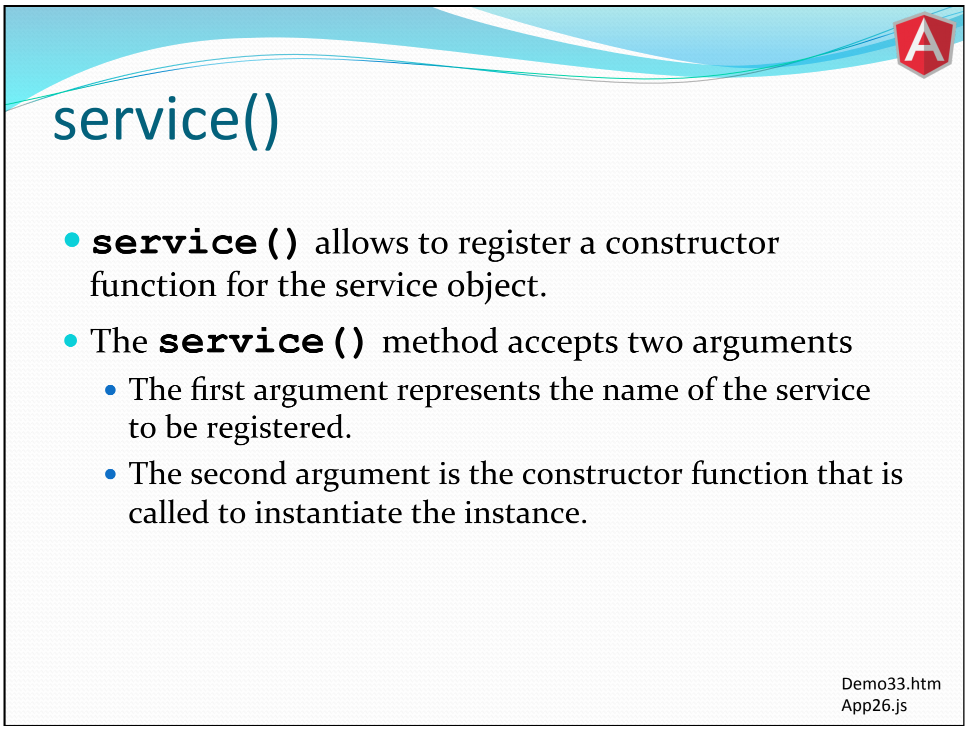
- Services are useful to encapsulate common logic accessed and reused by many controllers.
- Services are singleton objects and hence can be used to share model data between various controllers.
- Typical use-cases include a multi-section user registration form.

Demo32.htm
App25.js



Creating Services - Options

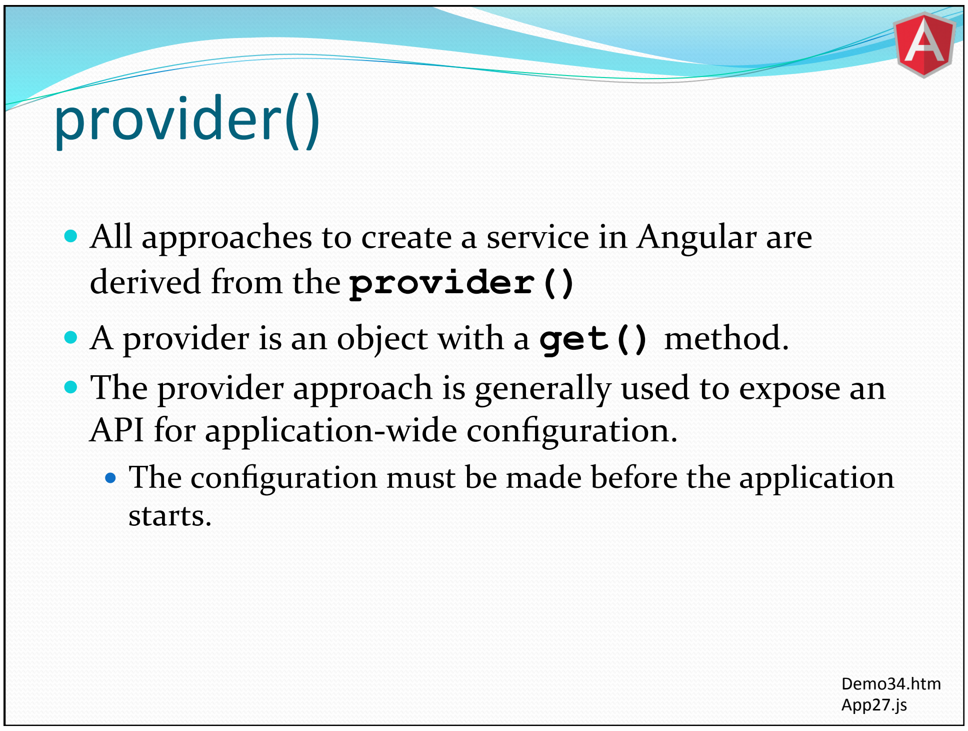
- The most common method for registering a service with the Angular app is through the **`factory()`** method
- The other approaches for creating services are
  - **`service()`**
  - **`provider()`**
  - **`constant()`**



service()

- **`service()`** allows to register a constructor function for the service object.
- The **`service()`** method accepts two arguments
  - The first argument represents the name of the service to be registered.
  - The second argument is the constructor function that is called to instantiate the instance.

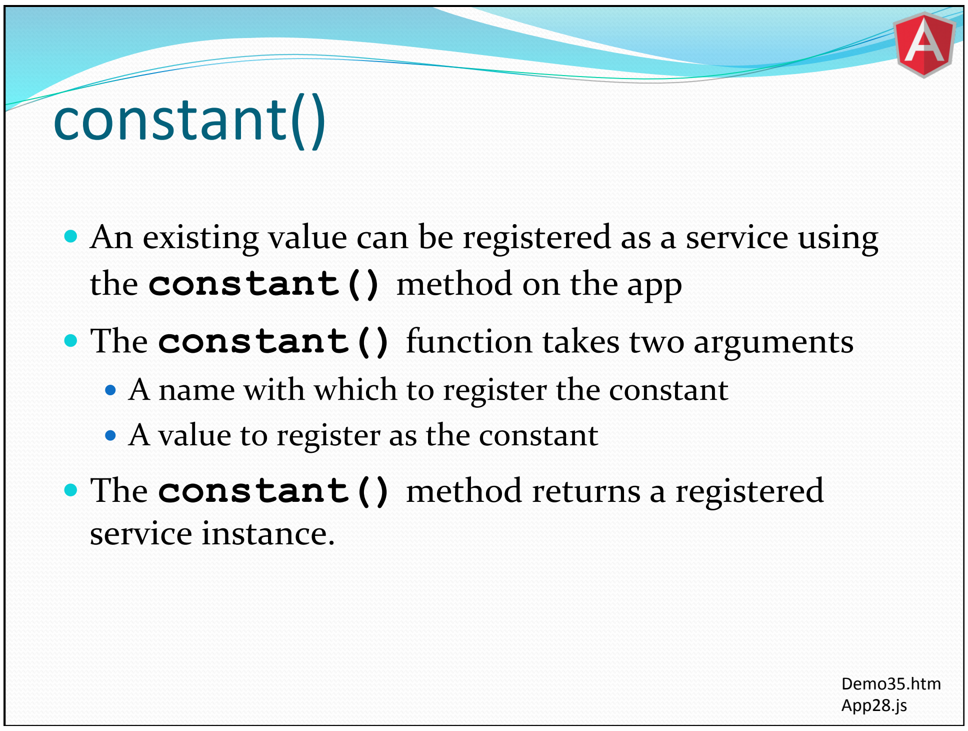
Demo33.htm
App26.js



provider()

- All approaches to create a service in Angular are derived from the **`provider()`**
- A provider is an object with a **`get()`** method.
- The provider approach is generally used to expose an API for application-wide configuration.
  - The configuration must be made before the application starts.

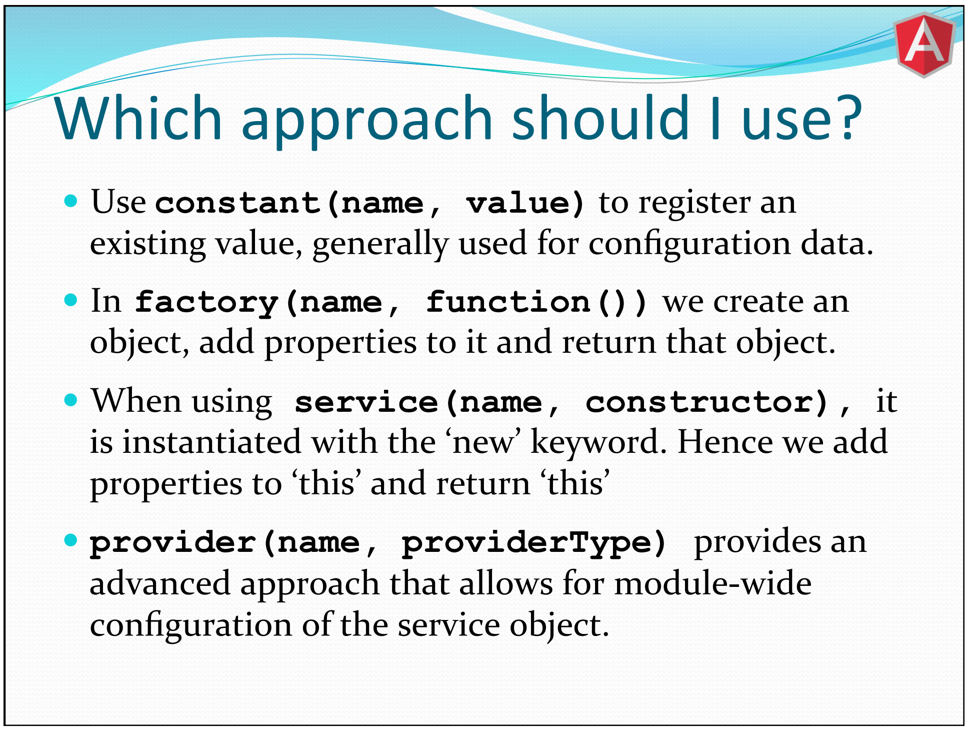
Demo34.htm
App27.js



constant()

- An existing value can be registered as a service using the **`constant()`** method on the app
- The **`constant()`** function takes two arguments
  - A name with which to register the constant
  - A value to register as the constant
- The **`constant()`** method returns a registered service instance.

Demo35.htm
App28.js



Which approach should I use?

- Use **`constant(name, value)`** to register an existing value, generally used for configuration data.
- In **`factory(name, function())`** we create an object, add properties to it and return that object.
- When using **`service(name, constructor),`** it is instantiated with the ‘new’ keyword. Hence we add properties to ‘this’ and return ‘this’
- **`provider(name, providerType)`** provides an advanced approach that allows for module-wide configuration of the service object.

Asynchronous AngularJS

- Callbacks
  - A function in JavaScript is a type that can be passed as an argument.
  - This can be invoked at an appropriate point in time / code.
  - Multiple callbacks that depend on each other can lead to messy and confusing code!

Promise

- Promise is an interface that represents a proxy value.
  - The value is initially unknown
- This promise will return a value in future.
- There can be several states for a promise
  - pending: initial state
  - fulfilled: successful operation
  - rejected: failed operation
  - settled: the Promise is either fulfilled or rejected but not pending

Promise in AngularJS

- Promises in AngularJS are provided by the **`$q`** service.
- $q is an Angular service that helps run functions asynchronously and use their return values when they are done processing.
- Deferred is an object that exposes the promise.
- It has three main methods; **`resolve()`**, **`reject()`** and **`notify()`**

Code listing

```
function sayHelloAsync (name) {
    return $q(function(resolve, reject) {
        setTimeout(function() {
            //Greet when your name is 'deepak'
            if (name == 'lakshman') {
                resolve('Hello, ' + name + '!');
            }
            else {
                reject('Greeting ' + name + ' is 
not allowed.');
            }
        }, 1000);
           });
}
```

Code listing...

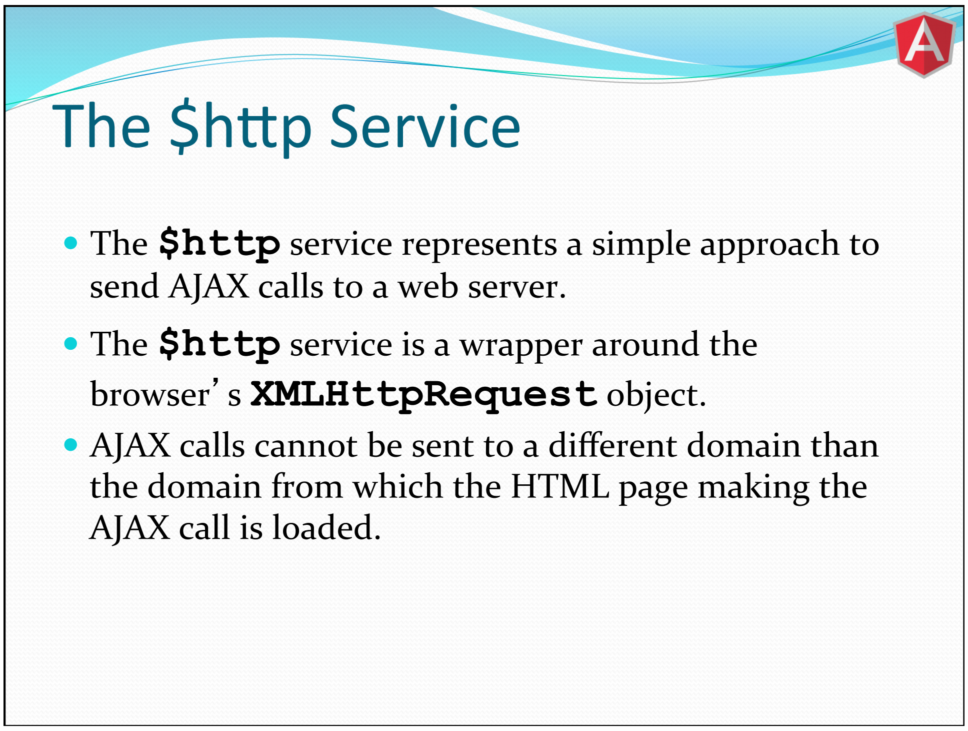
```
var helloPromise = sayHelloAsync('lakshman');

helloPromise.then(function (data) {
    console.log(data);
}, function (error) {
    console.error(data);
})
```

promiseDemo.htm
PromisesDemo.js

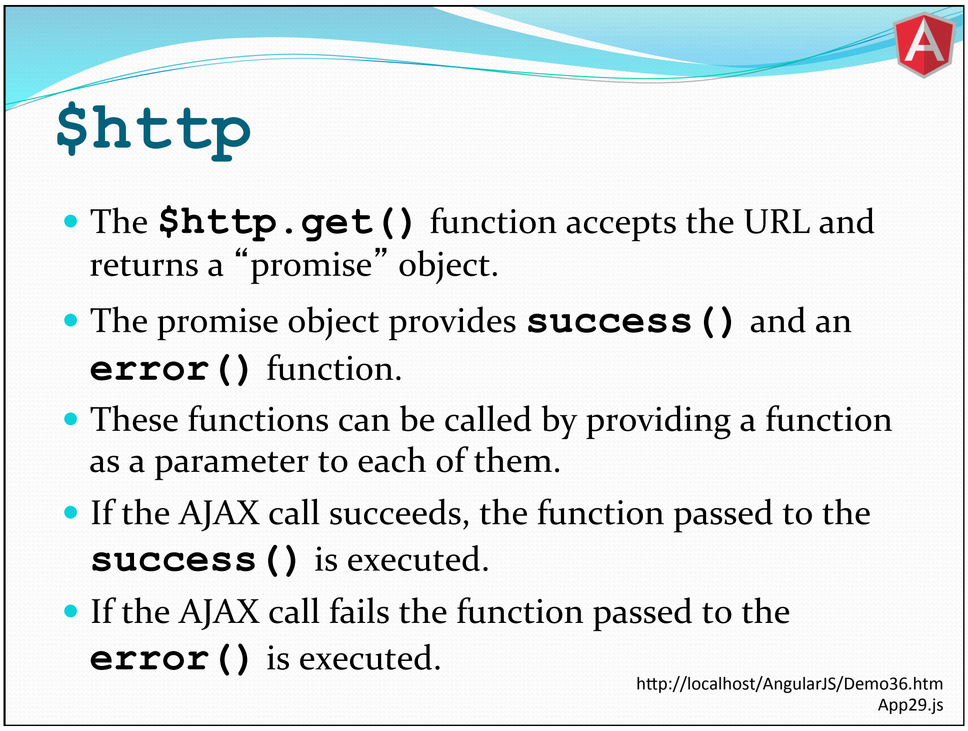
Talking to the external world!

- AngularJS web applications are completely client-side applications.
- Dynamic and responsive web applications can be built using AngularJS which may not involve any back end.
- Angular provides approaches to help integrate the AngularJS app with information from a remote server.



The $http Service

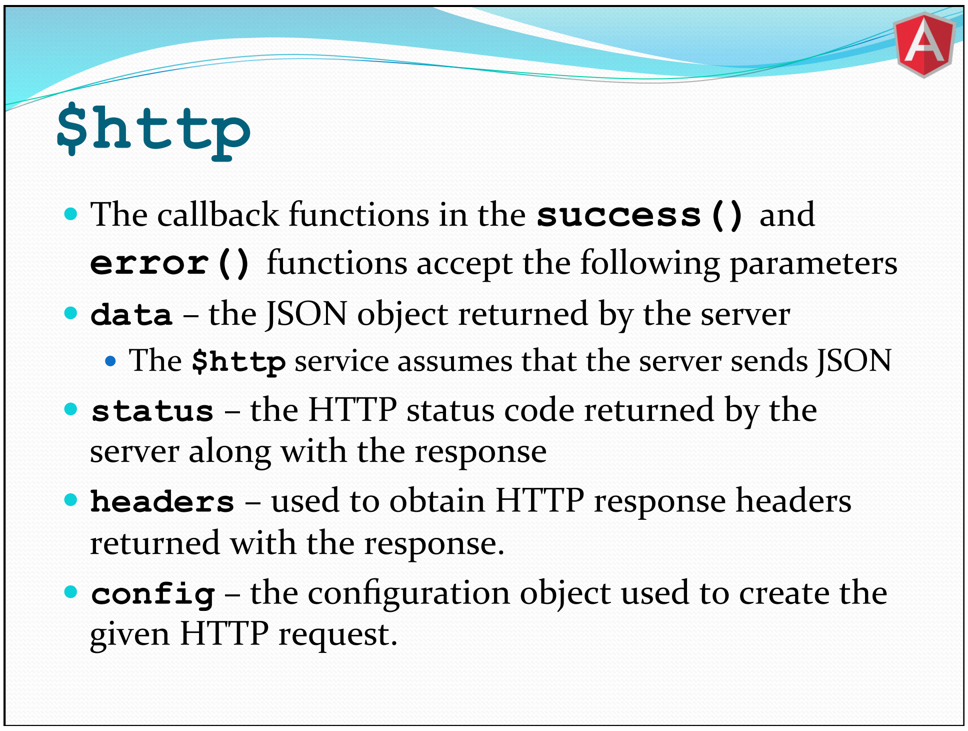
- The **`$http`** service represents a simple approach to send AJAX calls to a web server.
- The **`$http`** service is a wrapper around the browser’ s **`XMLHttpRequest`** object.
- AJAX calls cannot be sent to a different domain than the domain from which the HTML page making the AJAX call is loaded.



`$http`

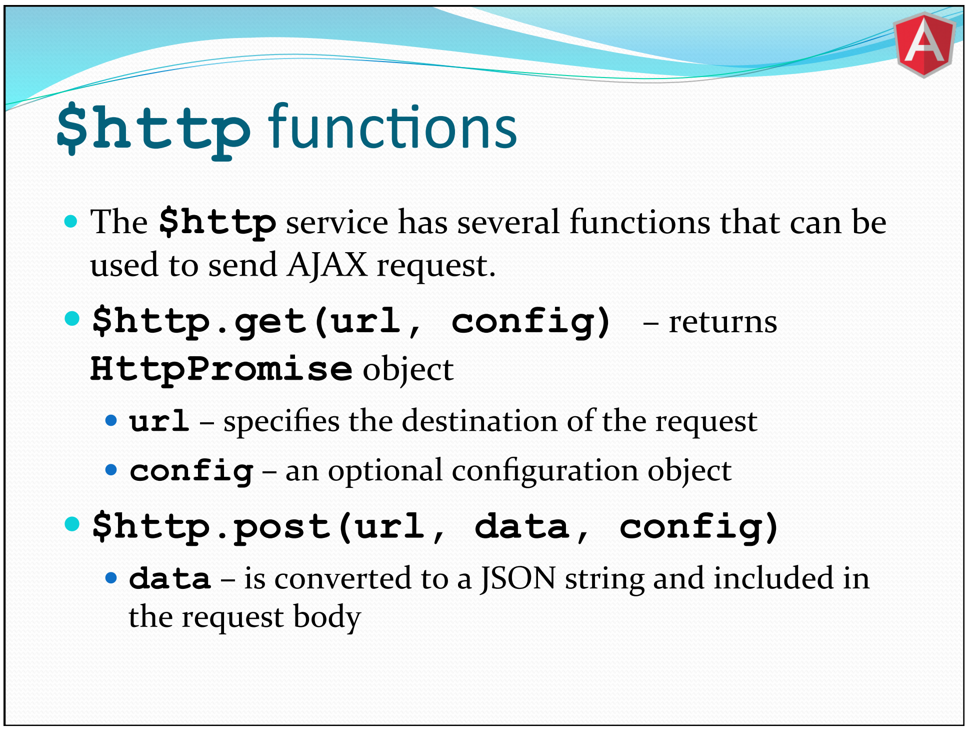
- The **`$http.get()`** function accepts the URL and returns a “promise” object.
- The promise object provides **`success()`** and an **`error()`** function.
- These functions can be called by providing a function as a parameter to each of them.
- If the AJAX call succeeds, the function passed to the **`success()`** is executed.
- If the AJAX call fails the function passed to the **`error()`** is executed.

http://localhost/AngularJS/Demo36.htm
App29.js



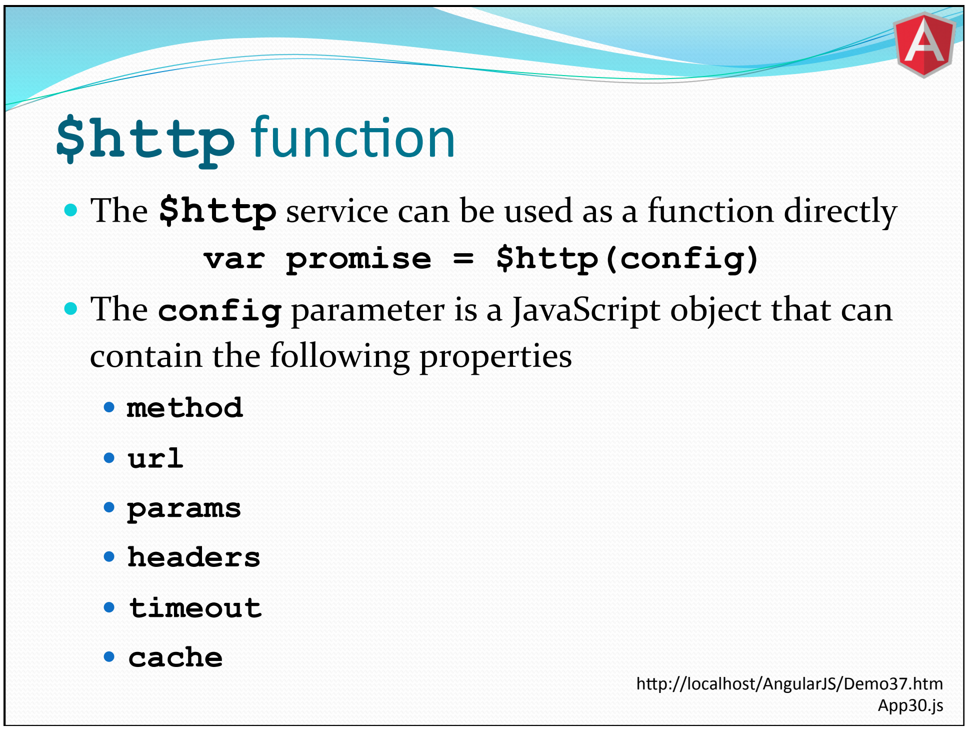
$http

- The callback functions in the **success()** and **error()** functions accept the following parameters
- **data** – the JSON object returned by the server
  - The **$http** service assumes that the server sends JSON
- **status** – the HTTP status code returned by the server along with the response
- **headers** – used to obtain HTTP response headers returned with the response.
- **config** – the configuration object used to create the given HTTP request.



$http functions

- The **$http** service has several functions that can be used to send AJAX request.
- **$http.get(url, config)** – returns **HttpPromise** object
  - **url** – specifies the destination of the request
  - **config** – an optional configuration object
- **$http.post(url, data, config)**
  - **data** – is converted to a JSON string and included in the request body



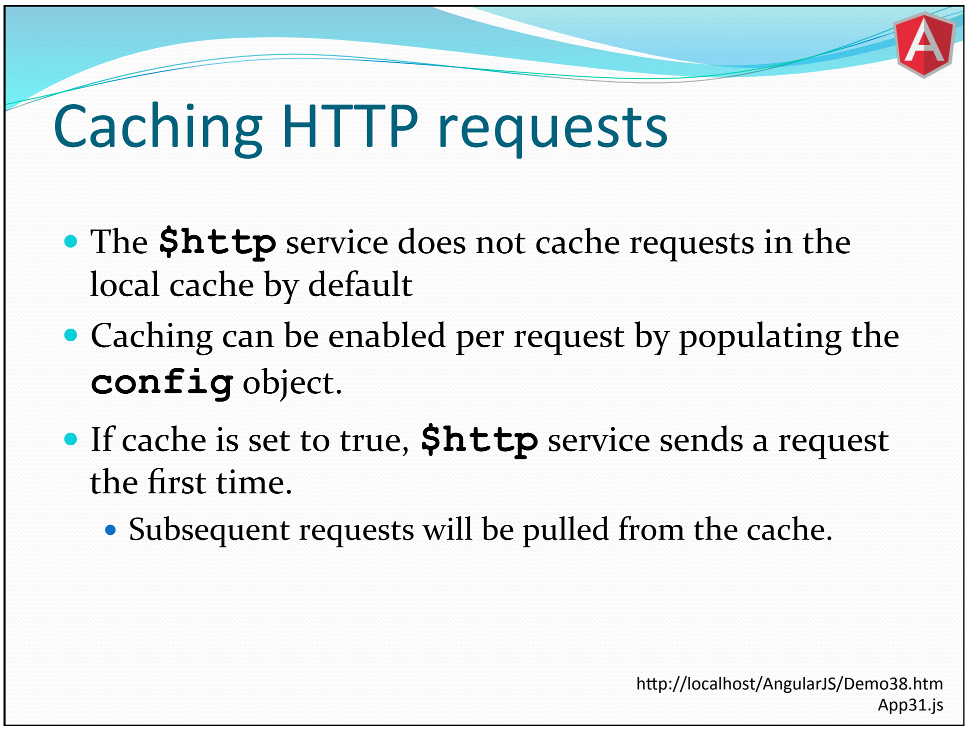
`$http` function

- The **`$http`** service can be used as a function directly

  `var promise = $http(config)`

- The **`config`** parameter is a JavaScript object that can contain the following properties
  - **`method`**
  - **`url`**
  - **`params`**
  - **`headers`**
  - **`timeout`**
  - **`cache`**

http://localhost/AngularJS/Demo37.htm
App30.js



Caching HTTP requests

- The **`$http`** service does not cache requests in the local cache by default
- Caching can be enabled per request by populating the **`config`** object.
- If cache is set to true, **`$http`** service sends a request the first time.
  - Subsequent requests will be pulled from the cache.

http://localhost/AngularJS/Demo38.htm
App31.js

Summary

- Services provide a mechanism to ensure data prevails for the lifetime of an app.
- Services are singleton objects instantiated once per app.
- Angular provides a number of built-in services that can be used in many scenarios.

Mobile apps

- AngularJS provides good support for mobile apps with modules developed by the Angular team and the community.
- Angular leverages HTML and CSS to help create a mobile-ready Angular app.
- Angular ver 1.2.2 onwards offers support for touch related events using the **`ngTouch`** module.

ngTouch

- **`ngTouch`** needs to be downloaded and installed.
- Mobile browsers work a little differently when compared to desktop browsers when dealing with click events.
- Mobile browsers detect a "tap" event and wait for about 300ms to detect further taps.
  - The wait is to check if there is a double-tap
- After the delay, the browser fires a click event.
  - Instead of dealing with the click event, the touch event can be detected.

ngTouch

- The **`ngTouch`** library handles the functionality seamlessly through the **`ng-click`** directive.
- This directive calls the correct click event.
  - The fast click event is called.
- **`ngTouch`** handles the removal of browser delay on **`ng-click`** events.
- The **`ngClick`** directive on mobile browsers works in the same manner as in desktop browsers.

swipe directives

- **ngTouch** introduces two new directives called swipe directives.
- These directives are used to capture user swipes, either left or right across the screen.
- These directives help in introducing contextual features in applications.
- **ngSwipeLeft** directive is used to detect a swipe from right to left on an element.
- **ngSwipeRight** is used to detect a swipe from left to right on an element.

http://localhost/angularjs/Demo55.htm
App46.js

Summary

- **ngTouch** module provides touch events and helpers for touch enabled devices.
- **ngTouch** is a powerful replacement for the default **ngClick** designed for touchscreen device usage.
- **ngSwipeLeft** and **ngSwipeRight** can be used for implementation of custom behaviors.

Architecture - Practices

General Guidelines

1. Modules

- Divide modules by type
  - Directives, Services, Filters etc.

2. Controllers

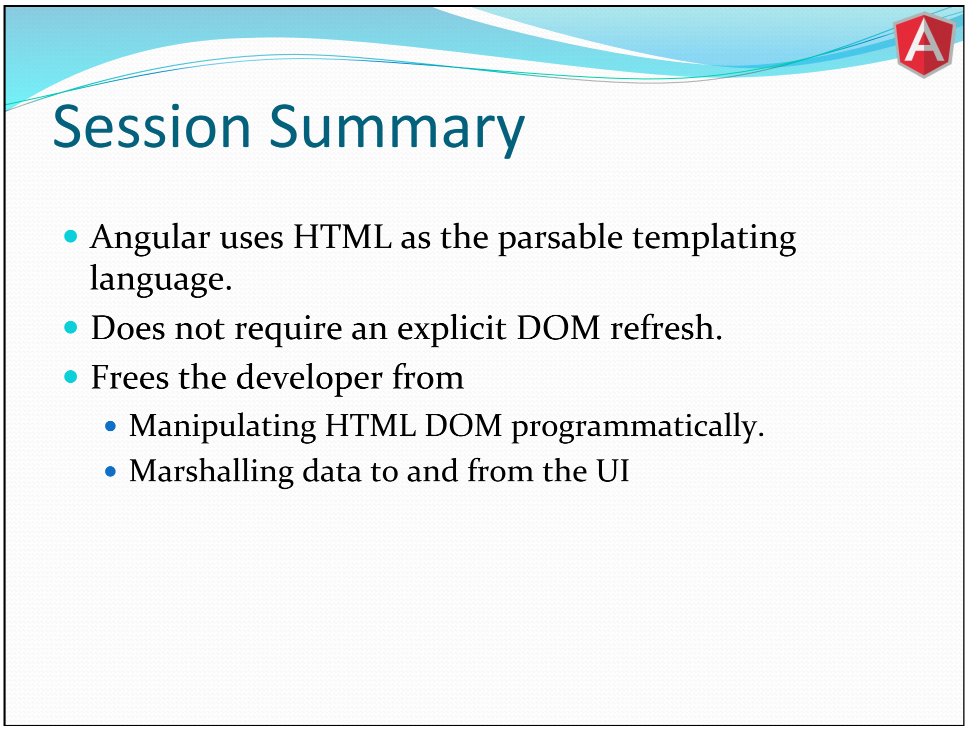
- Size of controllers can be reduced by shifting DOM handling away from them.
- The optimal approach to share data between controllers should be chosen.

3. Directives

- Directives can help reduce clutter in the controller.
- Directives need not always have a view template.

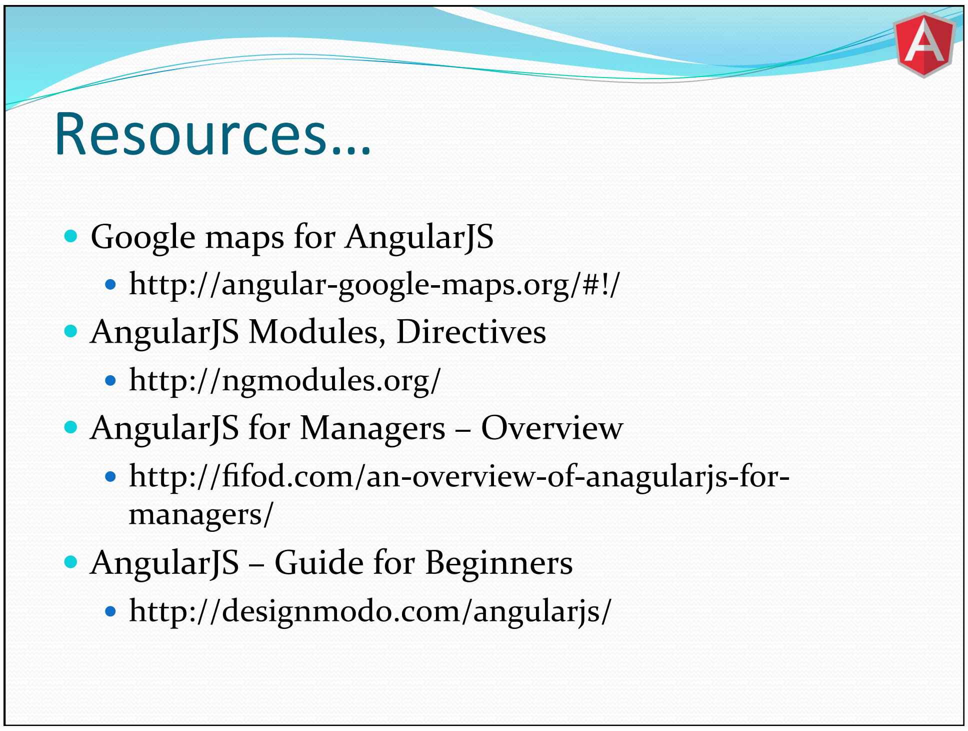
Session Summary

- AngularJS is a open-source web application framework maintained by Google and community.
- The design goal is to augment web applications with MVC/MVVW capability.
  - This simplifies development of applications.
- AngularJS assists in creating single-page applications employing HTML, CSS and JavaScript on the client-side.



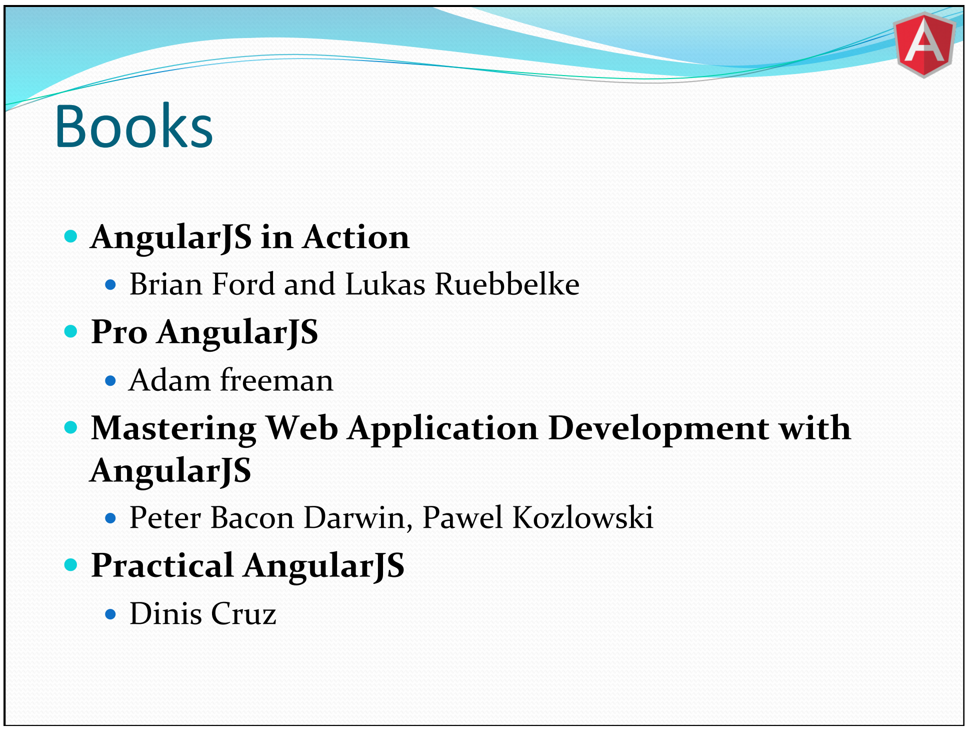
Session Summary

- Angular uses HTML as the parsable templating language.
- Does not require an explicit DOM refresh.
- Frees the developer from
  - Manipulating HTML DOM programmatically.
  - Marshalling data to and from the UI



Resources…

- Google maps for AngularJS
  - http://angular-google-maps.org/#!/
- AngularJS Modules, Directives
  - http://ngmodules.org/
- AngularJS for Managers – Overview
  - http://fifod.com/an-overview-of-anagularjs-for-managers/
- AngularJS – Guide for Beginners
  - http://designmodo.com/angularjs/



Books

- **AngularJS in Action**
  - Brian Ford and Lukas Ruebbelke
- **Pro AngularJS**
  - Adam freeman
- **Mastering Web Application Development with AngularJS**
  - Peter Bacon Darwin, Pawel Kozlowski
- **Practical AngularJS**
  - Dinis Cruz