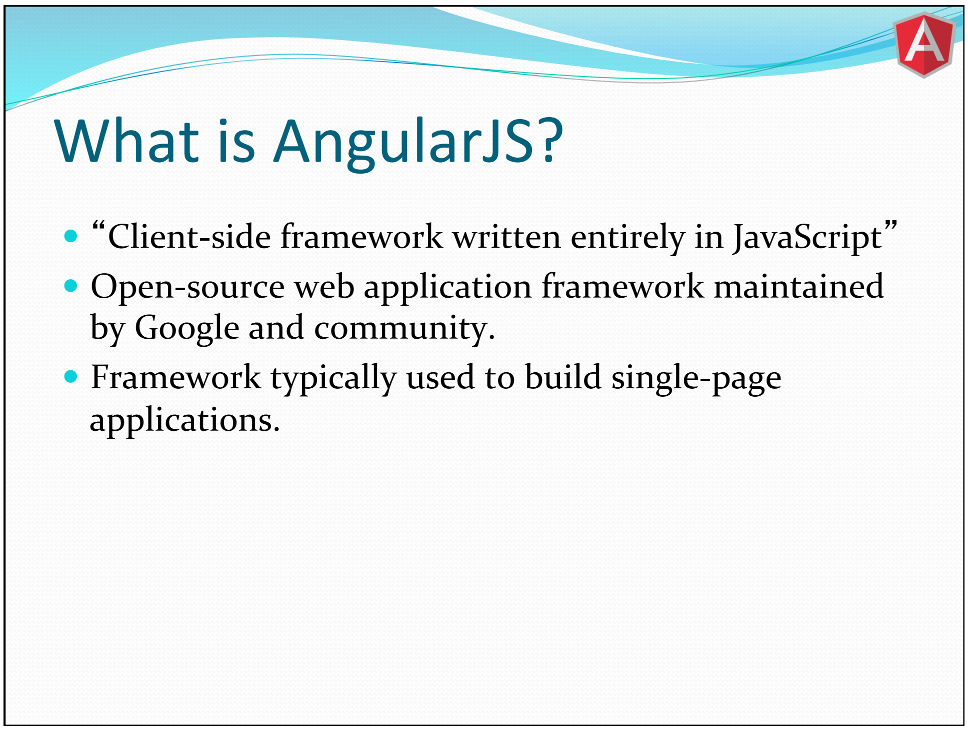
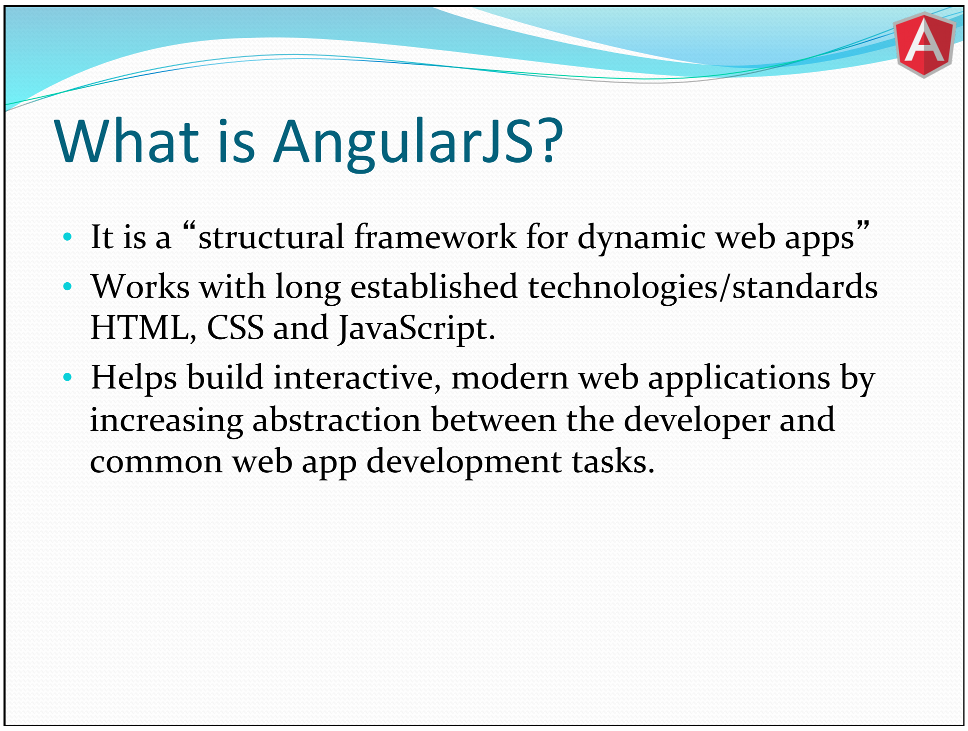
AngularJS

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



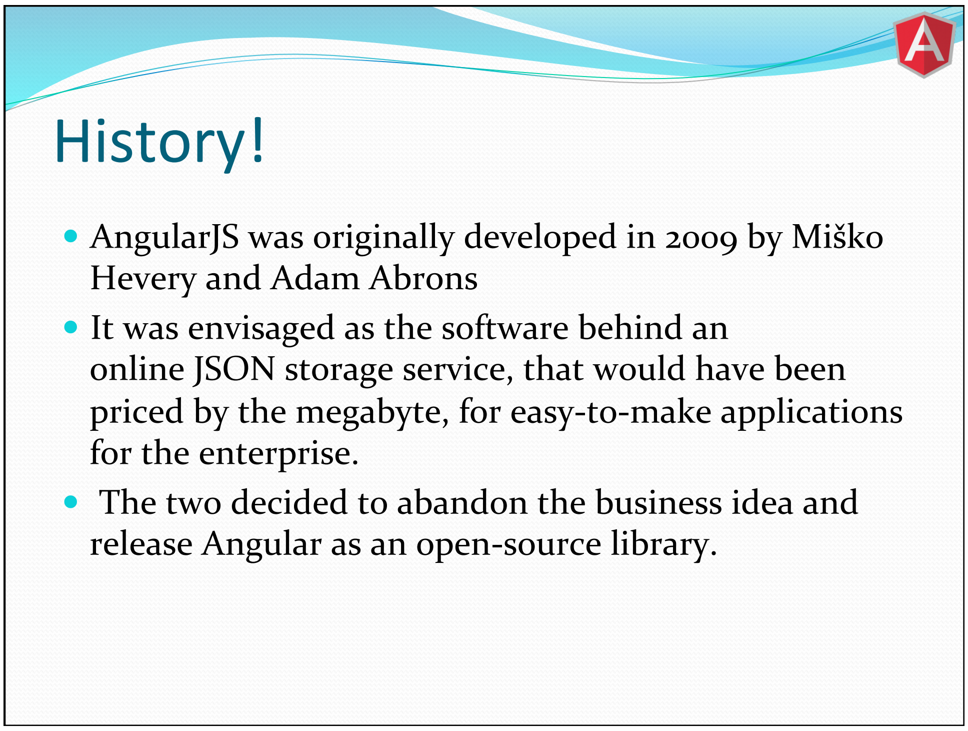
What is AngularJS?

- "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.



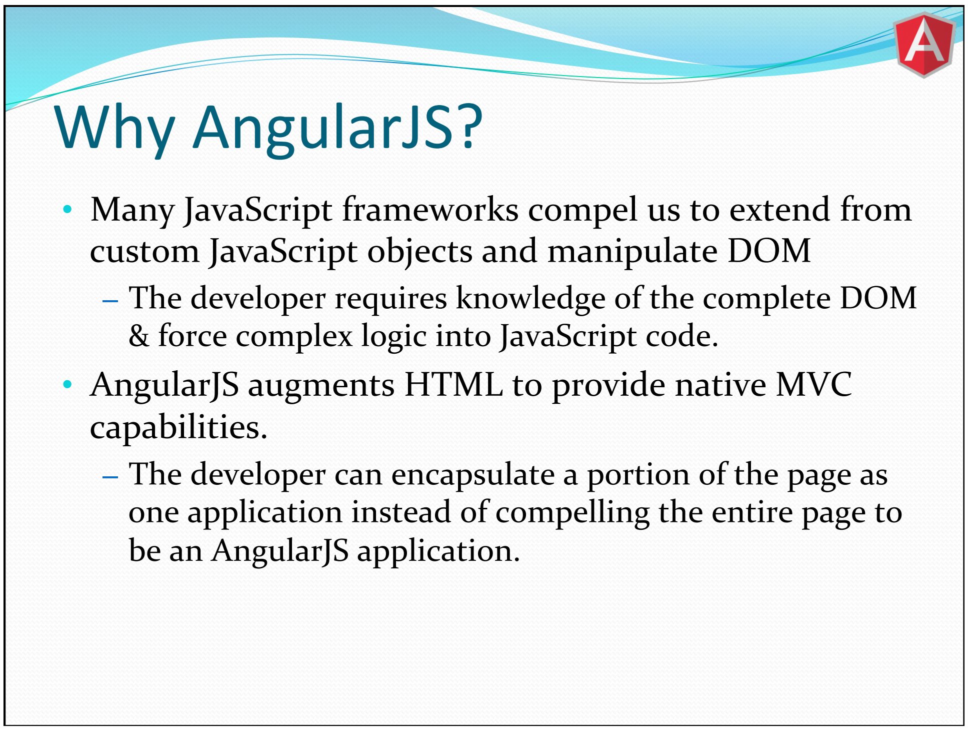
What is AngularJS?

- 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.



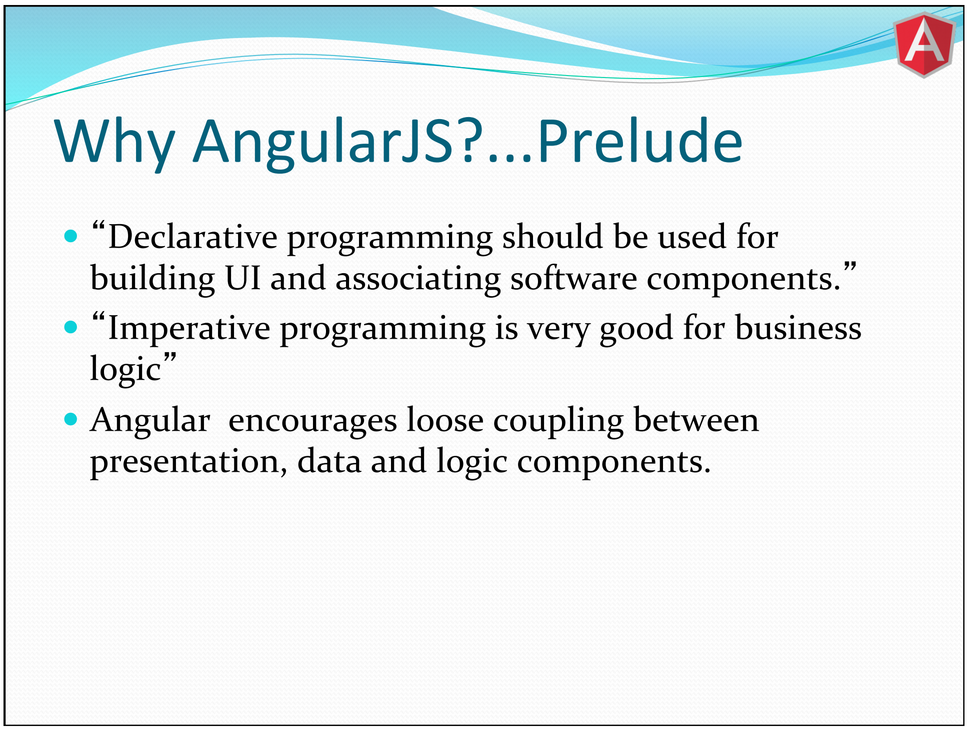
History!

- 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

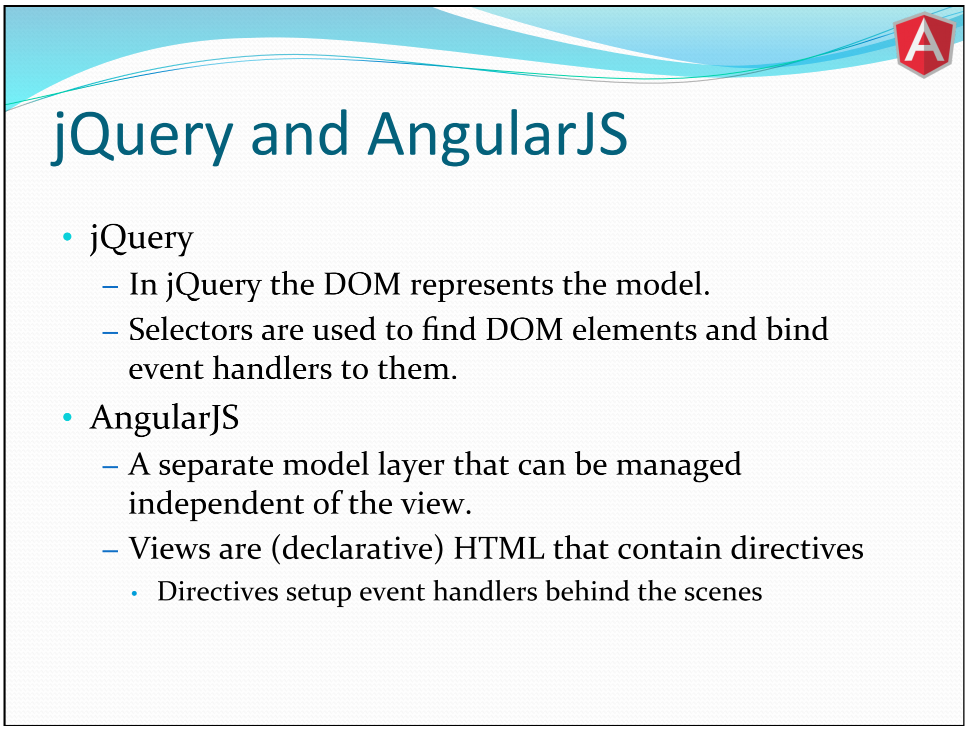
jQuery and AngularJS

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

```
$('.main-menu').dropdownMenu();
```

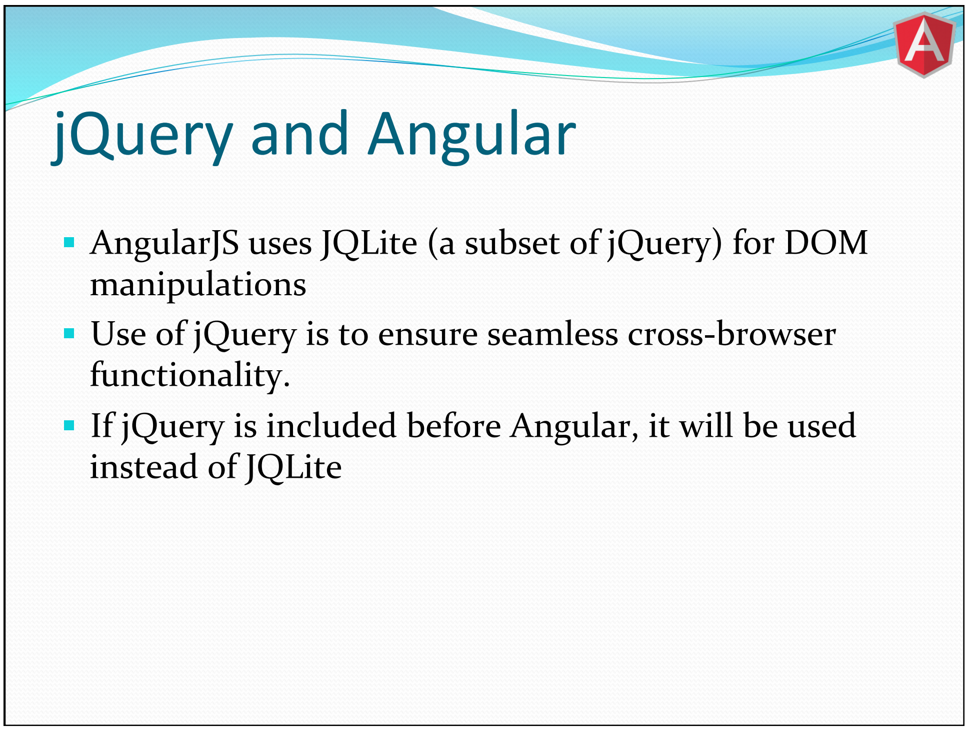
- AngularJS
  - Start with the architecture and finally design the view
  - View is the official record of the view based functionality

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



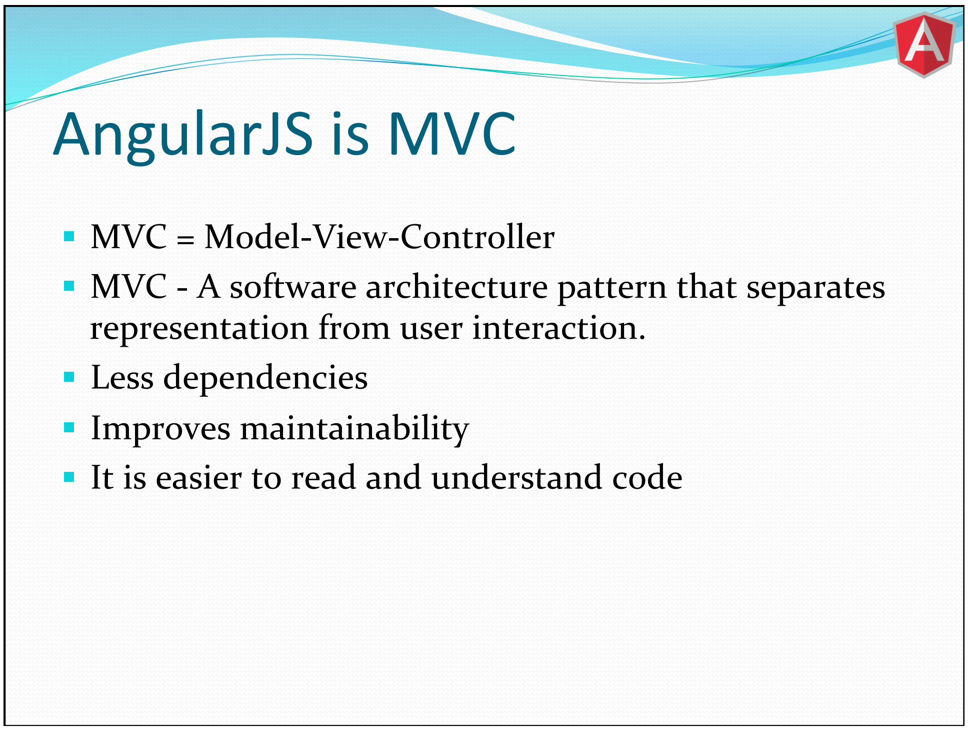
jQuery and AngularJS

- 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



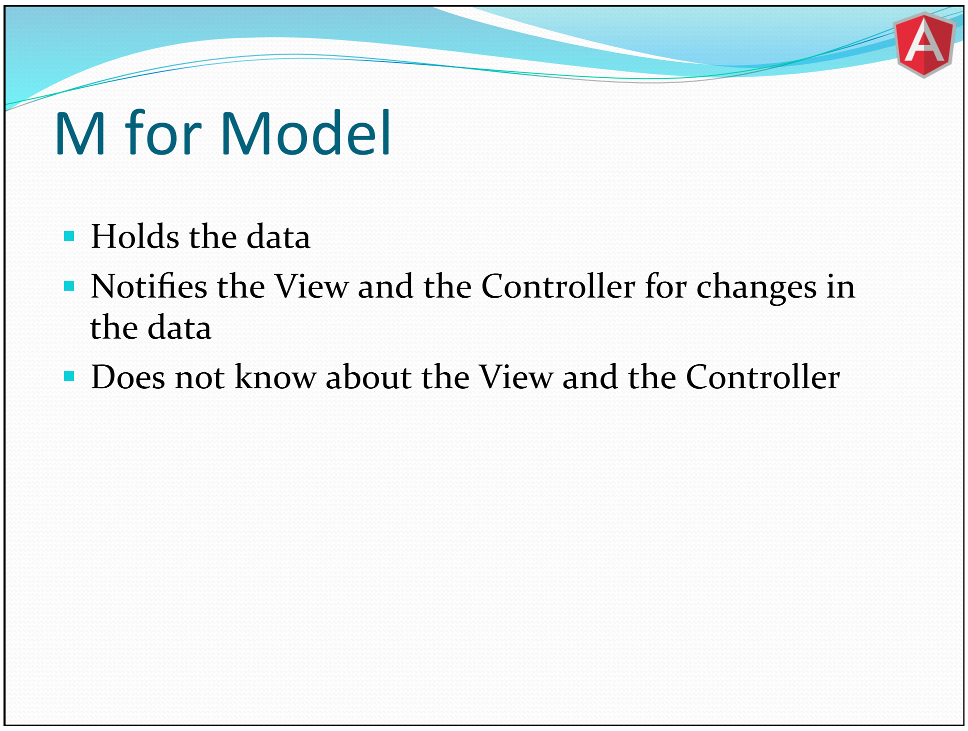
jQuery and Angular

- 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



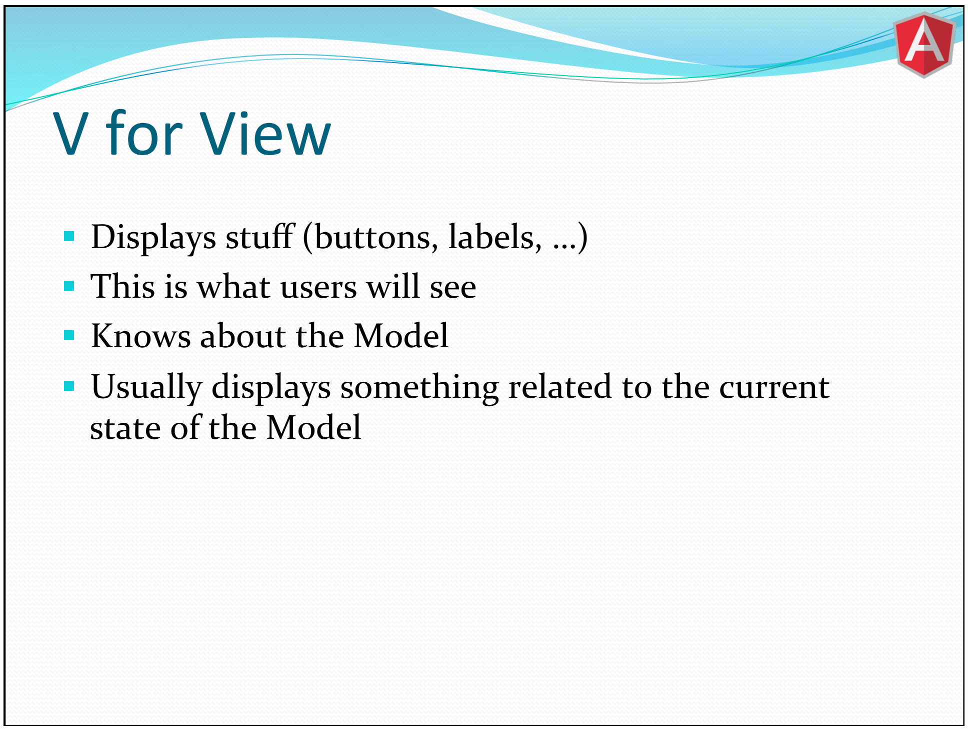
AngularJS is MVC

- 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

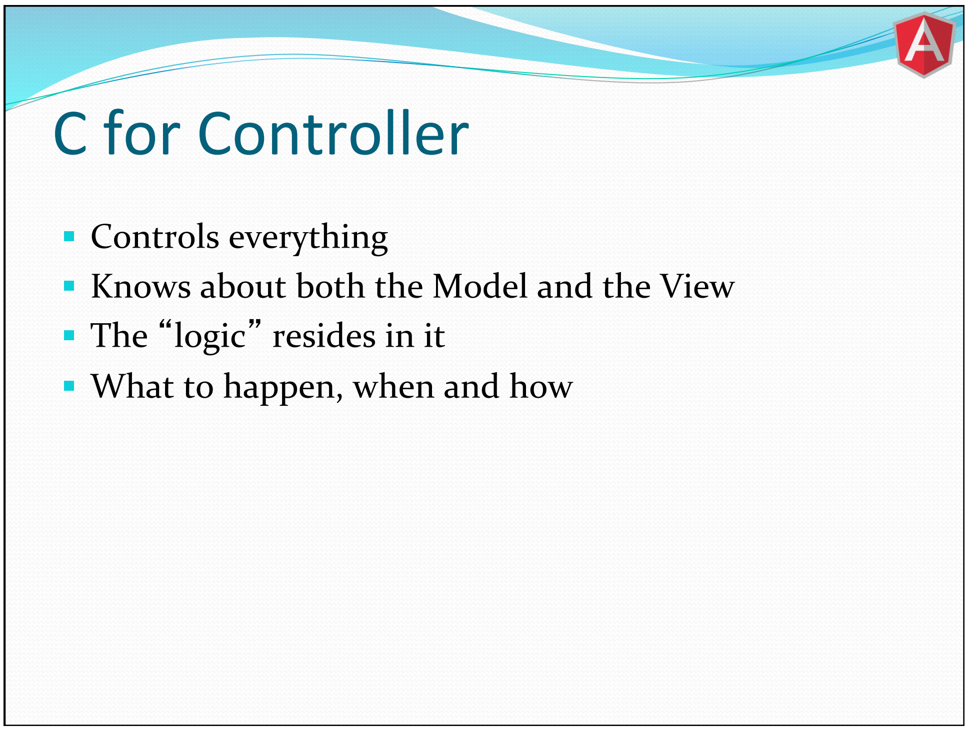


M for Model

- 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

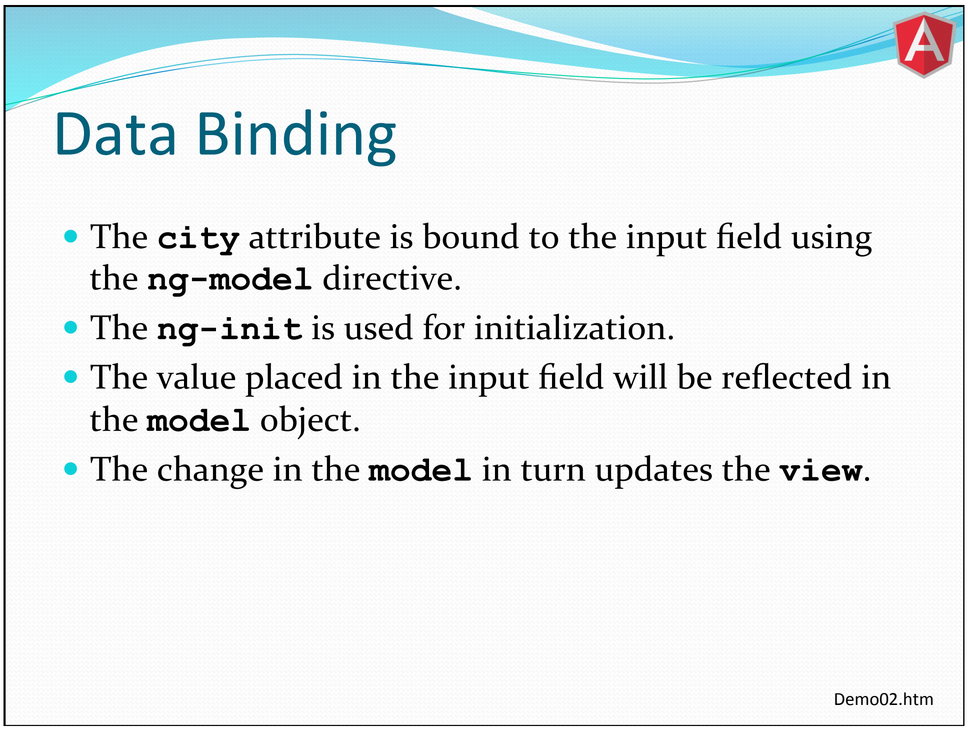
An AngularJS application

```
<!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…

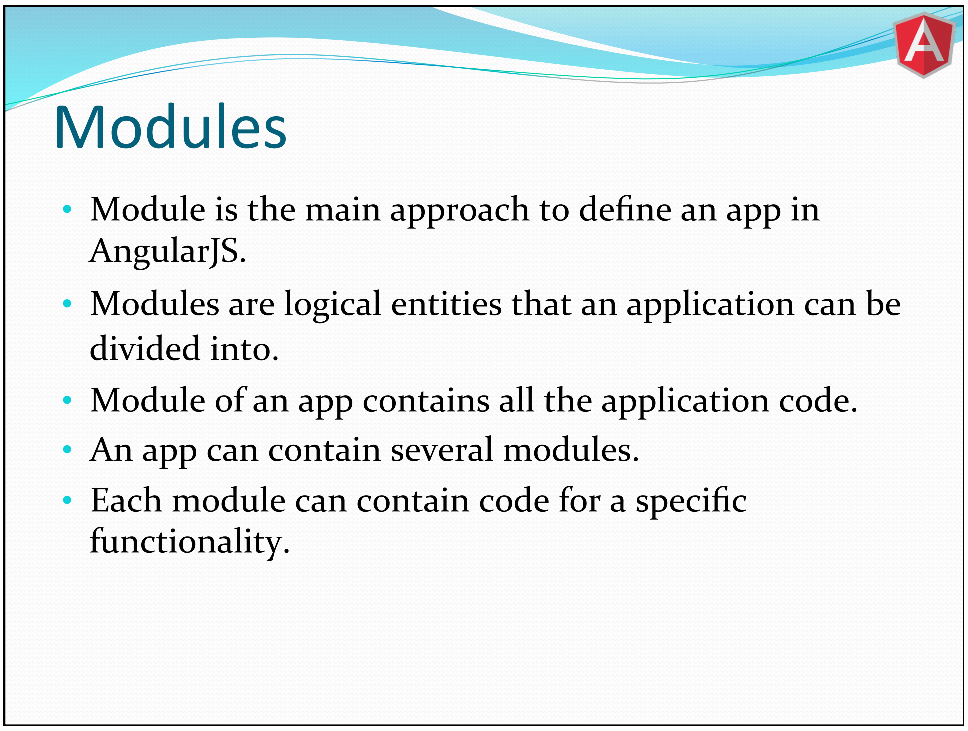
- The **ng-app** attribute is set on an element of DOM.
- The **ng-app** attribute declares that everything inside of it belongs to this Angular app.
  - This approach helps nest an Angular app in a web app.
- The only components that are impacted by AngularJS are the DOM elements that are declared inside 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

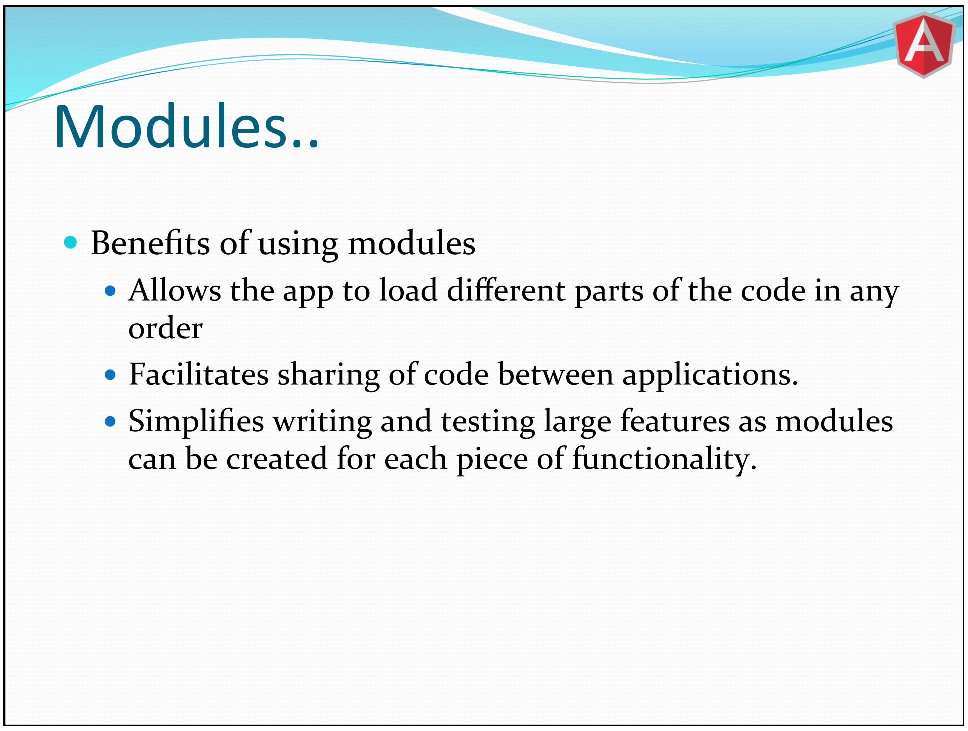
- The **`city`** attribute is bound to the input field using the **`ng-model`** directive.
- The **`ng-init`** is used for initialization.
- The value placed in the input field will be reflected in the **`model`** object.
- The change in the **`model`** in turn updates the **`view`**.

Demo02.htm



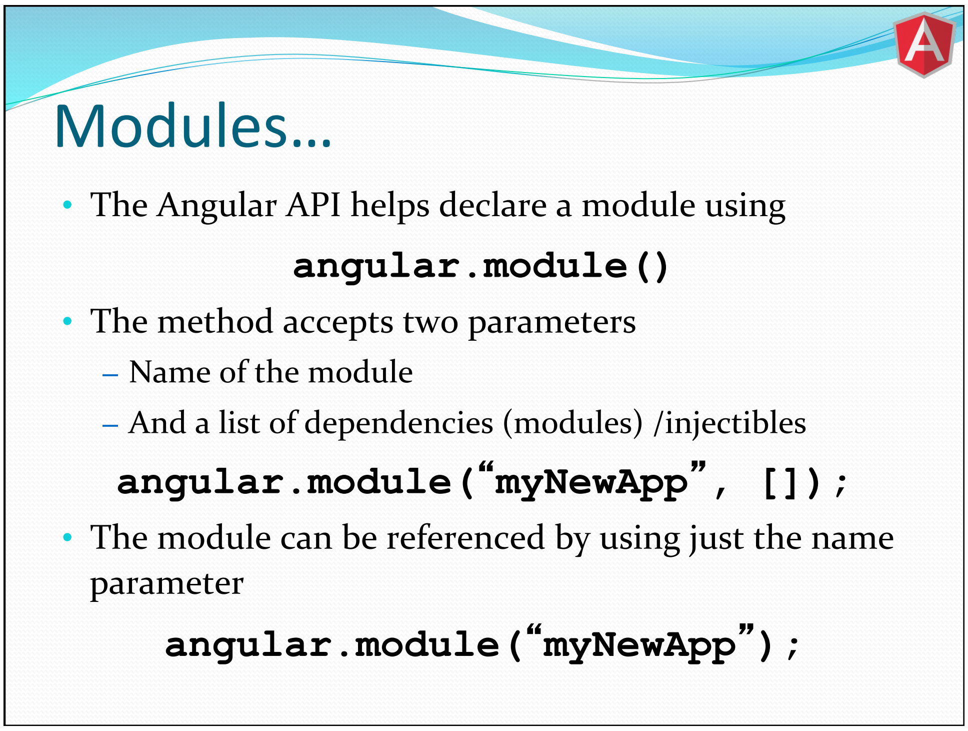
Modules

- 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");
```

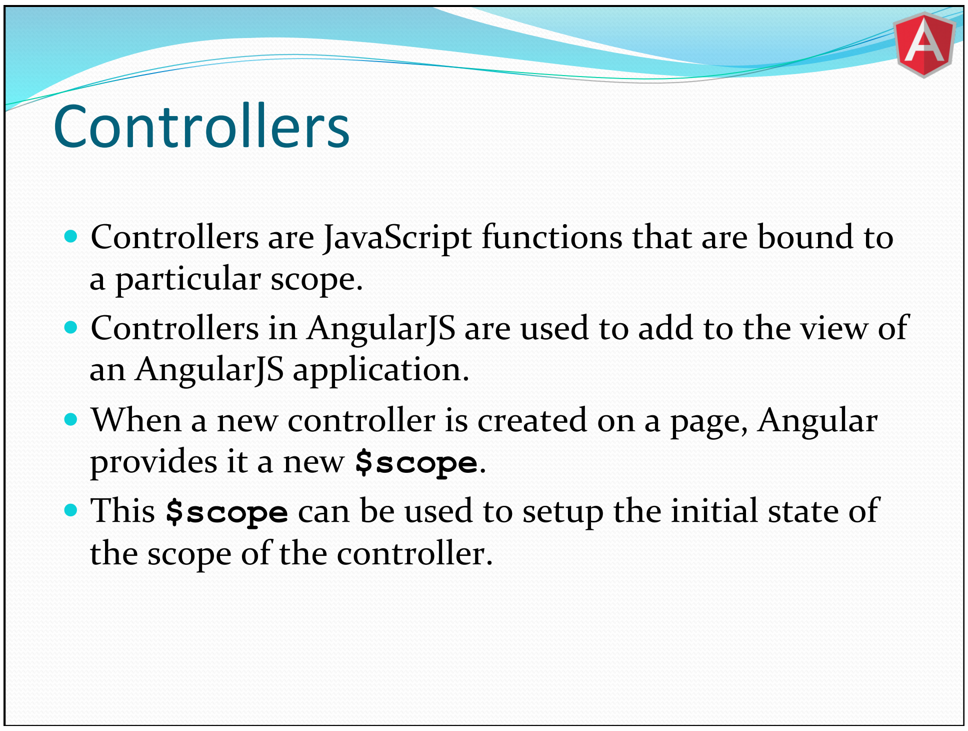
Scopes

- 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…

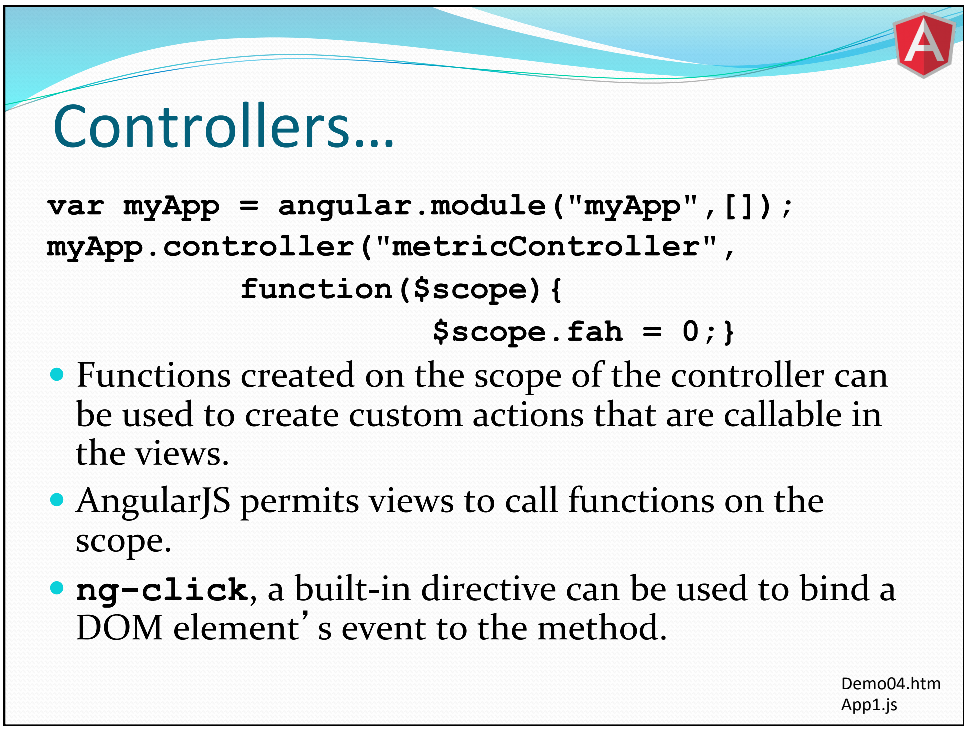
- Angular when run and generates a view, creates a binding from the root **`ng-app`** element to the **`$rootScope`**.
- The **`$rootScope`** is the eventual parent of all **`$scope`** objects.
- The **`$rootScope`** is analogous to having a global in Angular.
  - It is not recommended to attach a lot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

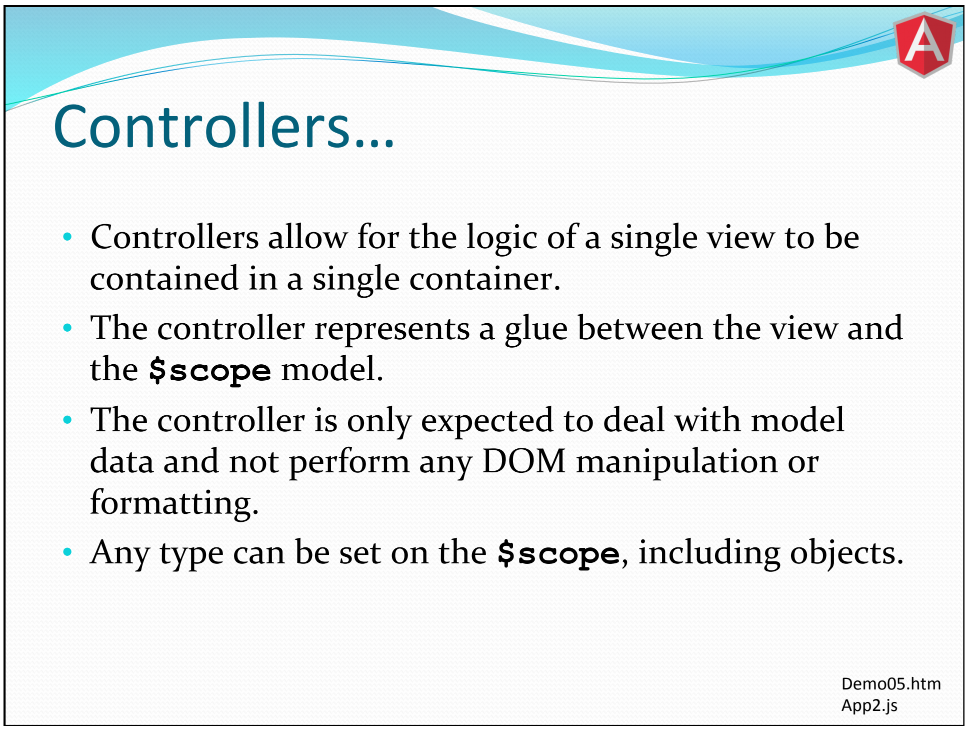


Controllers…

```
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.

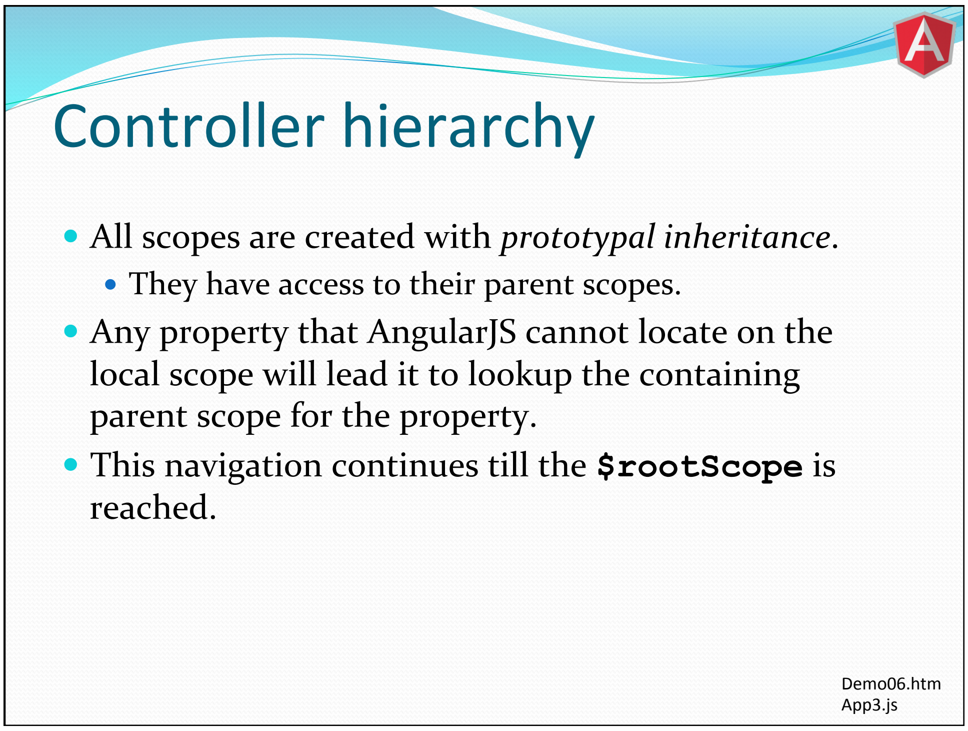
Demo04.htm
App1.js



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)

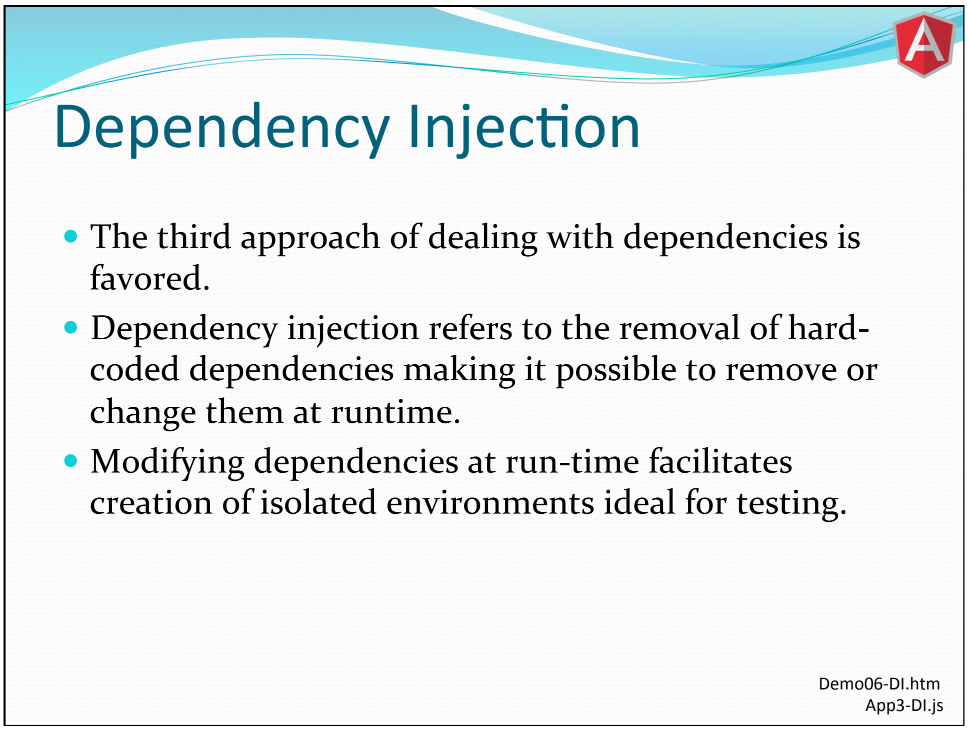
Dependency Injection

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

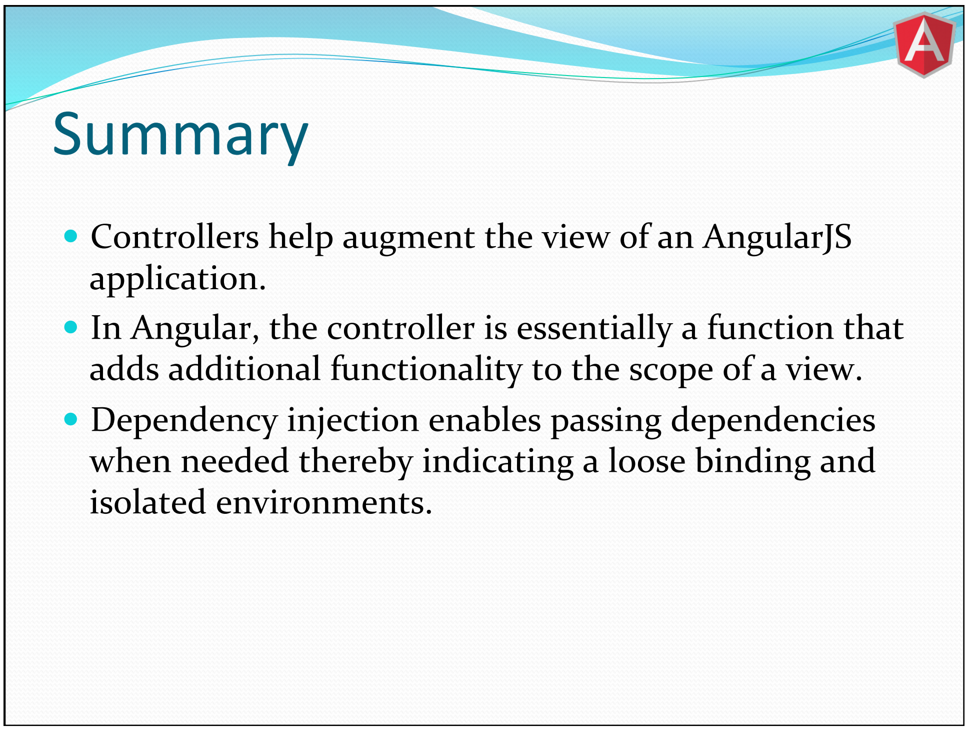
- There are three approaches by which an object can get hold of its dependencies
- *Create the dependency internally*
  - Using "new" operator
- *Lookup the dependency*
  - By referring to a global variable
- *Pass in the dependency where needed*



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

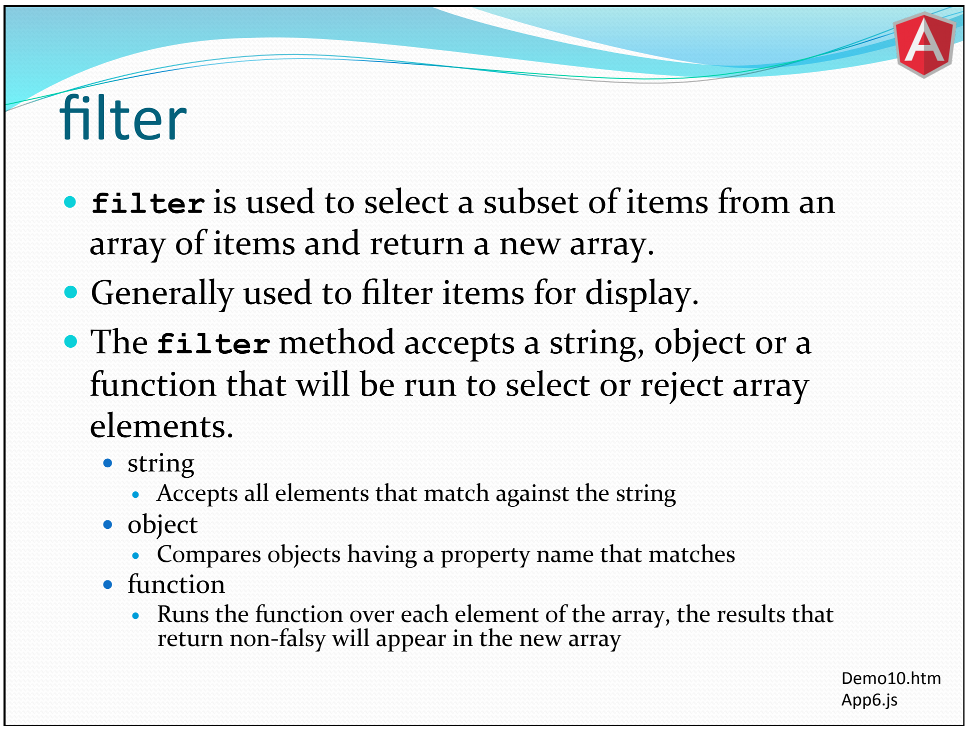
- The date filter allows to format the date in a required format style.
- It provides several built-in options.

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

- The date formatter enables customization of the date format.

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

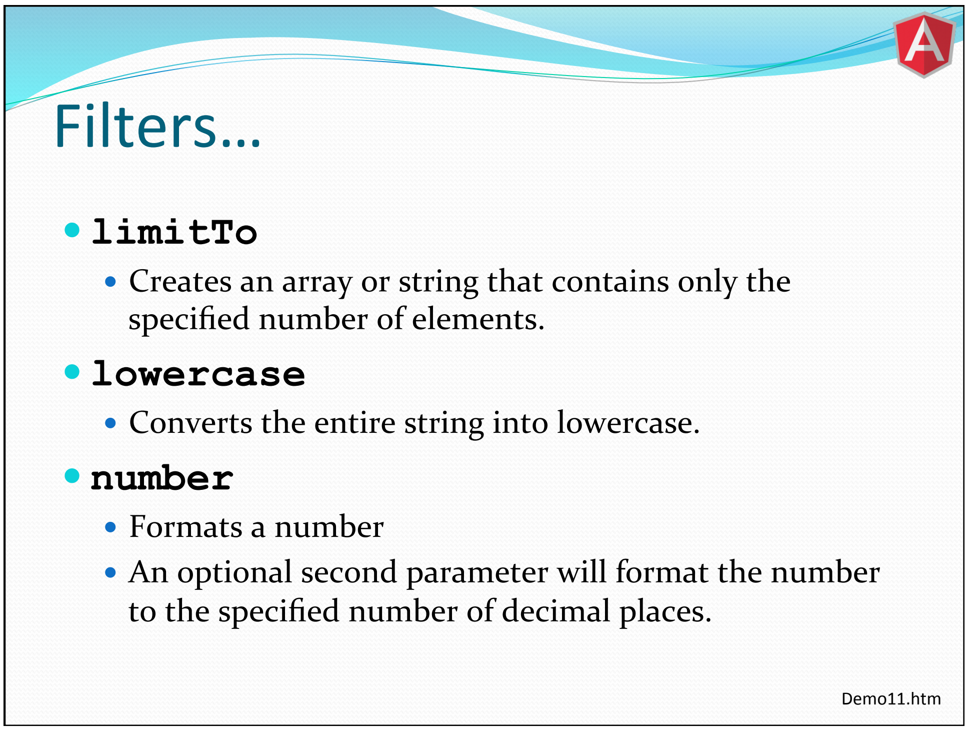
Demo09.htm
App5.js



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

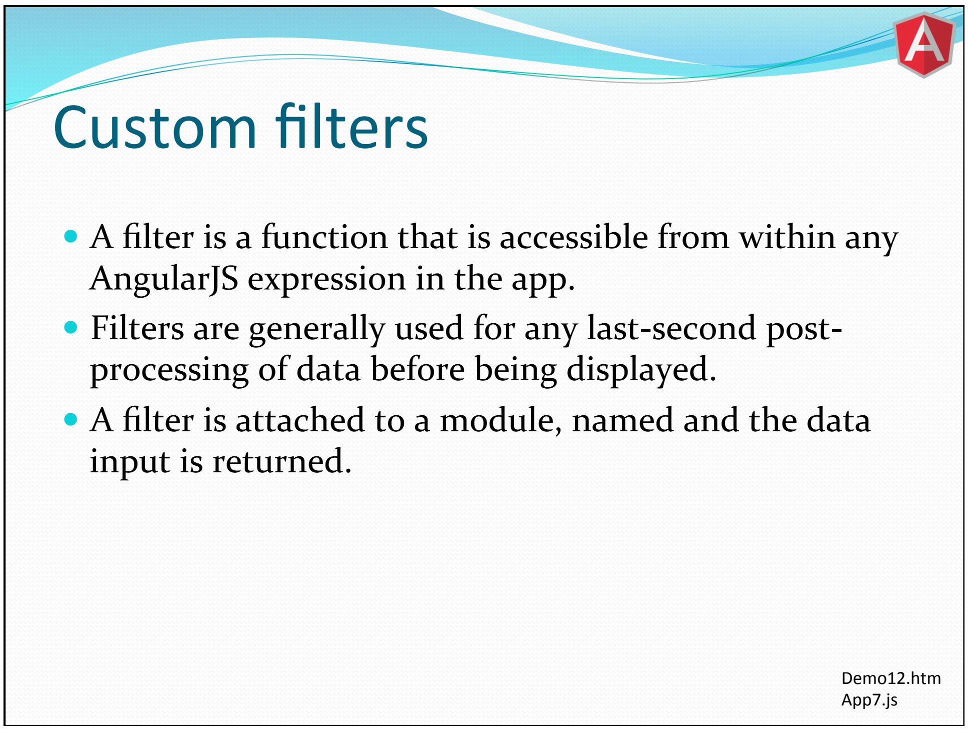
Demo10.htm
App6.js



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.

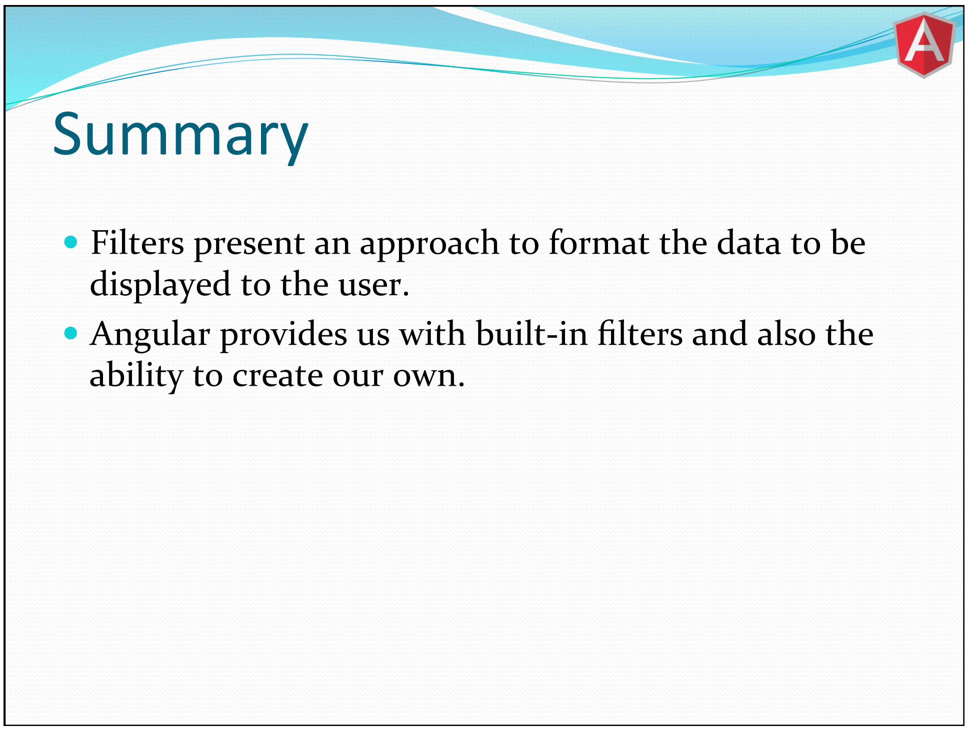
Demo11.htm



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



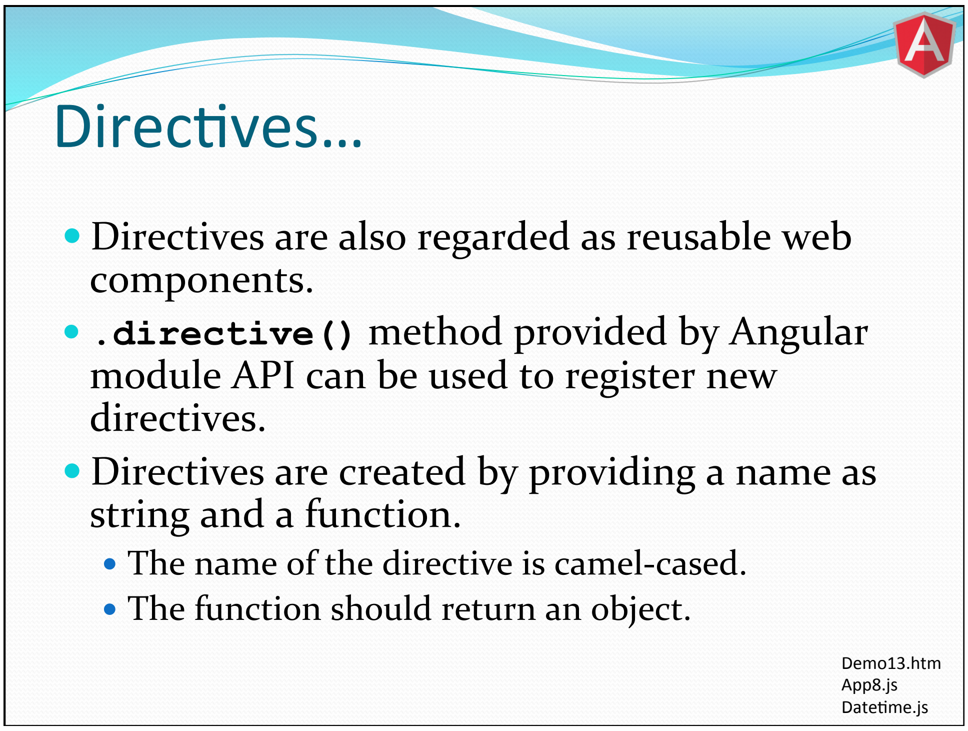
Summary

- 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

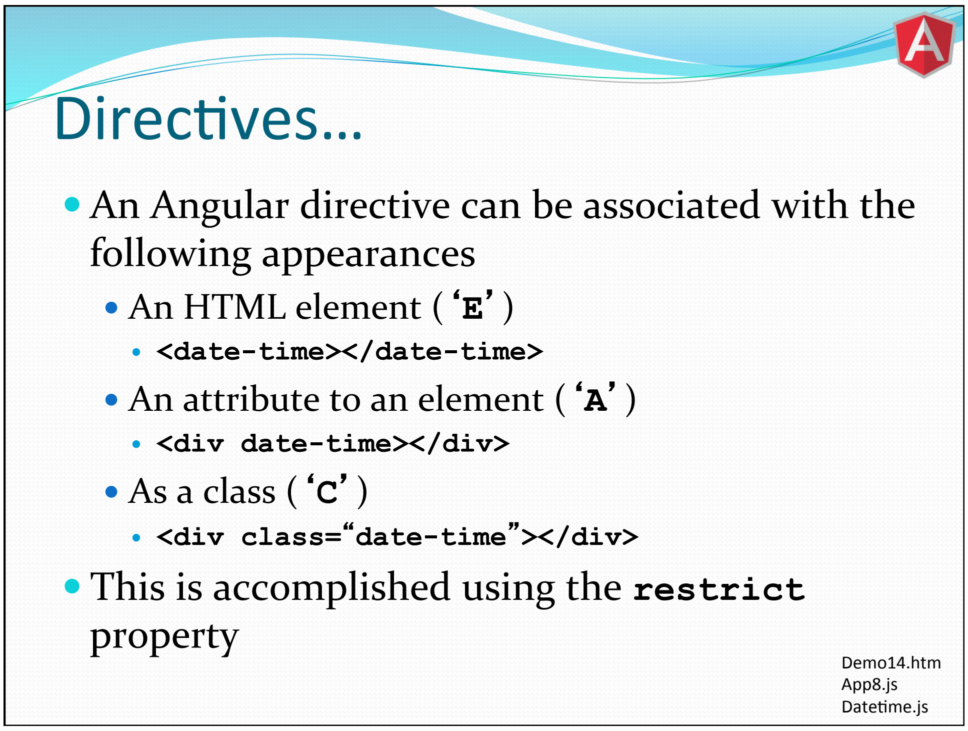
- Directives are markers on a DOM element (attribute, element or CSS class) that tell AngularJS to attach a specified behavior to that element.
- Directive in AngularJS is a function that can be run on a particular DOM element.
  - The function provides additional functionality to the element.
- Directives 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.

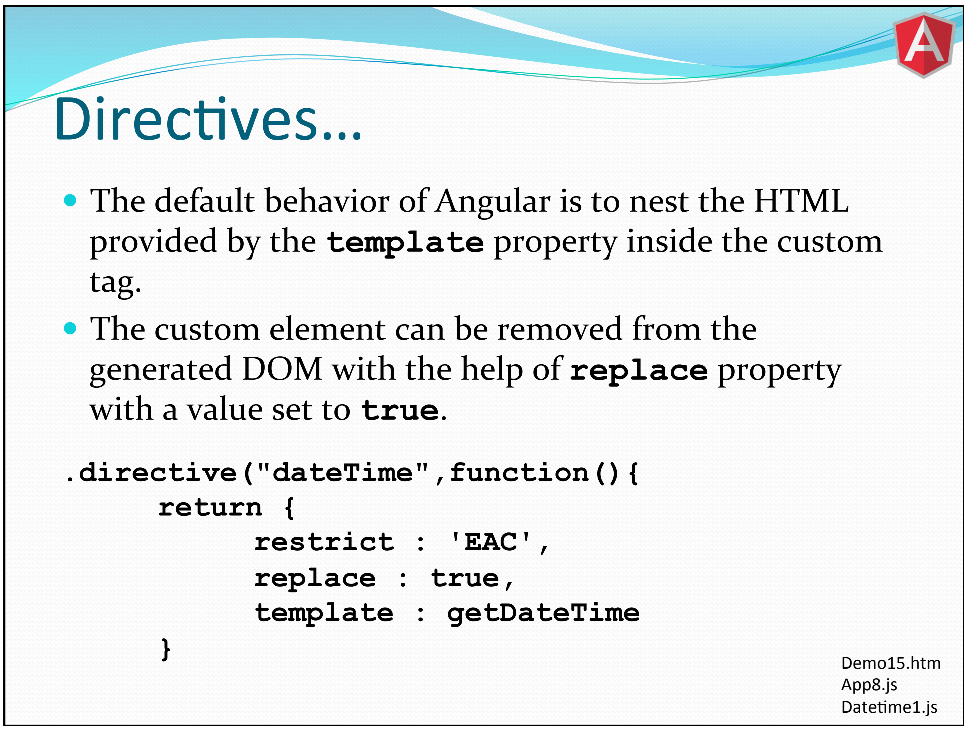
Demo13.htm
App8.js
Datetime.js



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

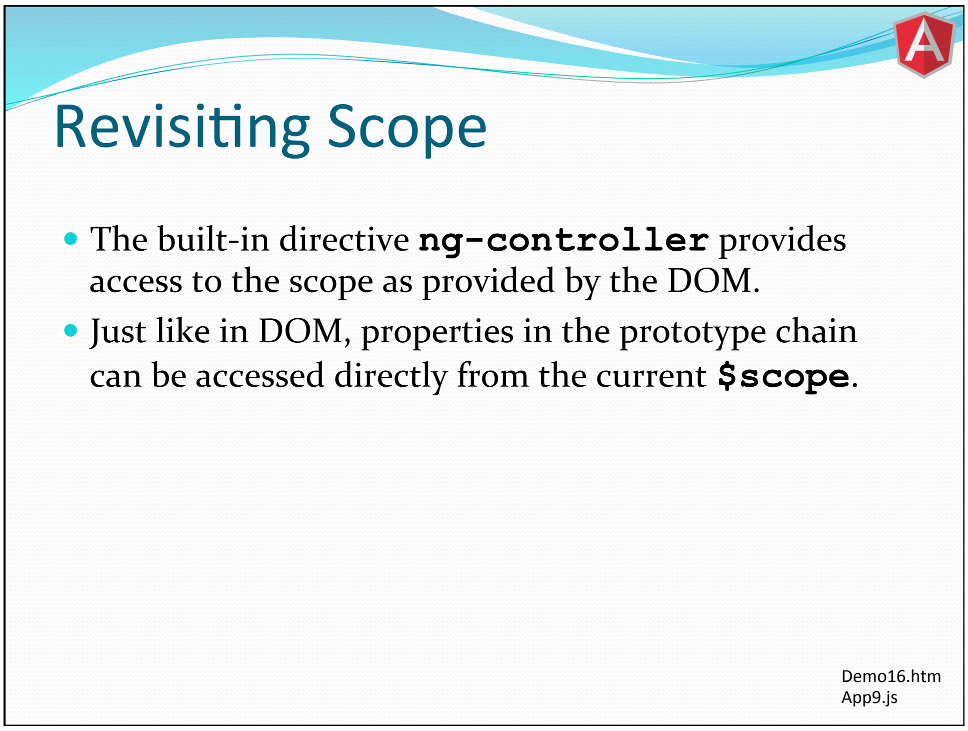


Directives…

- 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
      }
```

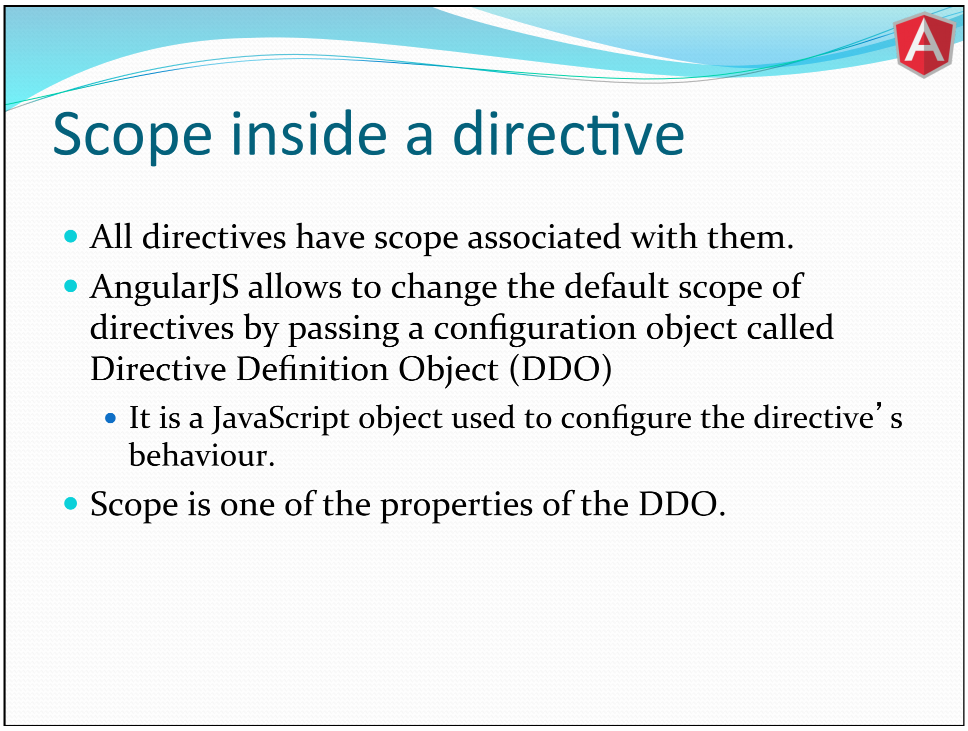
Demo15.htm
App8.js
Datetime1.js



Revisiting Scope

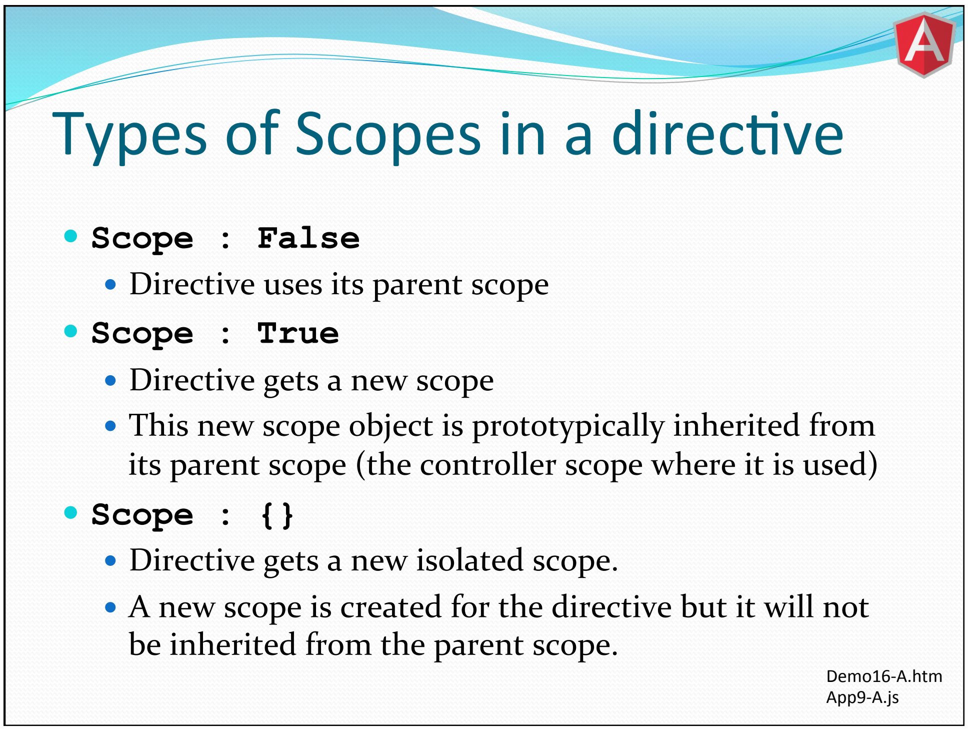
- The built-in directive **`ng-controller`** provides access to the scope as provided by the DOM.
- Just like in DOM, properties in the prototype chain can be accessed directly from the current **`$scope`**.

Demo16.htm
App9.js



Scope inside a directive

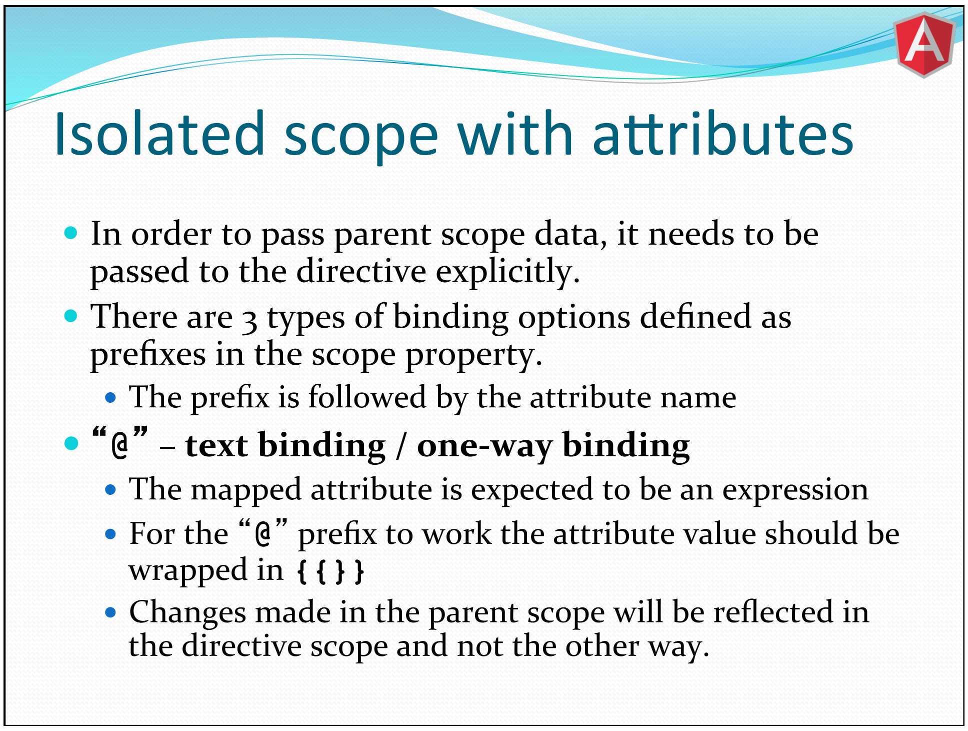
- All directives have scope associated with them.
- AngularJS allows to change the default scope of directives by passing a configuration object called Directive Definition Object (DDO)
  - It is a JavaScript object used to configure the directive’ s behaviour.
- Scope is one of the properties of the DDO.



Types of Scopes in a directive

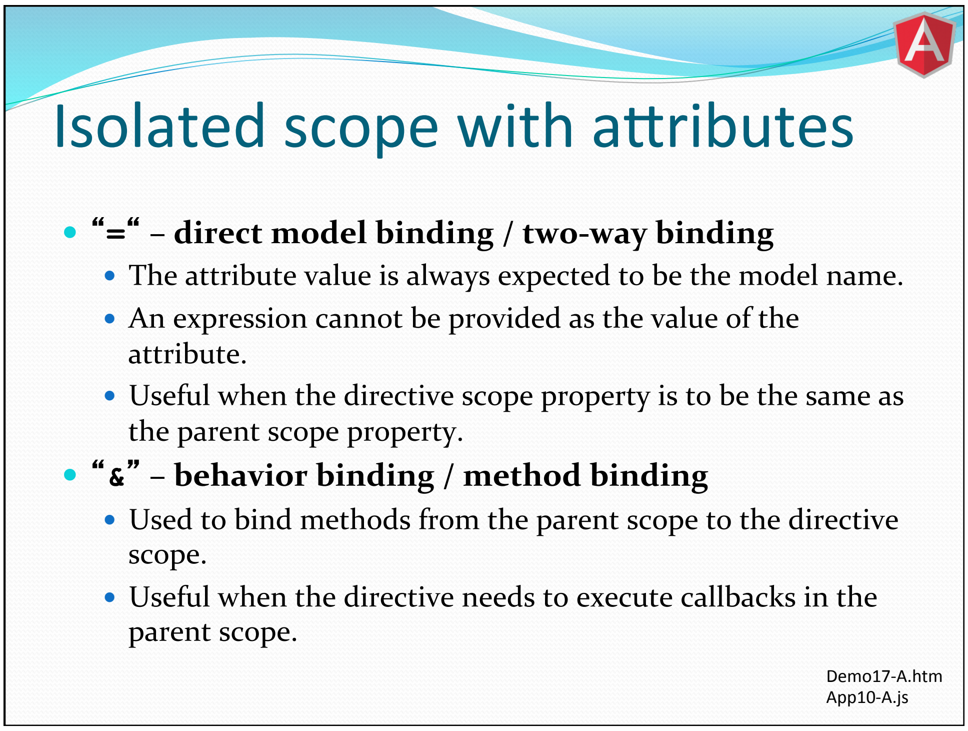
- **`Scope : False`**
  - Directive uses its parent scope
- **`Scope : True`**
  - Directive gets a new scope
  - This new scope object is prototypically inherited from its parent scope (the controller scope where it is used)
- **`Scope : {}`**
  - Directive gets a new isolated scope.
  - A new scope is created for the directive but it will not be inherited from the parent scope.

Demo16-A.htm
App9-A.js



Isolated scope with attributes

- In order to pass parent scope data, it needs to be passed to the directive explicitly.
- There are 3 types of binding options defined as prefixes in the scope property.
  - The prefix is followed by the attribute name
- **“`@`” – text binding / one-way binding**
  - The mapped attribute is expected to be an expression
  - For the “`@`” prefix to work the attribute value should be wrapped in **`{{}}`**
  - Changes made in the parent scope will be reflected in the directive scope and not the other way.



Isolated scope with attributes

- **“=“ – direct model binding / two-way binding**
  - The attribute value is always expected to be the model name.
  - An expression cannot be provided as the value of the attribute.
  - Useful when the directive scope property is to be the same as the parent scope property.
- **“&” – behavior binding / method binding**
  - Used to bind methods from the parent scope to the directive scope.
  - Useful when the directive needs to execute callbacks in the parent scope.

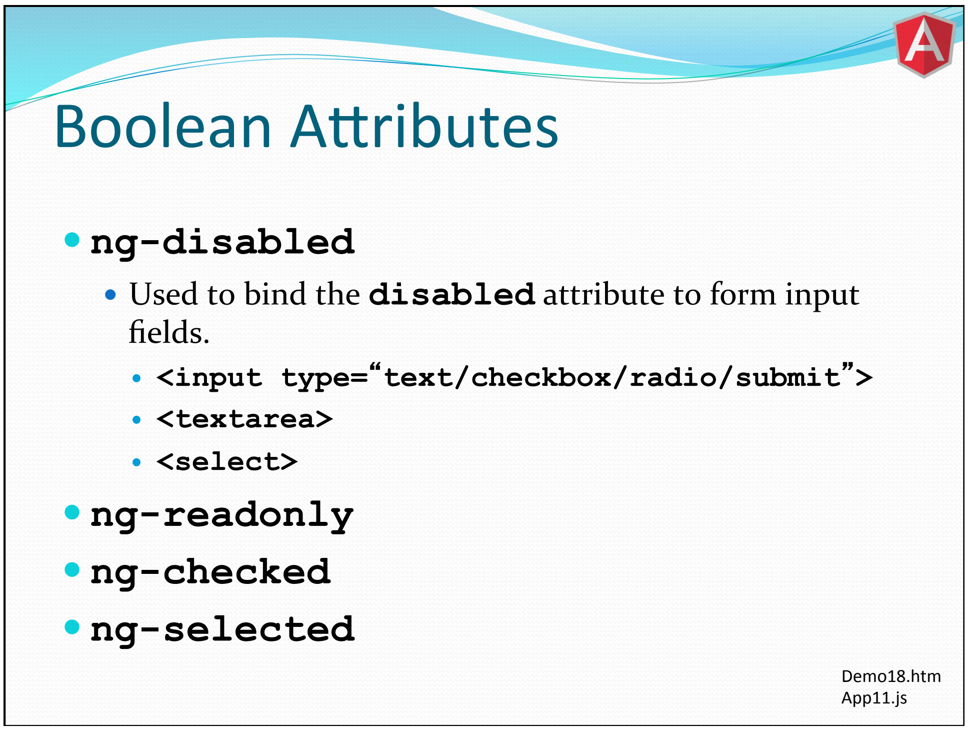
Demo17-A.htm
App10-A.js

Built-in Directives

- AngularJS provides a number of built-in directives.
- All directives are prefixed with the **`ng`** namespace as they are a part of the built-in library.
  - Custom directives should never be prefixed with **`ng`** namespace

Basic **`ng`** Attribute directives

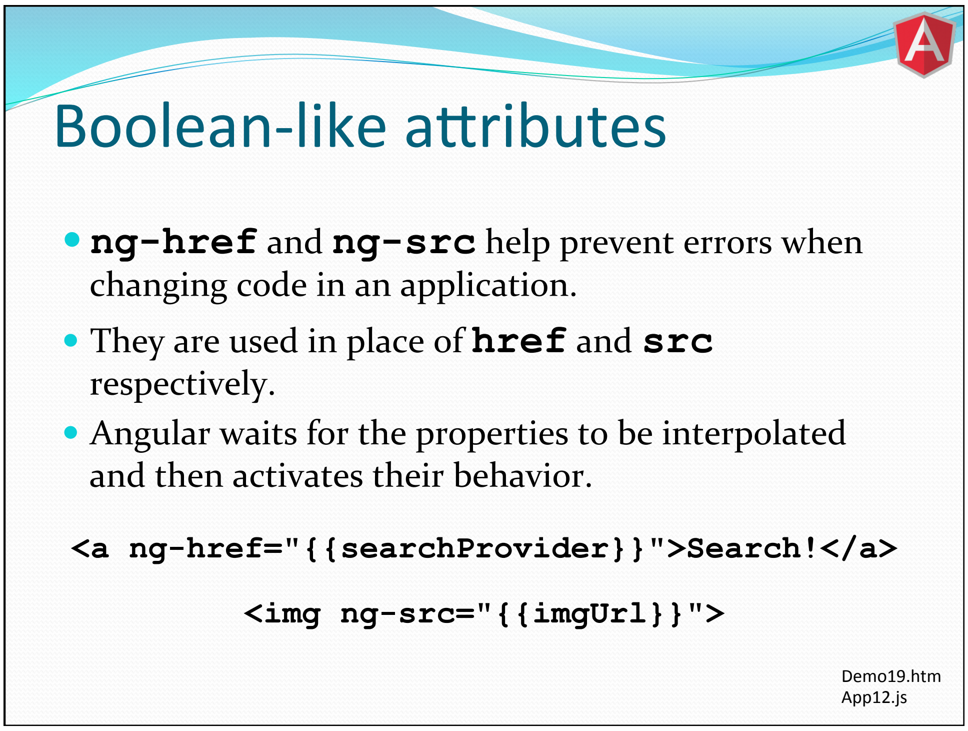
- **`ng-disabled`**
- **`ng-readonly`**
- **`ng-checked`**
- **`ng-selected`**
- **`ng-class`**
- **`ng-style`**
- **`ng-src`**
- **`ng-href`**



Boolean Attributes

- **`ng-disabled`**
  - Used to bind the **`disabled`** attribute to form input fields.
    - `<input type="“text/checkbox/radio/submit”>`
    - `<textarea>`
    - `<select>`
- **`ng-readonly`**
- **`ng-checked`**
- **`ng-selected`**

Demo18.htm
App11.js

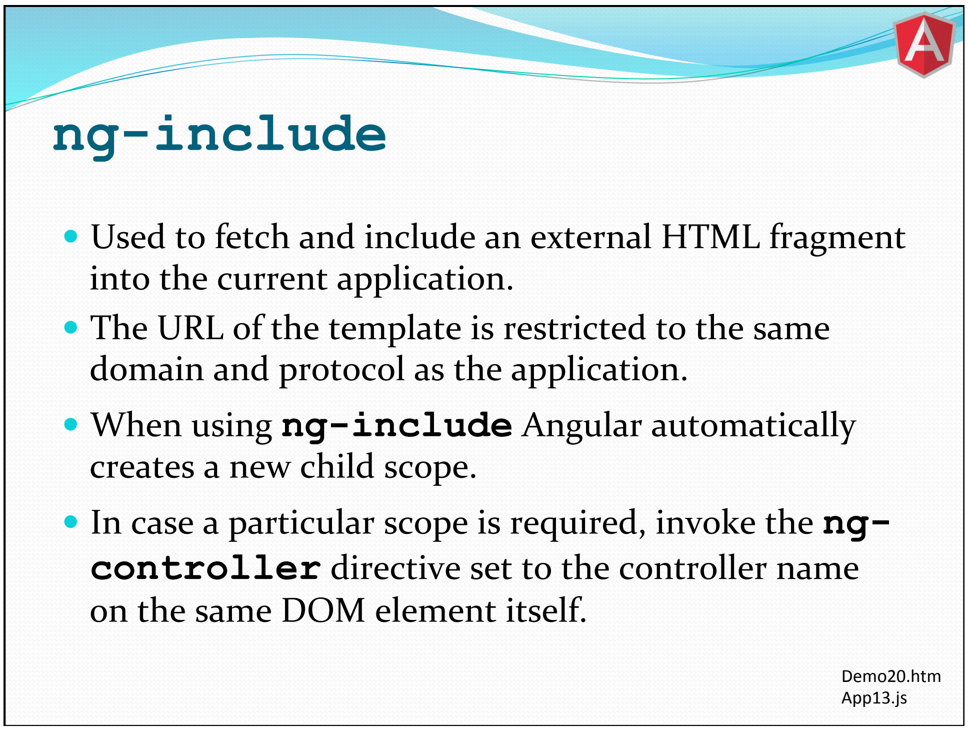


Boolean-like attributes

- **`ng-href`** and **`ng-src`** help prevent errors when changing code in an application.
- They are used in place of **`href`** and **`src`** respectively.
- Angular waits for the properties to be interpolated and then activates their behavior.

```
<a ng-href="{{searchProvider}}">Search!</a>

         <img ng-src="{{imgUrl}}">
```

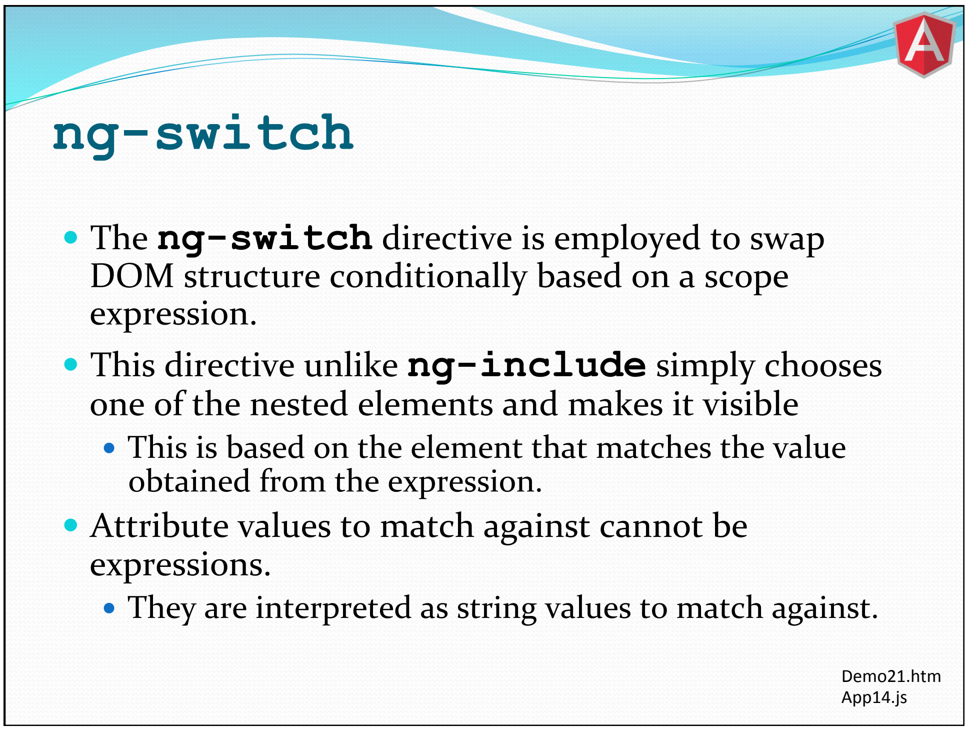
Demo19.htm
App12.js



ng-include

- Used to fetch and include an external HTML fragment into the current application.
- The URL of the template is restricted to the same domain and protocol as the application.
- When using **`ng-include`** Angular automatically creates a new child scope.
- In case a particular scope is required, invoke the **`ng-controller`** directive set to the controller name on the same DOM element itself.

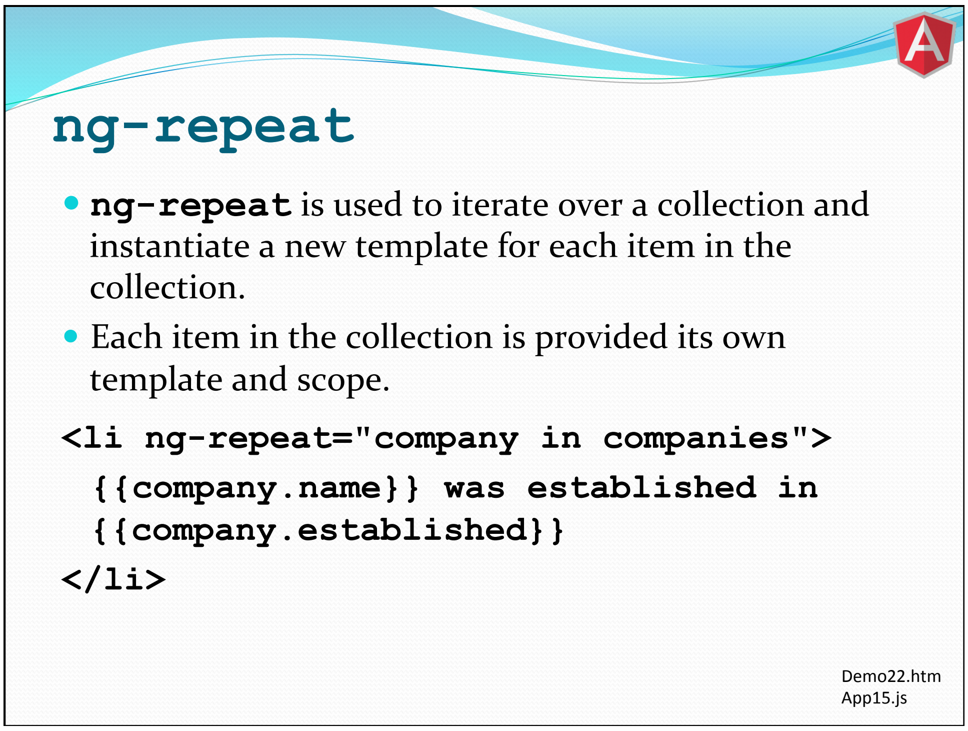
Demo20.htm
App13.js



ng-switch

- The **`ng-switch`** directive is employed to swap DOM structure conditionally based on a scope expression.
- This directive unlike **`ng-include`** simply chooses one of the nested elements and makes it visible
  - This is based on the element that matches the value obtained from the expression.
- Attribute values to match against cannot be expressions.
  - They are interpreted as string values to match against.

Demo21.htm
App14.js

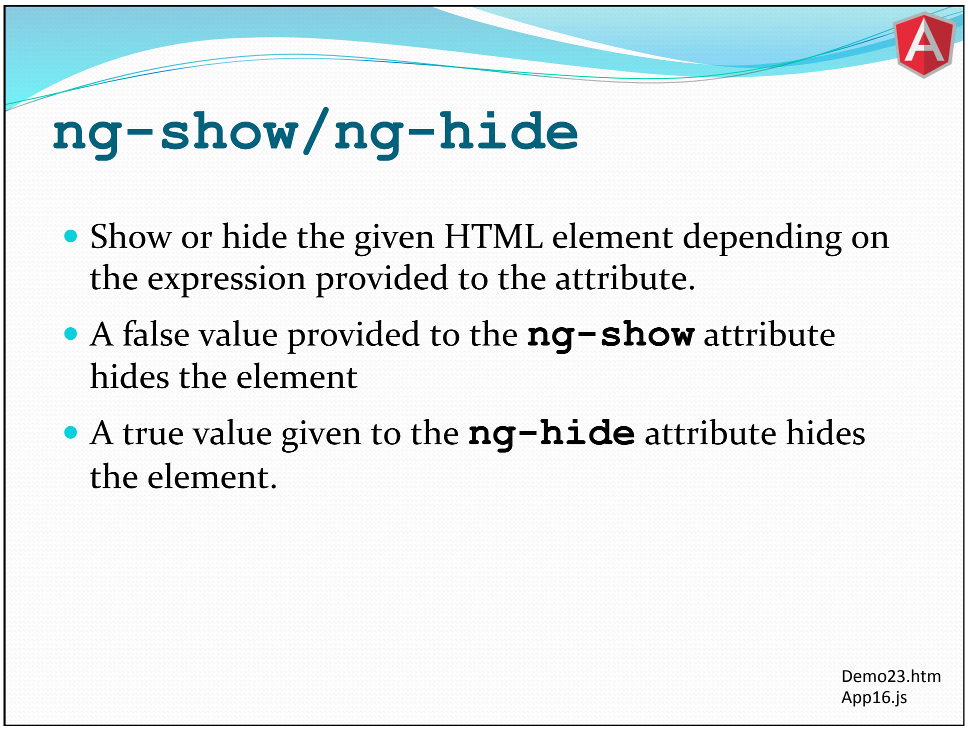


ng-repeat

- **`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>
```

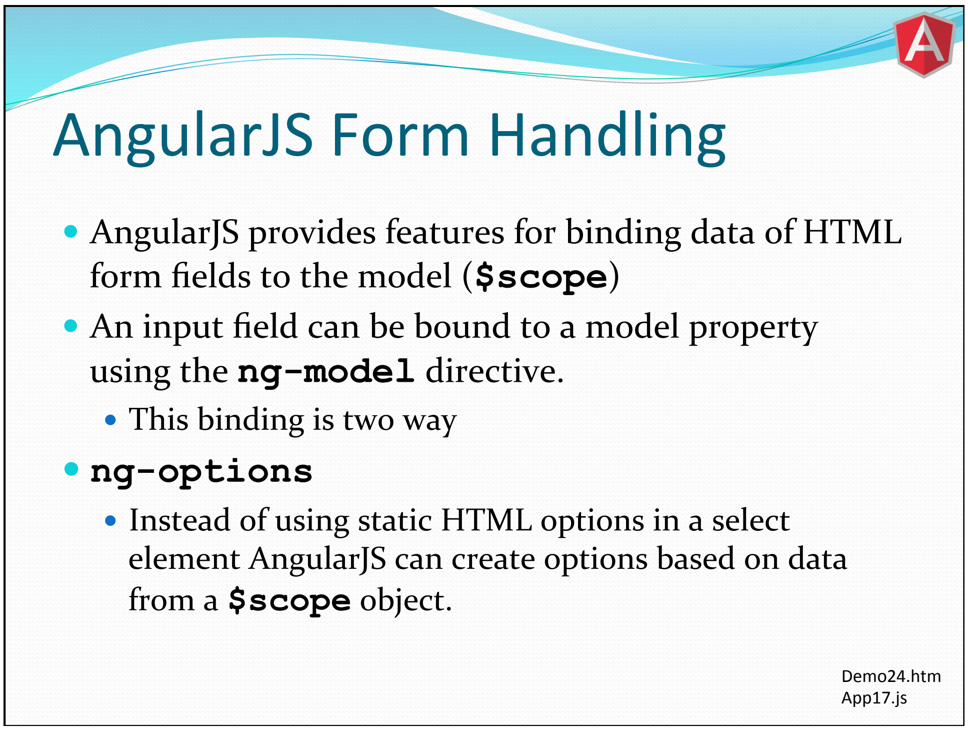
Demo22.htm
App15.js



ng-show/ng-hide

- Show or hide the given HTML element depending on the expression provided to the attribute.
- A false value provided to the **`ng-show`** attribute hides the element
- A true value given to the **`ng-hide`** attribute hides the element.

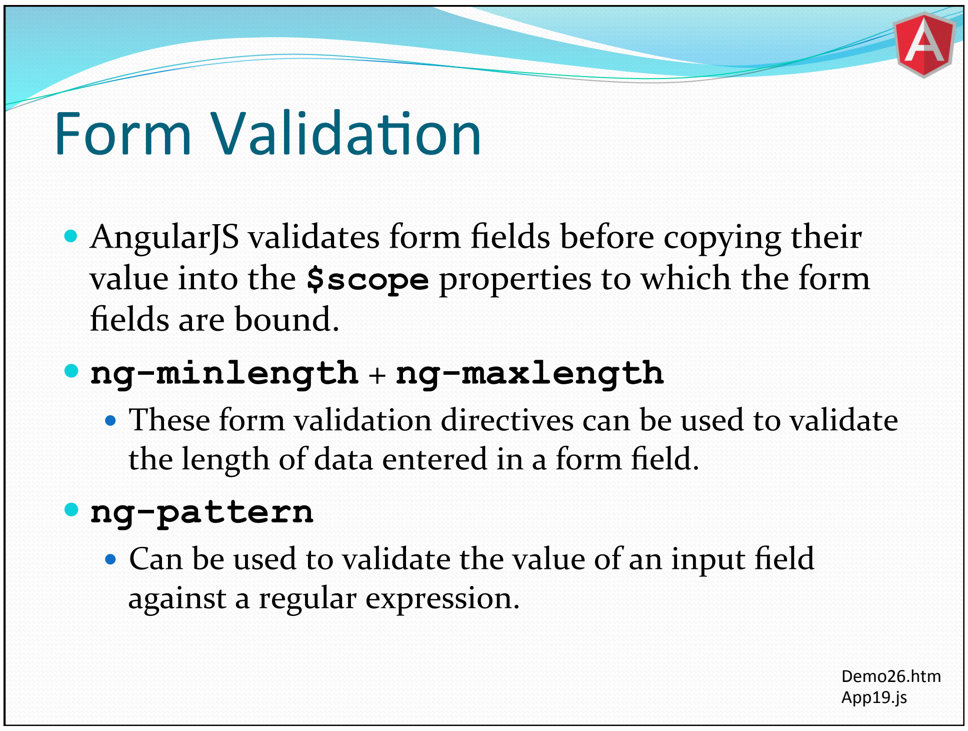
Demo23.htm
App16.js



AngularJS Form Handling

- AngularJS provides features for binding data of HTML form fields to the model (**`$scope`**)
- An input field can be bound to a model property using the **`ng-model`** directive.
  - This binding is two way
- **`ng-options`**
  - Instead of using static HTML options in a select element AngularJS can create options based on data from a **`$scope`** object.

Demo24.htm
App17.js



Form Validation

- AngularJS validates form fields before copying their value into the **`$scope`** properties to which the form fields are bound.
- **`ng-minlength`** + **`ng-maxlength`**
  - These form validation directives can be used to validate the length of data entered in a form field.
- **`ng-pattern`**
  - Can be used to validate the value of an input field against a regular expression.

Demo26.htm
App19.js

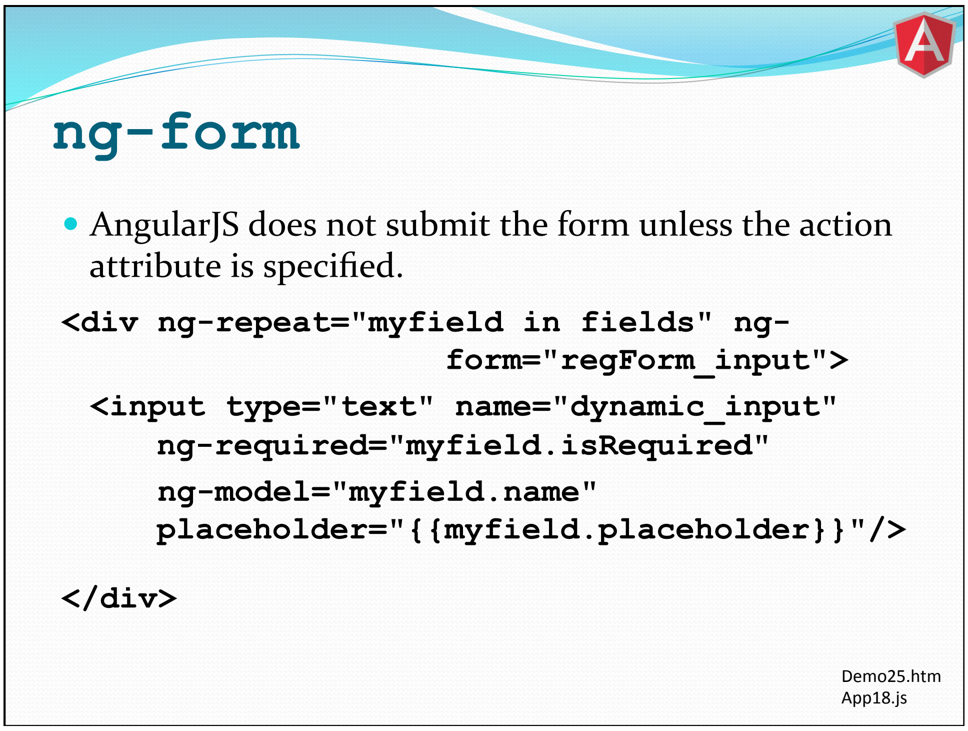
Form Validation…

- Both **`ngFormController`** and **`ngModelController`** objects contain a set of properties to tell if the form or input field is valid.
- **`$pristine`**
  - True if the form has not been changed, false if some fields have been changed
- **`$dirty`**
  - False if the form has not been changed, true if it has
- **`$valid`**
  - True if the form field (or all fields) is valid
- **`$invalid`**
  - False if the field (or all fields) is valid, true otherwise.

Demo27.htm
App20.js

ng-form

- **`ng-form`** is used when a form is to be nested within another.
  - This is a requirement as **`name`** attribute of input elements cannot be generated dynamically.
    - Validating a field using Angular requires a **`name`** attribute!
  - HTML **`<form>`** does not allow nesting of forms.
- Forms can be dynamically generated using **`ng-repeat`**.
- Outer form is valid only when all of the child forms are valid as well.
- The purpose of **`ng-form`** is to group controls and not a replacement for the **`<form>`**

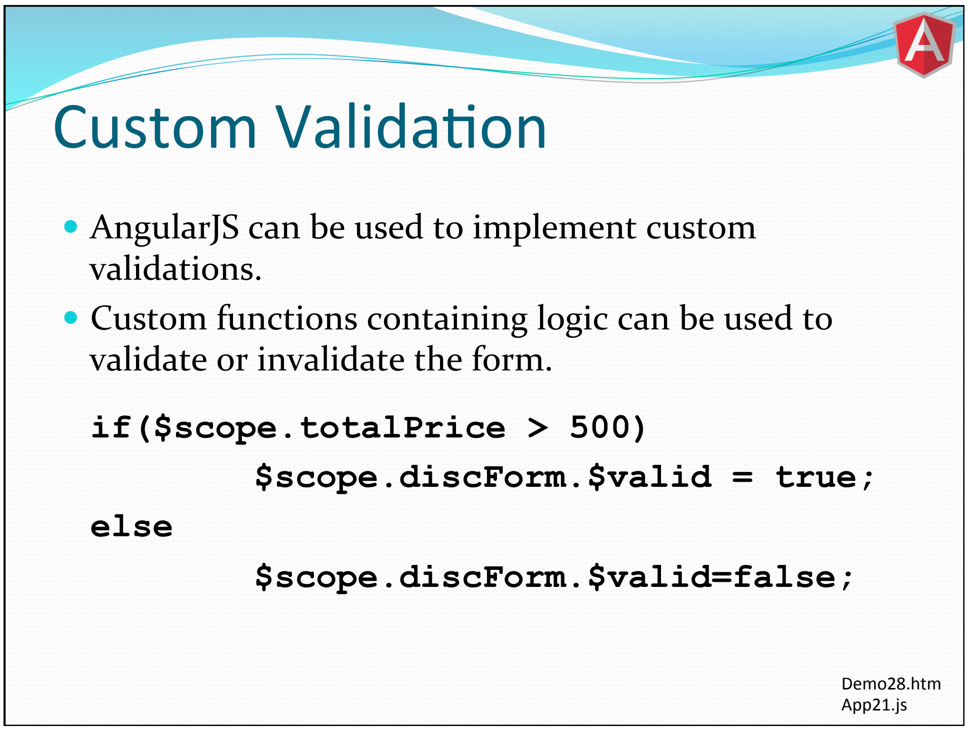


ng-form

- AngularJS does not submit the form unless the action attribute is specified.

```
<div ng-repeat="myfield in fields" ng-
                    form="regForm_input">
  <input type="text" name="dynamic_input"
      ng-required="myfield.isRequired"
      ng-model="myfield.name"
      placeholder="{{myfield.placeholder}}"/>

</div>
```

Demo25.htm
App18.js

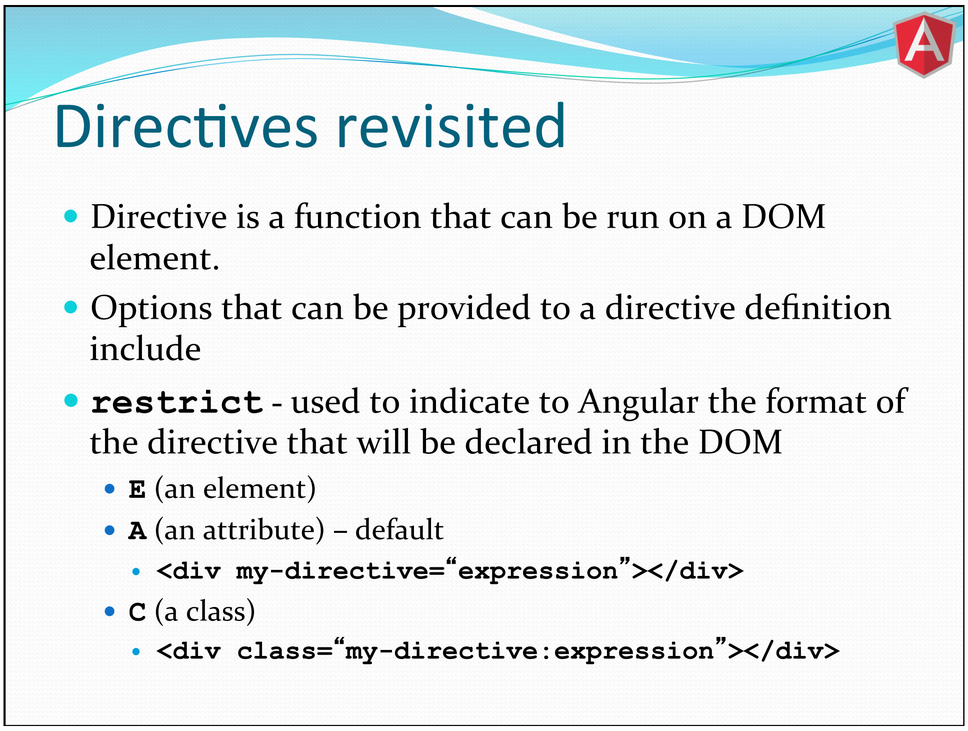


Custom Validation

- AngularJS can be used to implement custom validations.
- Custom functions containing logic can be used to validate or invalidate the form.

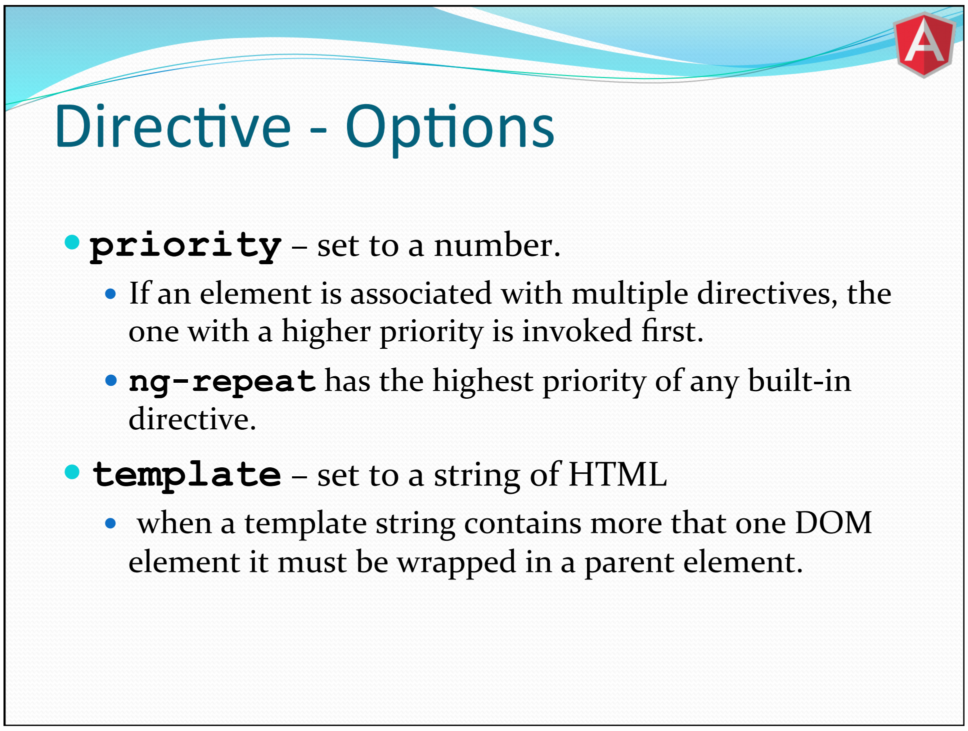
```
if($scope.totalPrice > 500)
        $scope.discForm.$valid = true;
else
        $scope.discForm.$valid=false;
```

Demo28.htm
App21.js



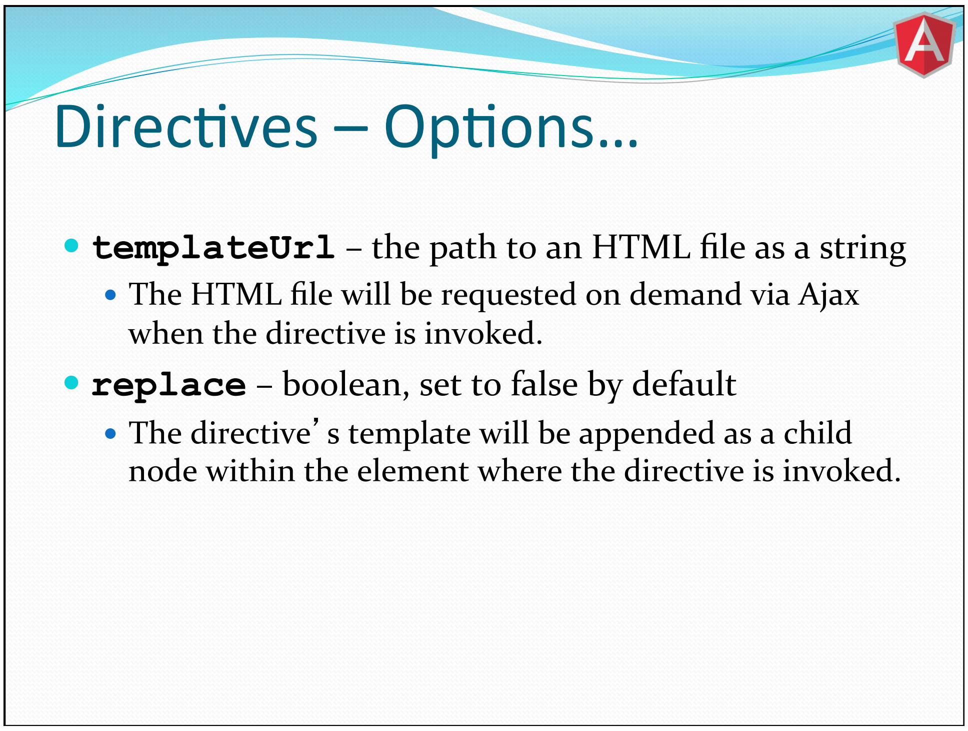
Directives revisited

- Directive is a function that can be run on a DOM element.
- Options that can be provided to a directive definition include
- `restrict` - used to indicate to Angular the format of the directive that will be declared in the DOM
  - `E` (an element)
  - `A` (an attribute) – default
    - `<div my-directive="expression"></div>`
  - `C` (a class)
    - `<div class="my-directive:expression"></div>`



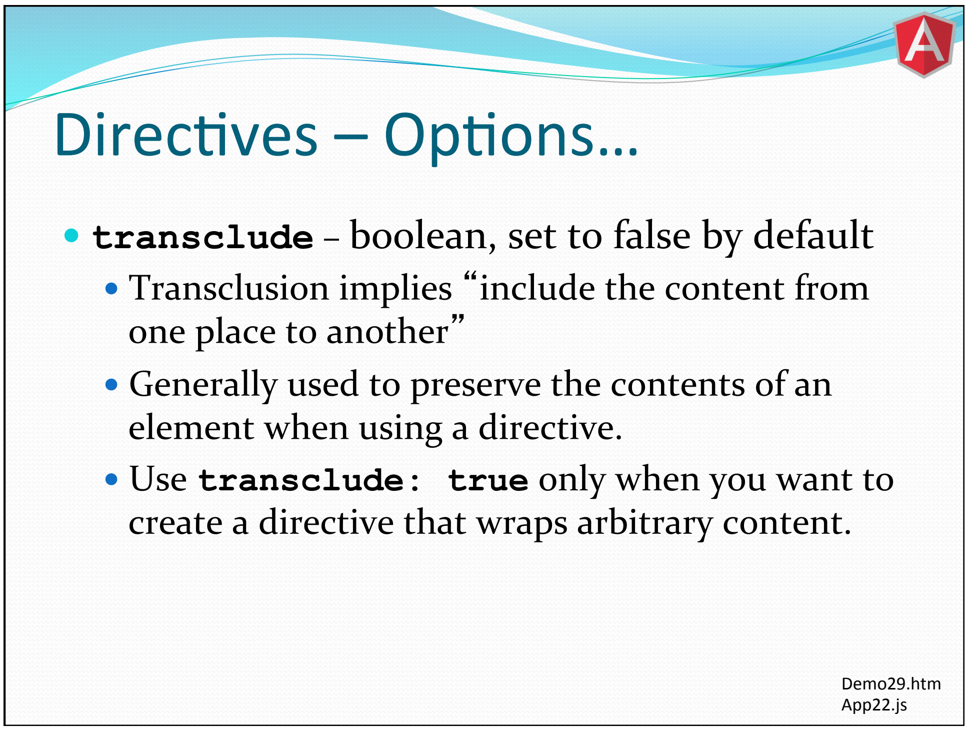
Directive - Options

- `priority` – set to a number.
  - If an element is associated with multiple directives, the one with a higher priority is invoked first.
  - `ng-repeat` has the highest priority of any built-in directive.
- `template` – set to a string of HTML
  - when a template string contains more that one DOM element it must be wrapped in a parent element.



Directives – Options…

- **`templateUrl`** – the path to an HTML file as a string
  - The HTML file will be requested on demand via Ajax when the directive is invoked.
- **`replace`** – boolean, set to false by default
  - The directive's template will be appended as a child node within the element where the directive is invoked.



Directives – Options…

- **`transclude`** – boolean, set to false by default
  - Transclusion implies "include the content from one place to another"
  - Generally used to preserve the contents of an element when using a directive.
  - Use **`transclude: true`** only when you want to create a directive that wraps arbitrary content.

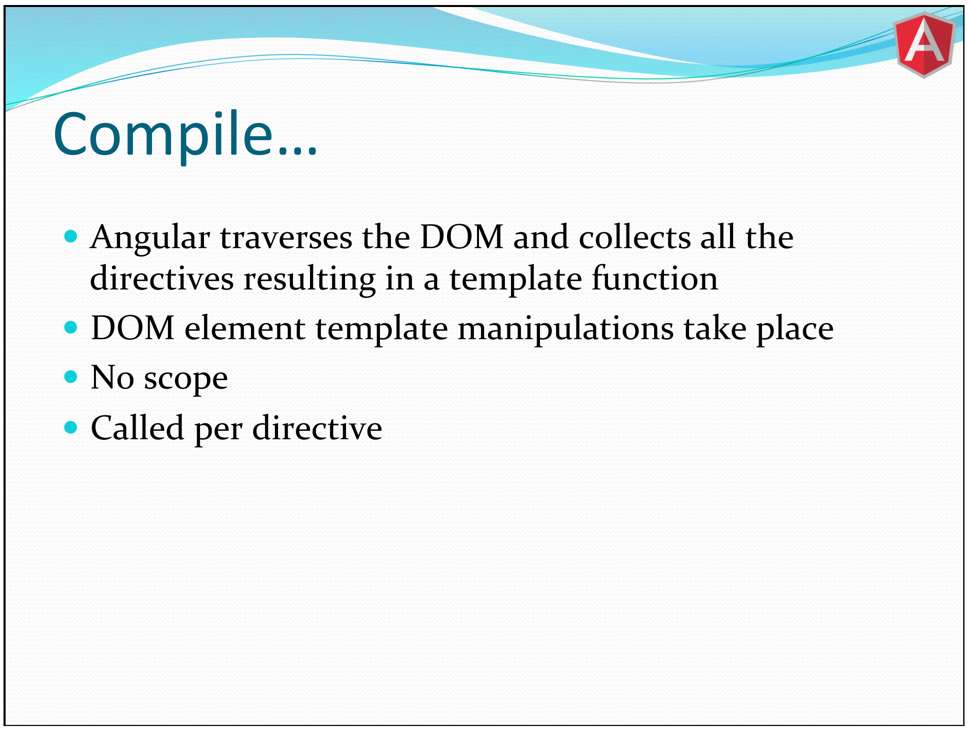
Demo29.htm
App22.js

Directive processing in AngularJS

- AngularJS processes the DOM when it detects that the DOM is ready.
- AngularJS first compiles all directives before it links them to their scope.
- The link phase is split up into pre-link and post-link phases.

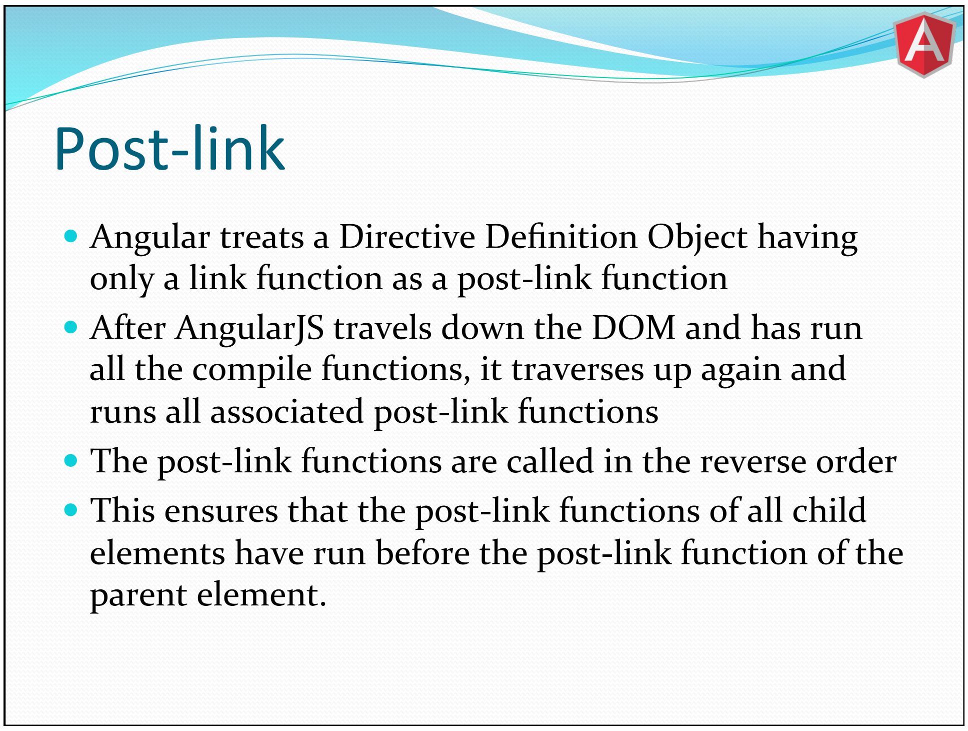
Compile

- A **`compile`** function (if defined) on the Directive Definition Object is called and the element's DOM is passed as an argument
- The DOM of the element is still the DOM that is initially created by the browser using original HTML markup.
- AngularJS recursively traverses deeper into the DOM and repeats the compilation for nested elements if applicable.



Compile…

- Angular traverses the DOM and collects all the directives resulting in a template function
- DOM element template manipulations take place
- No scope
- Called per directive



Post-link

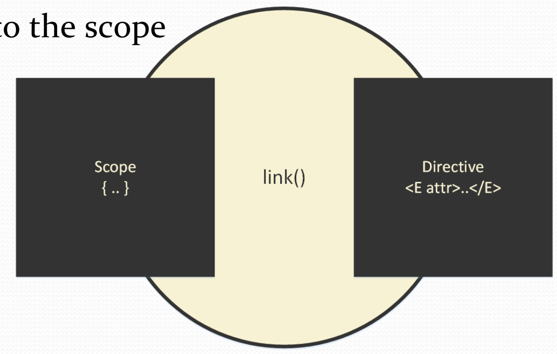
- Angular treats a Directive Definition Object having only a link function as a post-link function
- After AngularJS travels down the DOM and has run all the compile functions, it traverses up again and runs all associated post-link functions
- The post-link functions are called in the reverse order
- This ensures that the post-link functions of all child elements have run before the post-link function of the parent element.

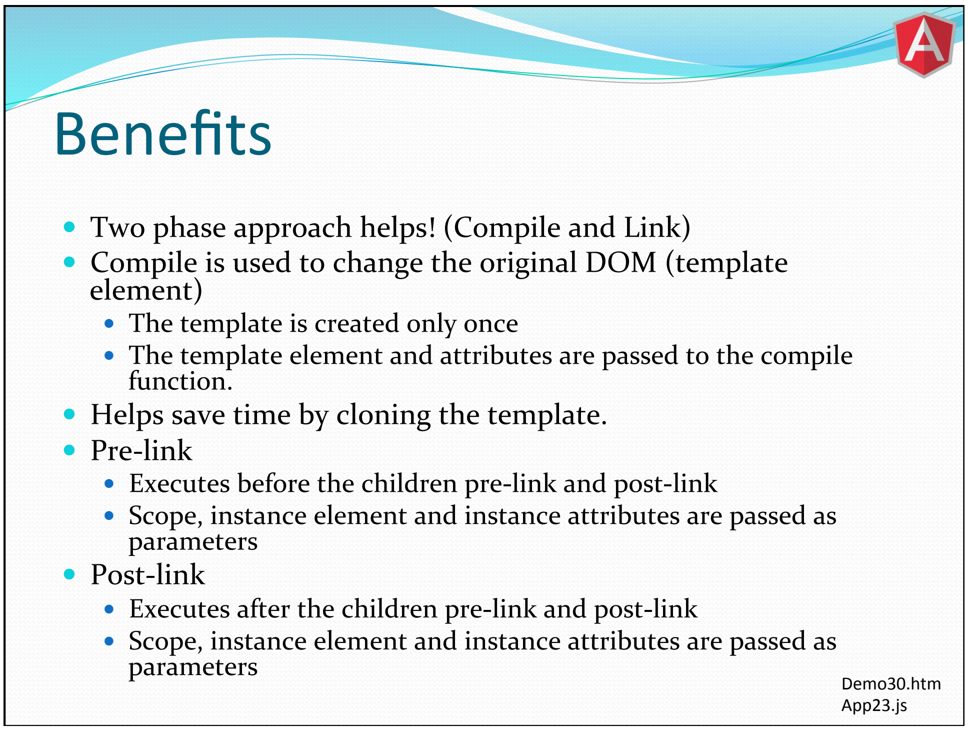
Pre-link

- AngularJS provides a pre-link function to run code before any of the child element’s post-link functions have run
- Pre-link function on an element instance is run before any pre-link or post-link function of any of its child elements.

Linking

- DOM element instance manipulations
- Binds scope to DOM
- Called per element instance
- Binds the directive to the scope

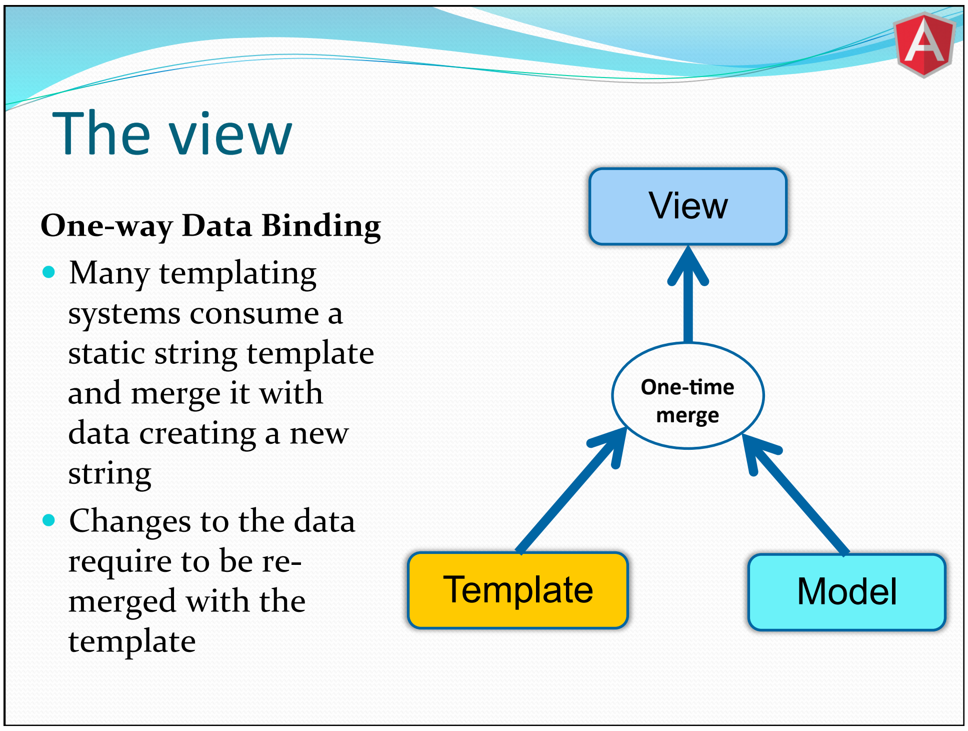




Benefits

- Two phase approach helps! (Compile and Link)
- Compile is used to change the original DOM (template element)
  - The template is created only once
  - The template element and attributes are passed to the compile function.
- Helps save time by cloning the template.
- Pre-link
  - Executes before the children pre-link and post-link
  - Scope, instance element and instance attributes are passed as parameters
- Post-link
  - Executes after the children pre-link and post-link
  - Scope, instance element and instance attributes are passed as parameters

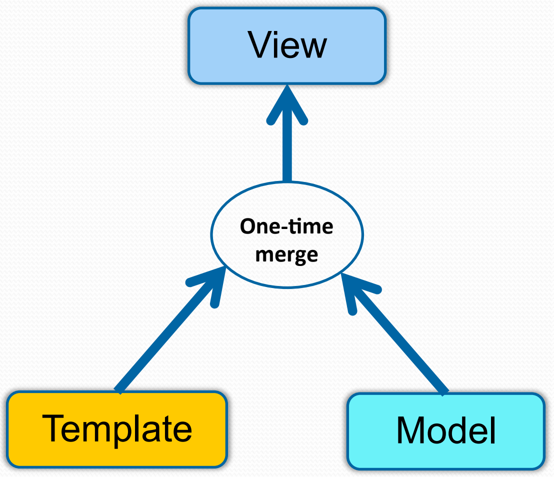
Demo30.htm
App23.js

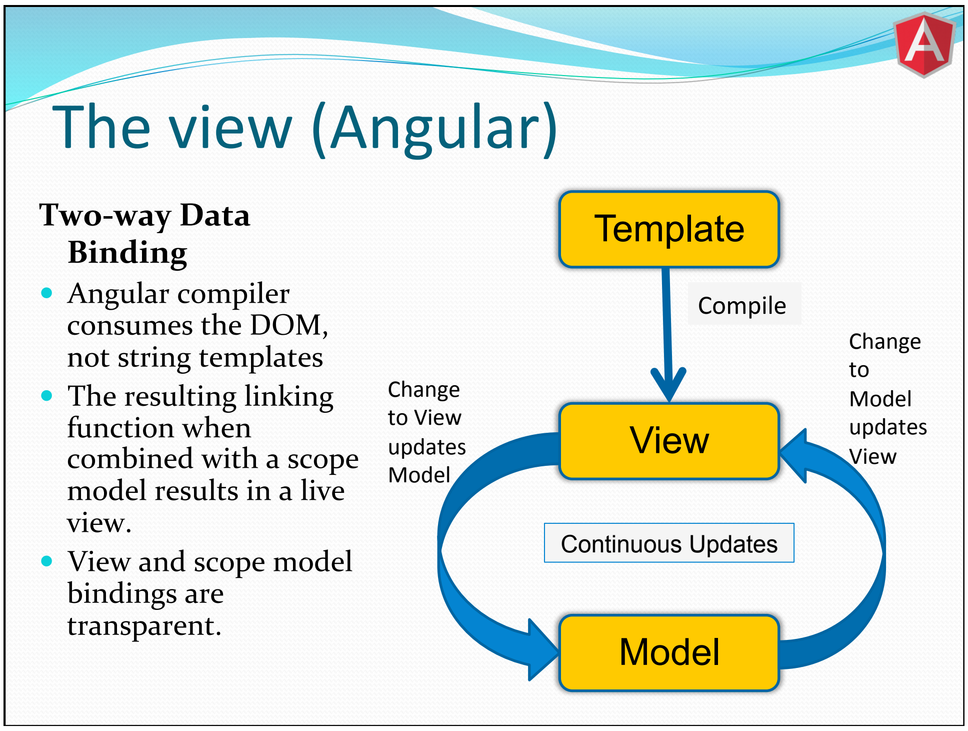


The view

One-way Data Binding

- Many templating systems consume a static string template and merge it with data creating a new string
- Changes to the data require to be re-merged with the template

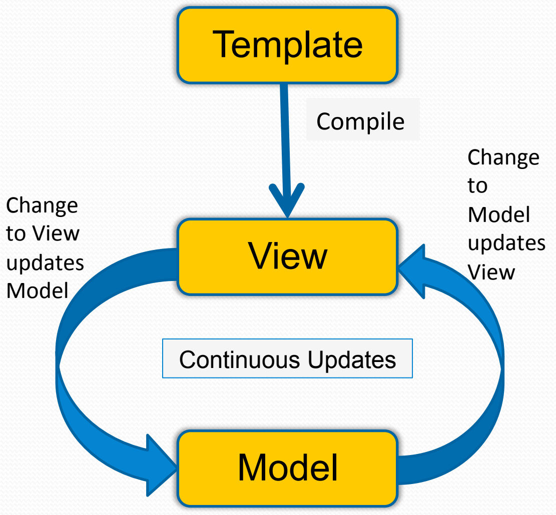


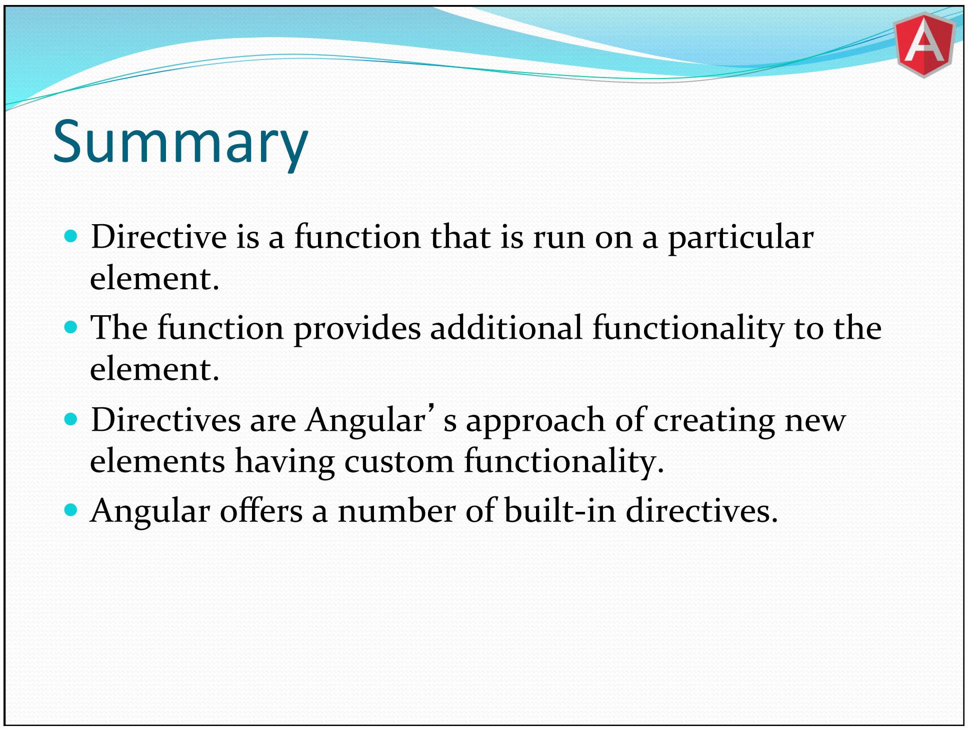


The view (Angular)

Two-way Data Binding

- Angular compiler consumes the DOM, not string templates
- The resulting linking function when combined with a scope model results in a live view.
- View and scope model bindings are transparent.





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

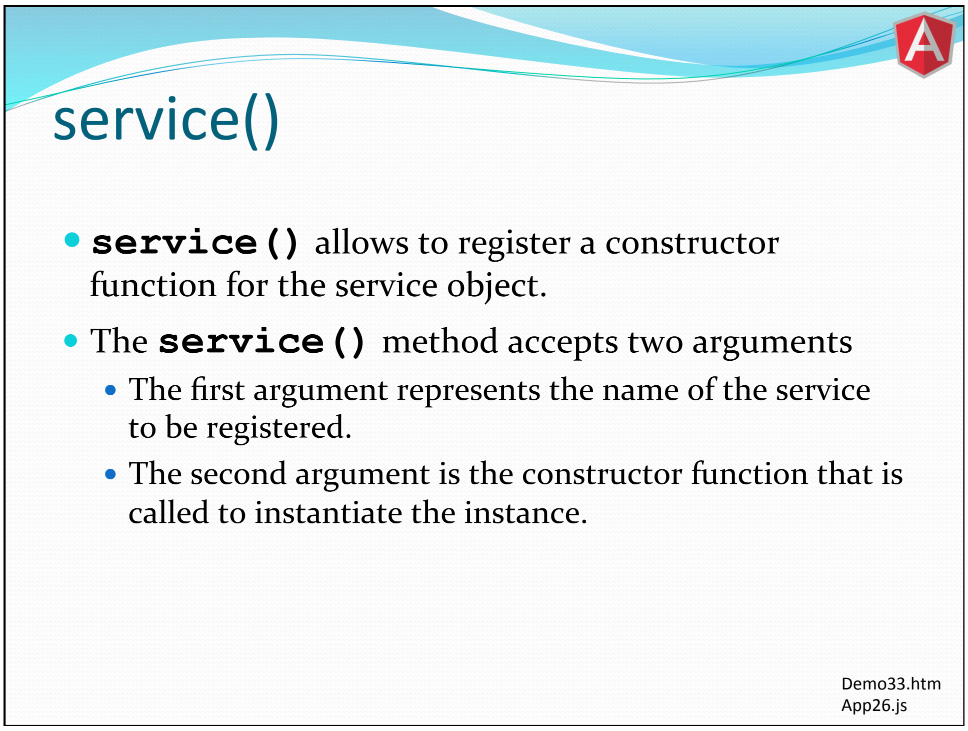
Data sharing between Controllers

- 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

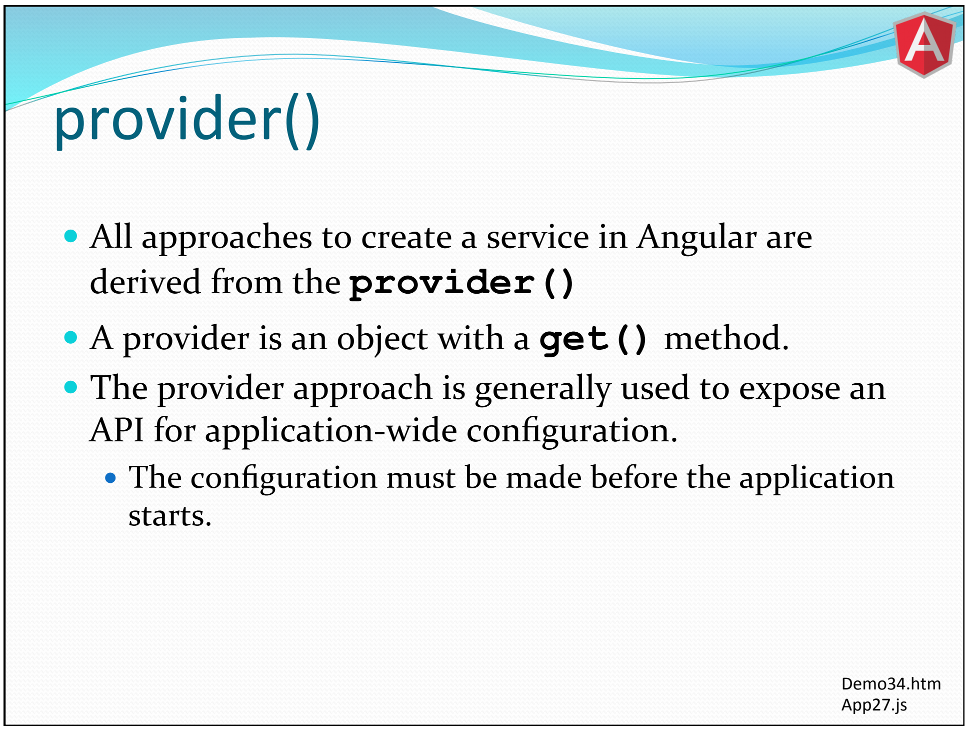
- The most common method for registering a service with the Angular app is through the **`factory()`** method
- The other approaches for creating services are
  - **`service()`**
  - **`provider()`**
  - **`constant()`**



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?

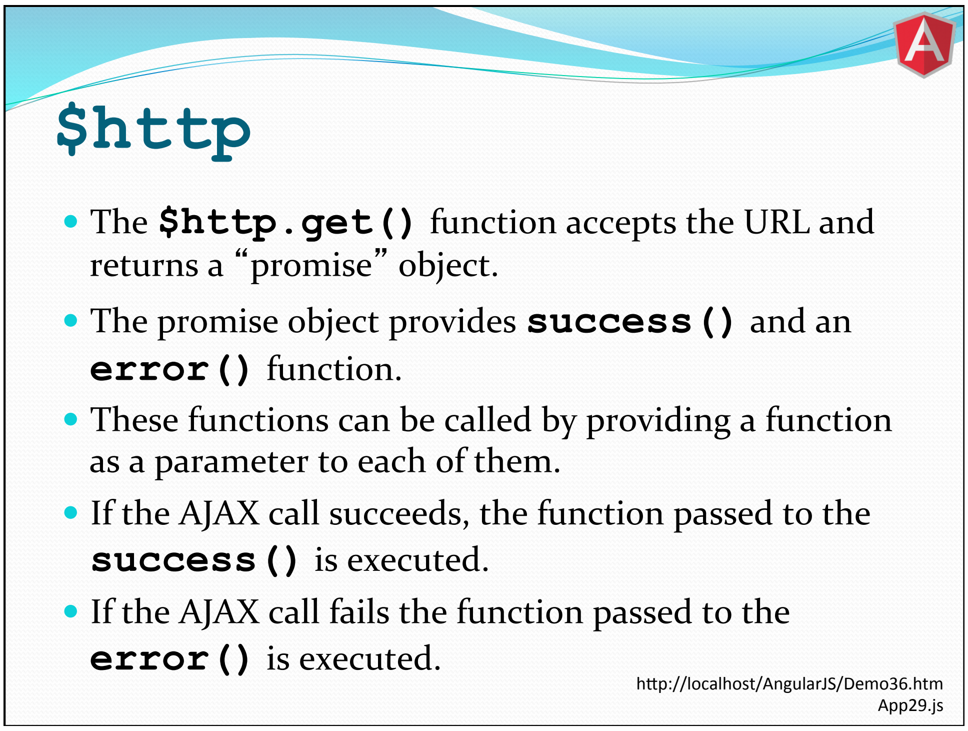
- Use **`constant(name, value)`** to register an existing value, generally used for configuration data.
- In **`factory(name, function())`** we create an object, add properties to it and return that object.
- When using **`service(name, constructor),`** it is instantiated with the 'new' keyword. Hence 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.

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

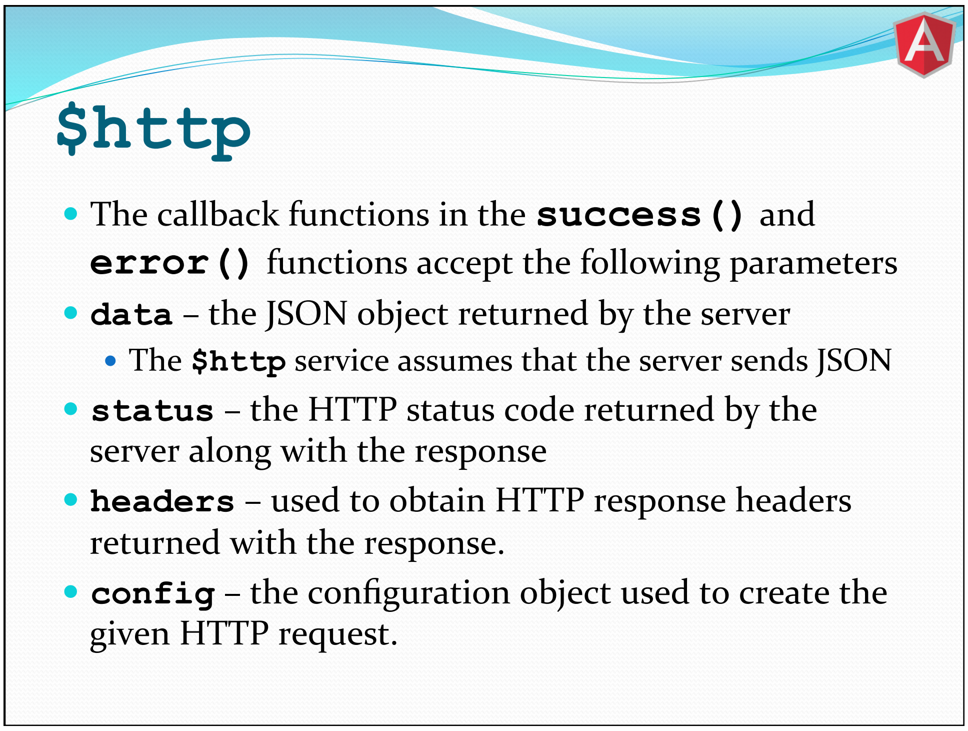
- The **`$http`** service represents a simple approach to send AJAX calls to a web server.
- The **`$http`** service is a wrapper around the browser’s **`XMLHttpRequest`** object.
- AJAX calls cannot be sent to a different domain than the domain from which the HTML page making the AJAX call is loaded.



`$http`

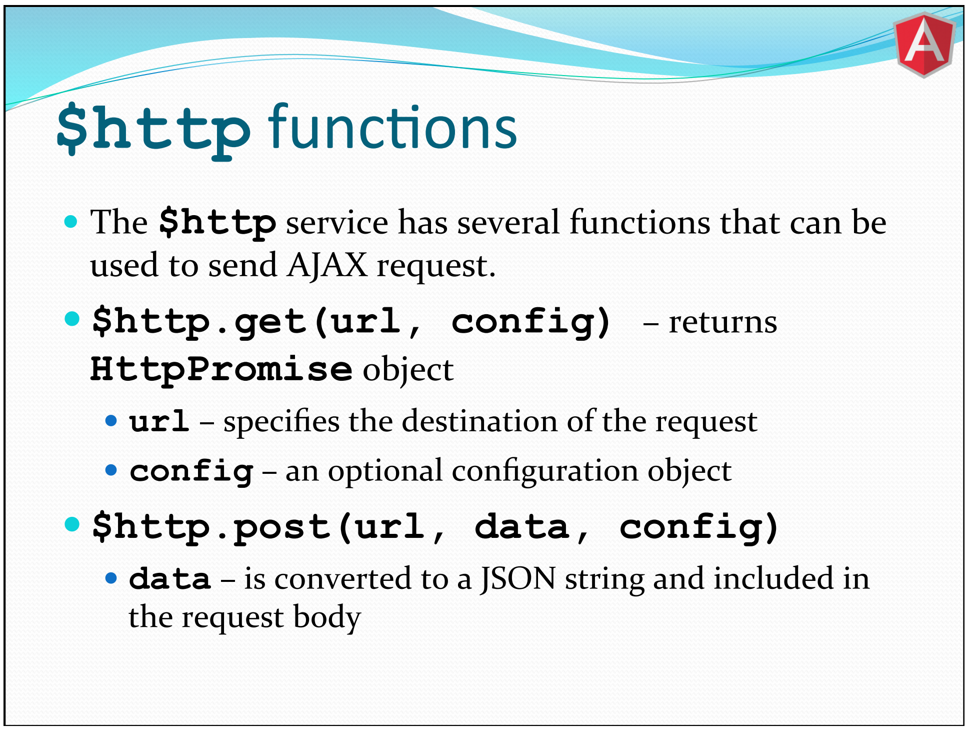
- The **`$http.get()`** function accepts the URL and returns a "promise" object.
- The promise object provides **`success()`** and an **`error()`** function.
- These functions can be called by providing a function as a parameter to each of them.
- If the AJAX call succeeds, the function passed to the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

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



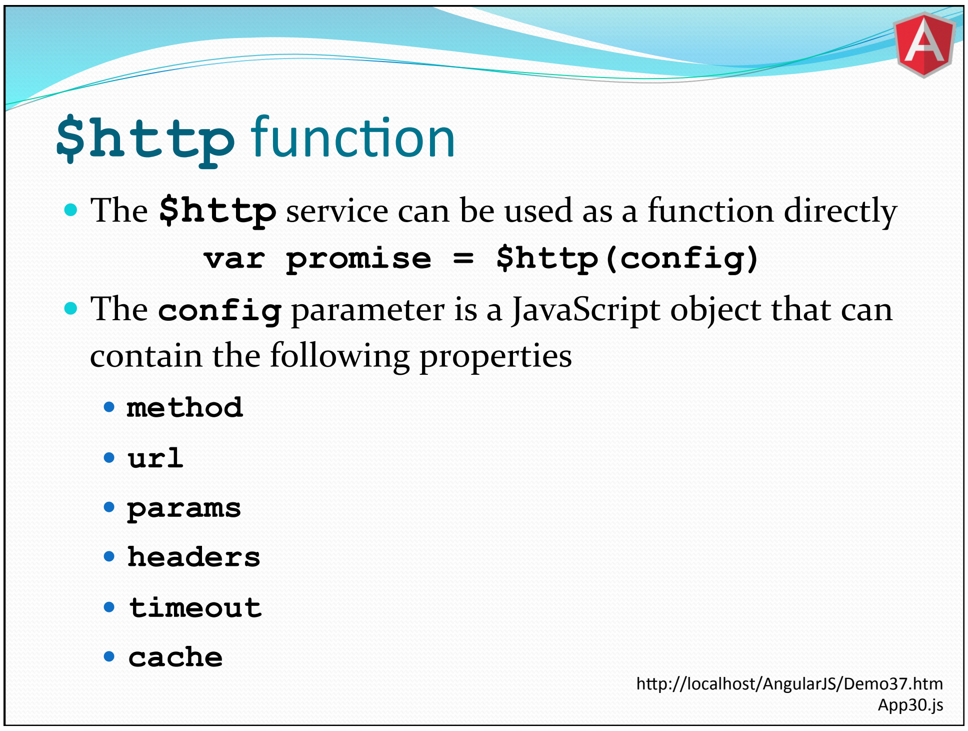
`$http`

- 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

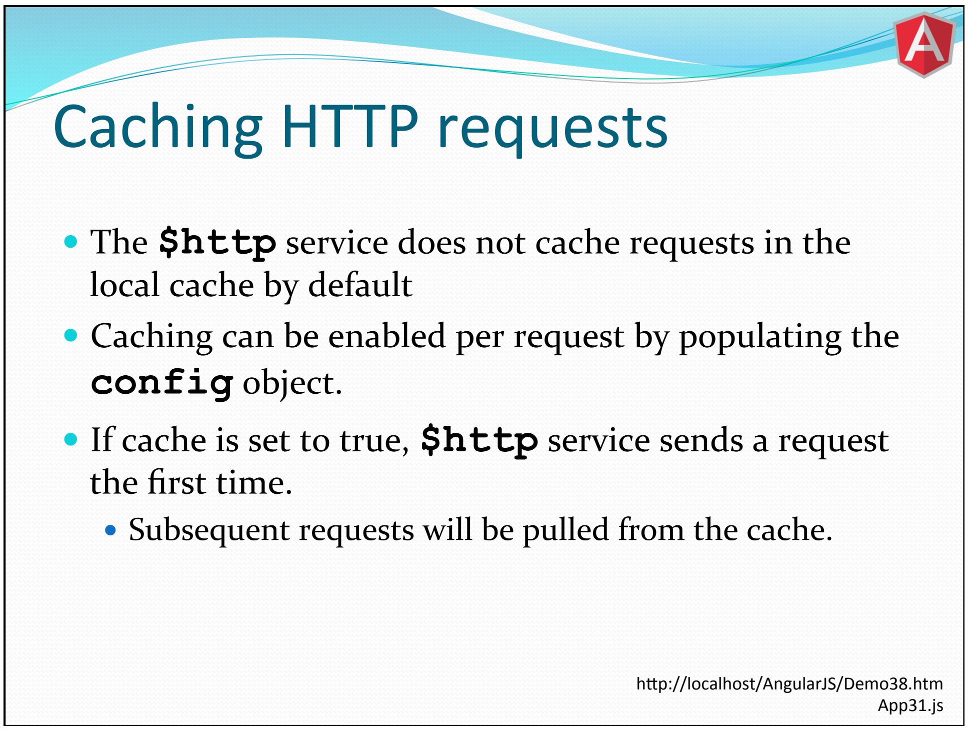


`$http` function

- 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`**

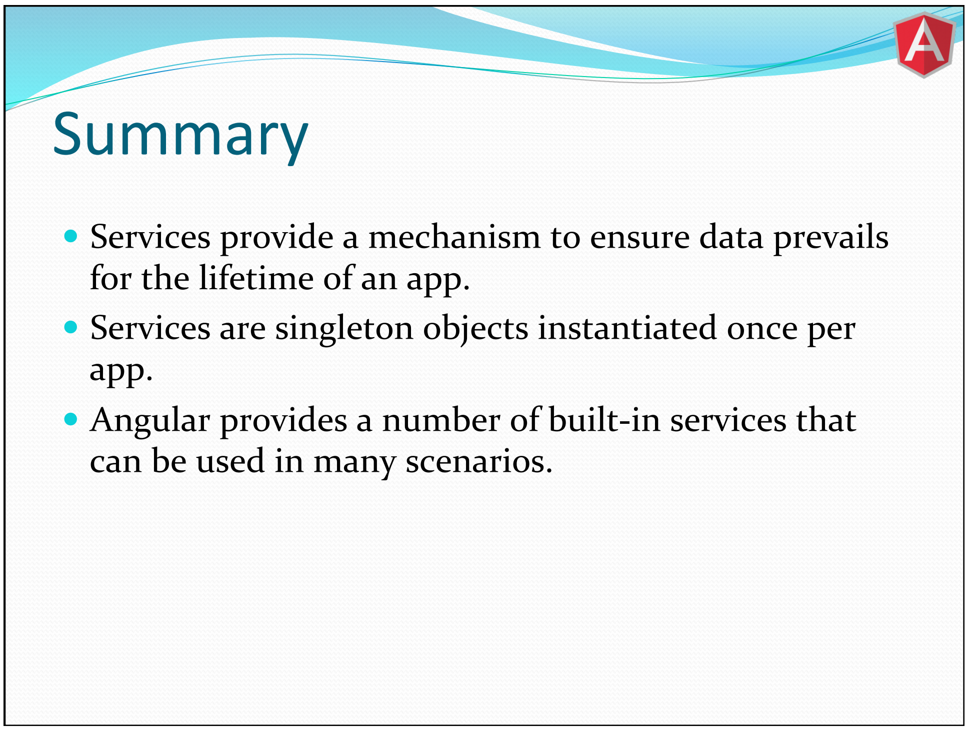
http://localhost/AngularJS/Demo37.htm
App30.js



Caching HTTP requests

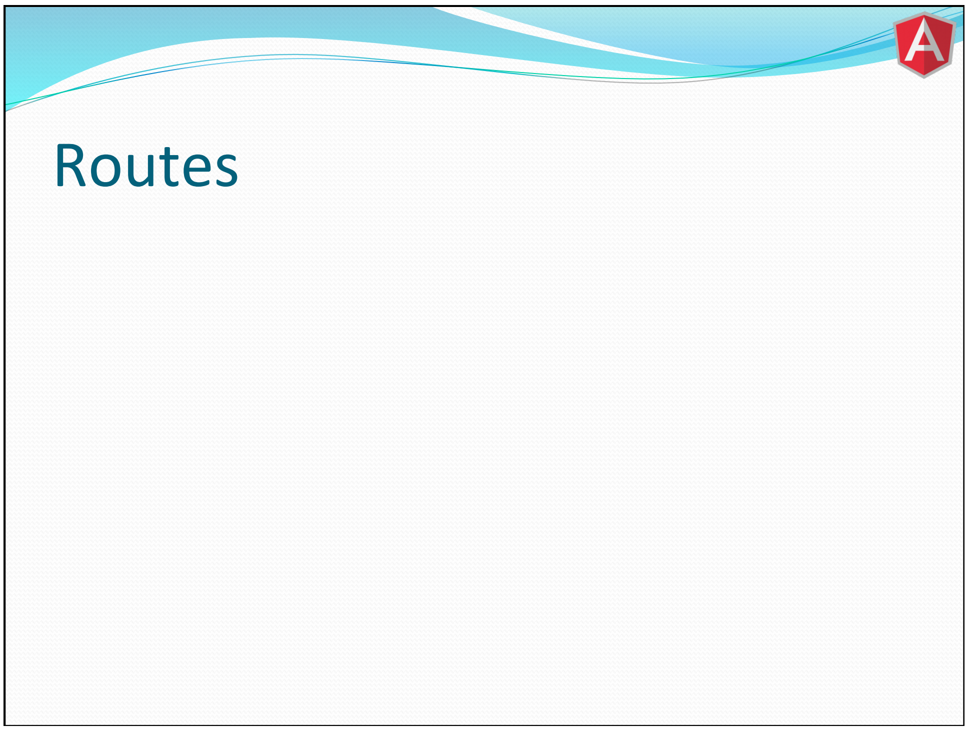
- The **`$http`** service does not cache requests in the local cache by default
- Caching can be enabled per request by populating the **`config`** object.
- If cache is set to true, **`$http`** service sends a request the first time.
  - Subsequent requests will be pulled from the cache.

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

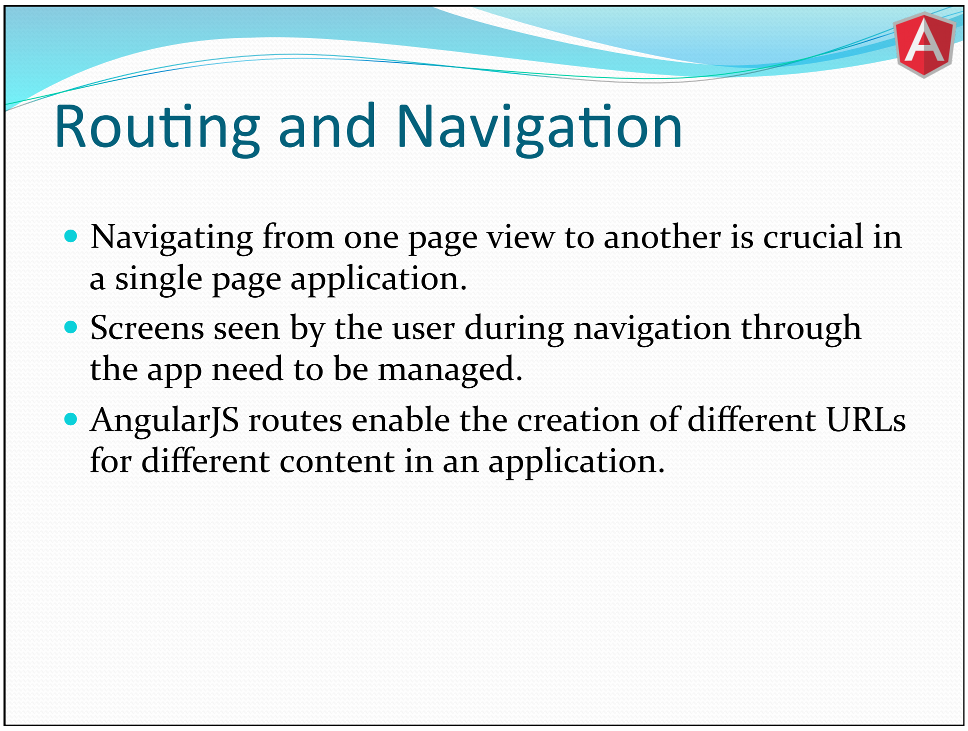


Summary

- 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.



Routes



Routing and Navigation

- Navigating from one page view to another is crucial in a single page application.
- Screens seen by the user during navigation through the app need to be managed.
- AngularJS routes enable the creation of different URLs for different content in an application.

Route

- Having different URLs for different content allows users to bookmark URLs to specific content.
- Each bookmarkable URL in AngularJS is called a *route*.
- AngularJS allows us to declare routes on the **`$routeProvider`**, a provider of the **`$route`** service.
- **`$routeProvider`** helps take advantage of the browser' s history navigation.

Installation

- As of ver 1.2.2 **`ngRoutes`** has been pulled out of the core of Angular into its own module.
- This requires that it be referenced explicitly before use.
- The angular-route is to be referenced in the HTML after the reference to Angular itself

```
<script src="angular.js"></script>
<script src="angular-route.js"></script>
```

Installation…

- The application's AngularJS module needs to declare a dependency on the AngularJS route module.

```
var module = angular.module("sampleApp",
                            ['ngRoute']);
```

- The application's module needs to declare this dependency in order to use the **`ngRoute`** module.

Layout Template

- The **`ng-view`** directive can be used to specify where in the DOM the rendered template of the current route should be placed.
- The **`ng-view`** directive is a special directive that is included with the **`ngRoute`** module.
- It serves as a placeholder for **`$route`** view content.
- **`ng-view`** has a 1000 priority.
  - All other directives on the element with the **`ng-view`** will be overlooked.

`ngView` directive

Steps followed by **`ngView`** directive

- When the **`$routeChangeSuccess`** event is fired, the view is updated
- If a template is associated with the current node
  - Creates a new scope
  - Removes the last view, cleans up the last scope
  - Links the new scope and the new template
  - Links the controller to the scope, if specified

Routes

- In order to create a route on a specific module or app, the **`config`** function is used.

```
angular.module('myApp', [])
    .config(['$routeProvider',
         function($routeProvider) {
    }]);
```

Routes…

- The **when** method is used to add a specific route.
  - The **when** method accepts two parameters
    - Path
    - Route

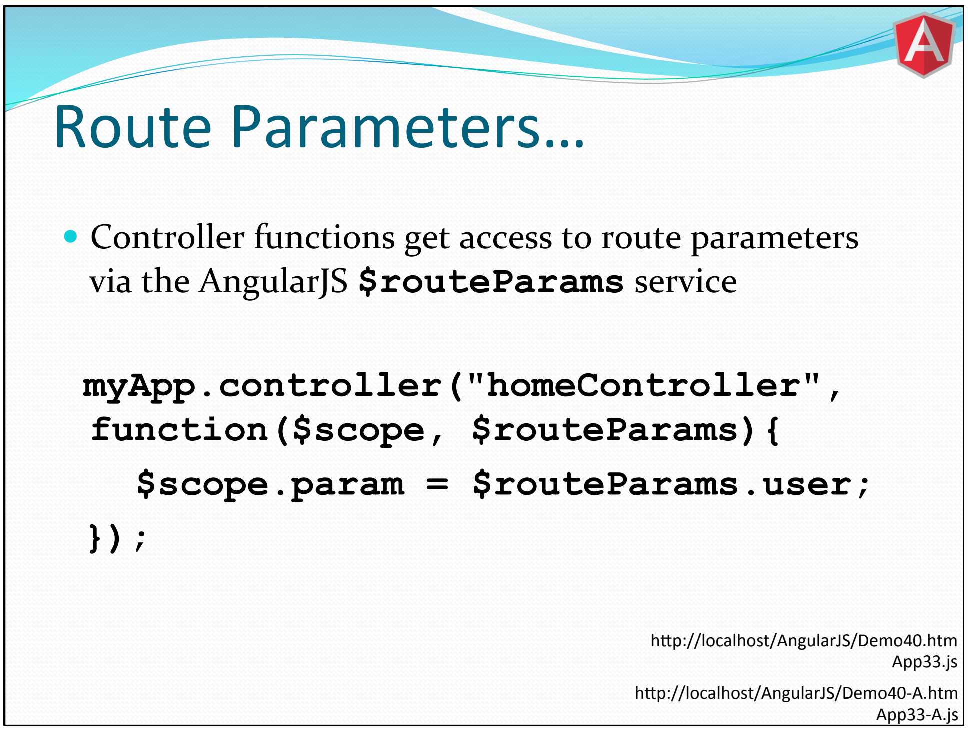
```
angular.module('myApp', [])
.config(['$routeProvider',
     function($routeProvider) {
       $routeProvider
          .when('/', {
              templateUrl: 'views/home.html',
              controller: 'HomeController'
   });
}]);
```

http://localhost/AngularJS/Demo39.htm
App32.js

Route Parameters

- Parameters can be embedded into the route path.

  `<a href="#/home/101A">Home</a><br/>`
- AngularJS extracts from the URL whatever comes after the **`#/route/`**
- AngularJS can extract values from the route path if parameters are defined in the route paths when the **`$routeProvider`** is configured.



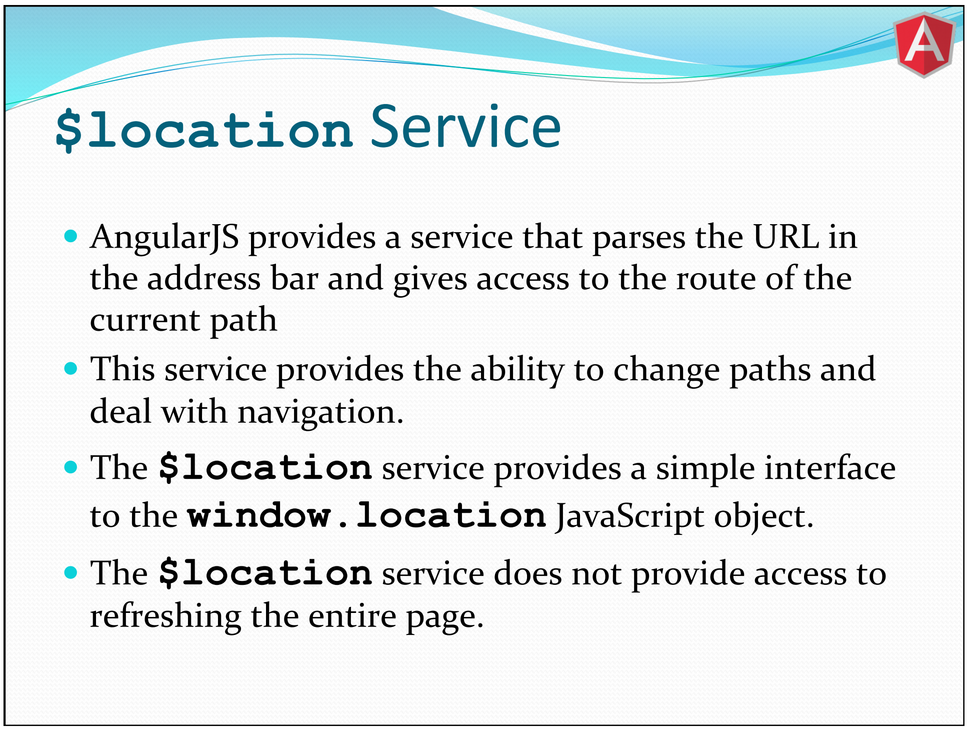
Route Parameters…

- Controller functions get access to route parameters via the AngularJS **`$routeParams`** service

```
myApp.controller("homeController",
 function($scope, $routeParams){
   $scope.param = $routeParams.user;
 });
```

http://localhost/AngularJS/Demo40.htm
App33.js

http://localhost/AngularJS/Demo40-A.htm
App33-A.js



`$location` Service

- AngularJS provides a service that parses the URL in the address bar and gives access to the route of the current path
- This service provides the ability to change paths and deal with navigation.
- The **`$location`** service provides a simple interface to the **`window.location`** JavaScript object.
- The **`$location`** service does not provide access to refreshing the entire page.

`$location` Service

- **`path()`**
  - Returns the current path
  - Can also be used to change the current path and redirect to another URL in the app.
- **`replace()`**
  - Replaces the current path with the new one.
  - **`replace()`** removes the current path from the history

http://localhost/AngularJS/Demo41.htm
App34.js

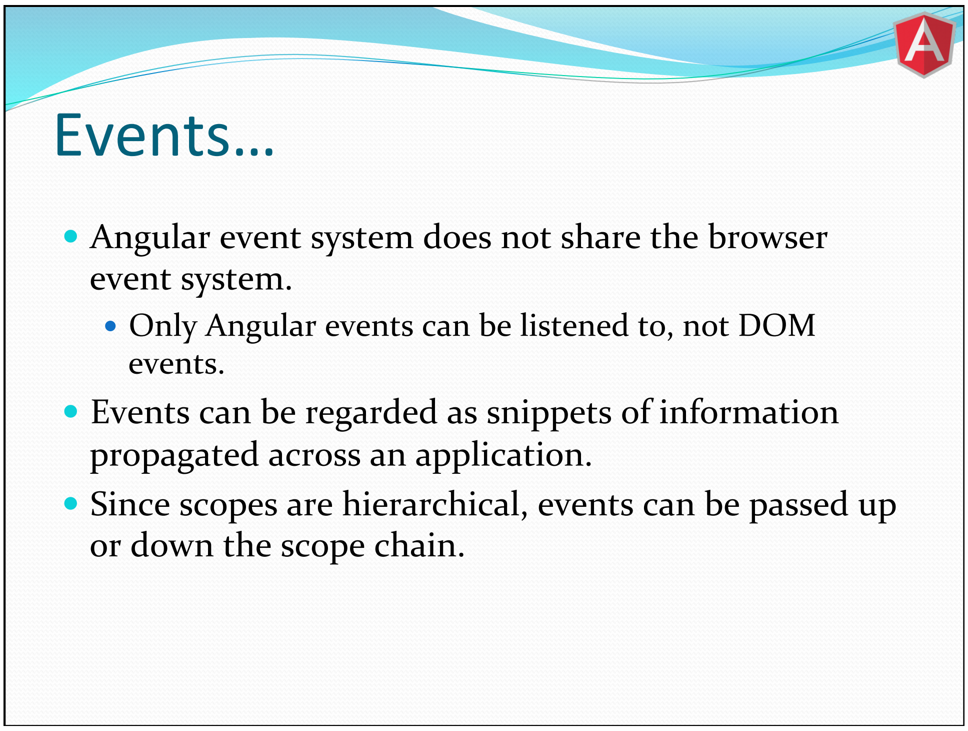
Summary

- Routing helps divide an application into logical views and bind different views to different controllers.
- Routing in Angular is handled by a service provider called **`$routeProvider`**.
- Single Page Applications can be built in a straight-forward manner using Angular capabilities.

Events

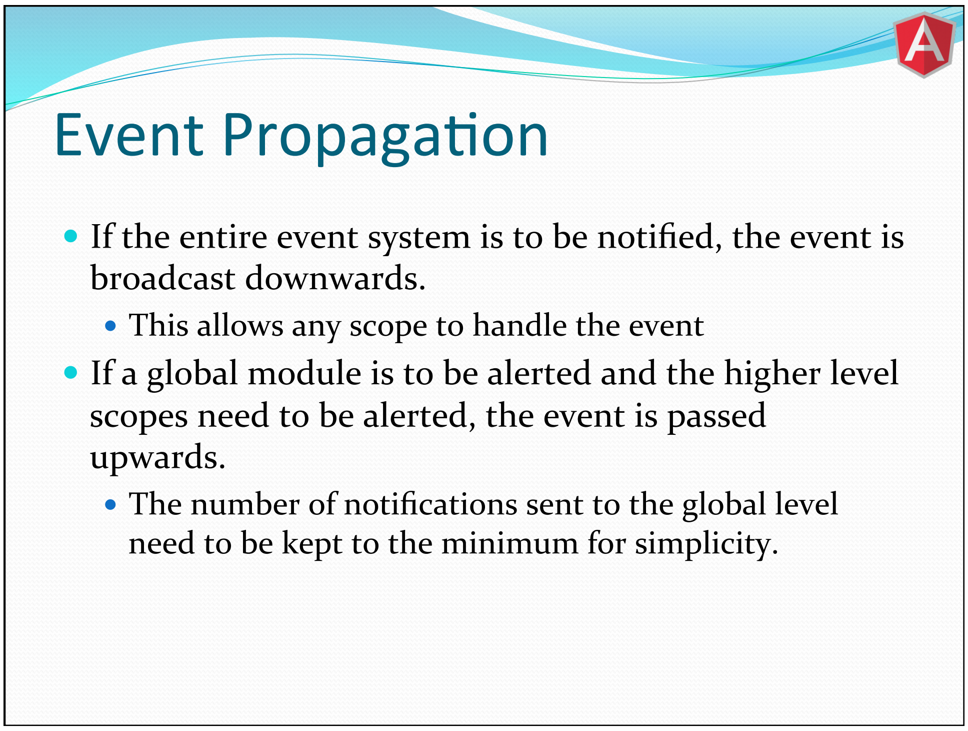
Events

- Angular' s scopes are hierarchical in nature.
- They communicate back and forth through parent-child relationships.
- Scopes don' t share variables and may perform completely different functions from one another.
- Angular provides the ability to communicate between scopes by propagating events up and down the chain.



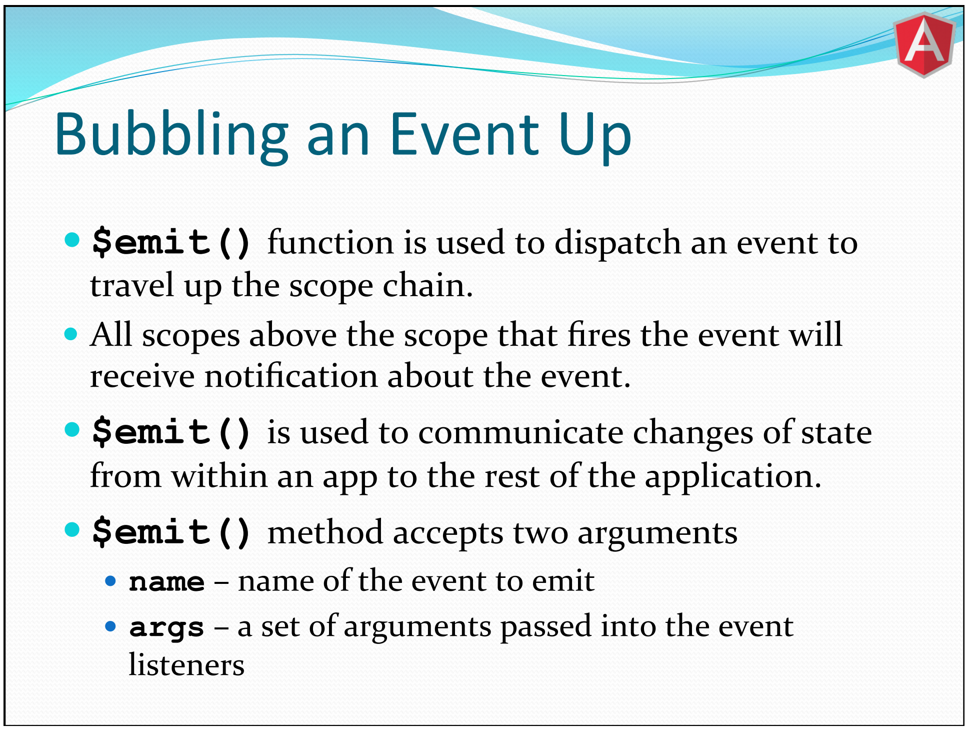
Events…

- Angular event system does not share the browser event system.
  - Only Angular events can be listened to, not DOM events.
- Events can be regarded as snippets of information propagated across an application.
- Since scopes are hierarchical, events can be passed up or down the scope chain.



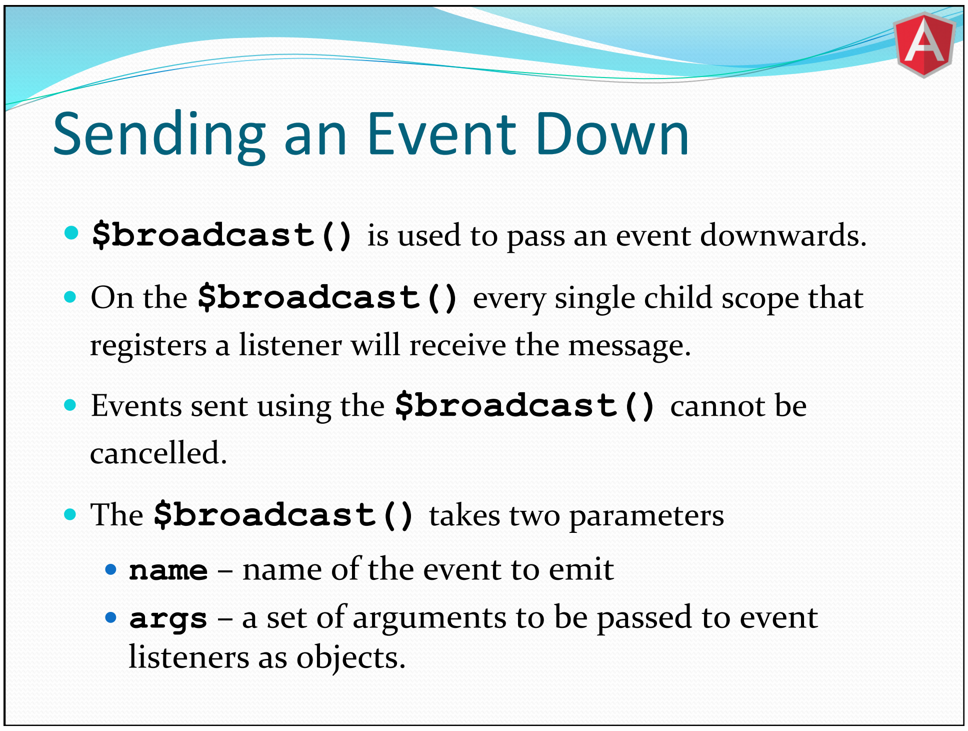
Event Propagation

- If the entire event system is to be notified, the event is broadcast downwards.
  - This allows any scope to handle the event
- If a global module is to be alerted and the higher level scopes need to be alerted, the event is passed upwards.
  - The number of notifications sent to the global level need to be kept to the minimum for simplicity.



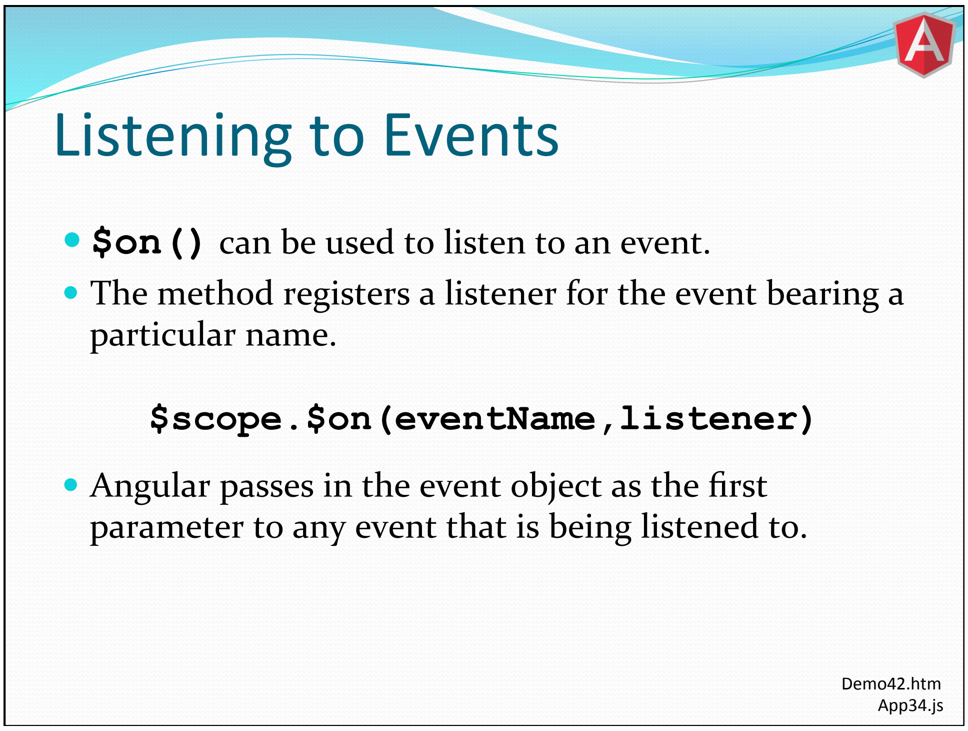
Bubbling an Event Up

- **`$emit()`** function is used to dispatch an event to travel up the scope chain.
- All scopes above the scope that fires the event will receive notification about the event.
- **`$emit()`** is used to communicate changes of state from within an app to the rest of the application.
- **`$emit()`** method accepts two arguments
  - **`name`** – name of the event to emit
  - **`args`** – a set of arguments passed into the event listeners



Sending an Event Down

- **`$broadcast()`** is used to pass an event downwards.
- On the **`$broadcast()`** every single child scope that registers a listener will receive the message.
- Events sent using the **`$broadcast()`** cannot be cancelled.
- The **`$broadcast()`** takes two parameters
  - **`name`** – name of the event to emit
  - **`args`** – a set of arguments to be passed to event listeners as objects.



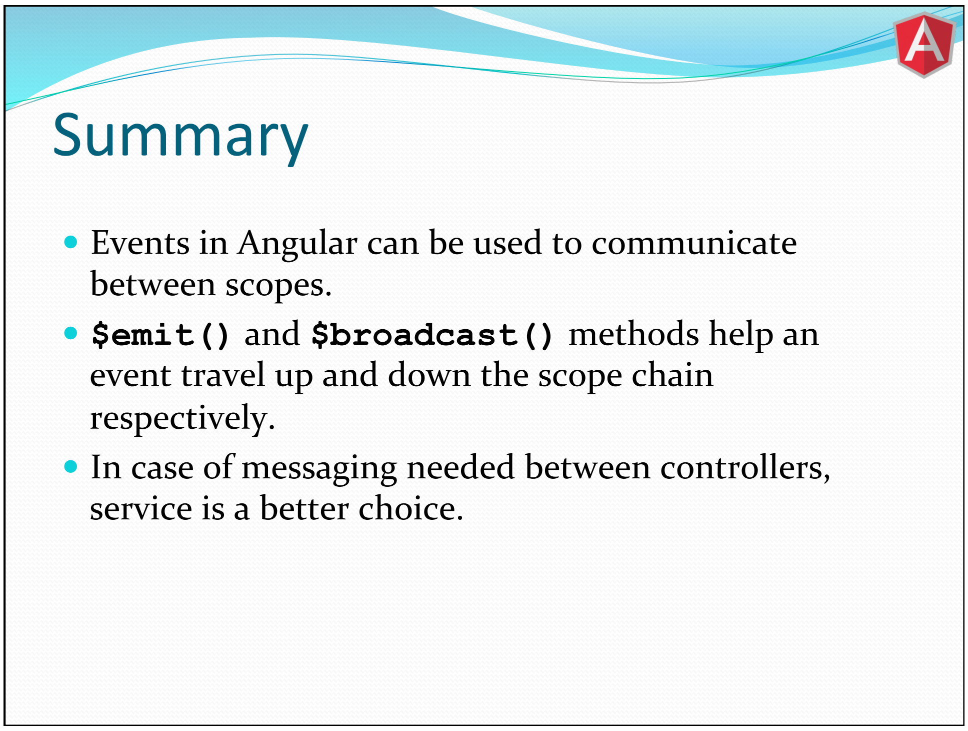
Listening to Events

- `$on()` can be used to listen to an event.
- The method registers a listener for the event bearing a particular name.

```
$scope.$on(eventName,listener)
```

- Angular passes in the event object as the first parameter to any event that is being listened to.

Demo42.htm
App34.js



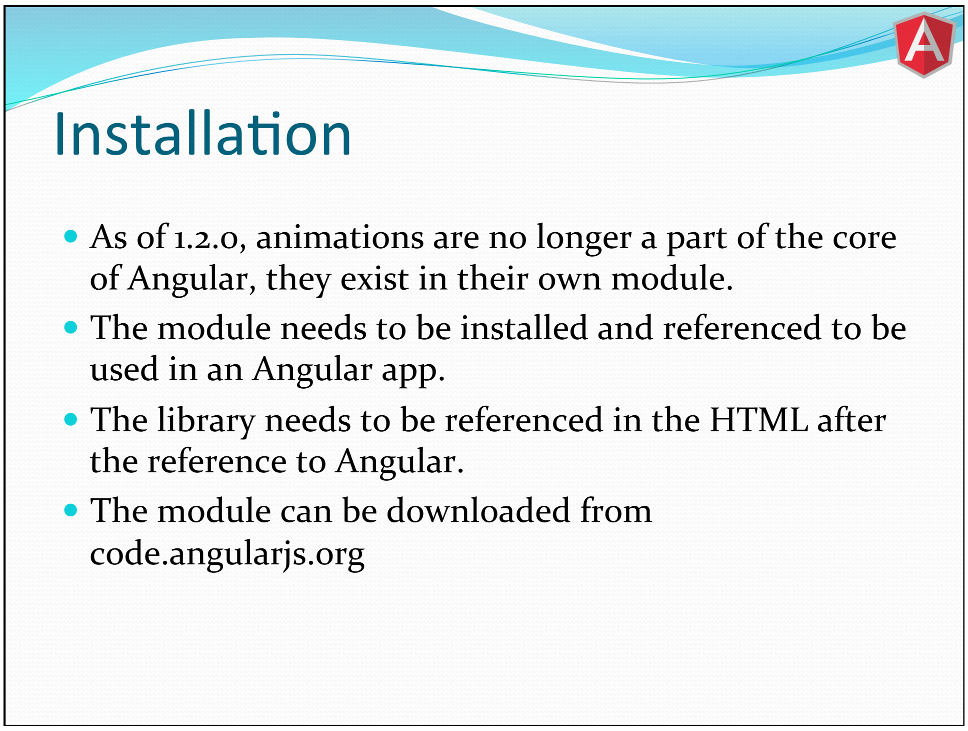
Summary

- Events in Angular can be used to communicate between scopes.
- `$emit()` and `$broadcast()` methods help an event travel up and down the scope chain respectively.
- In case of messaging needed between controllers, service is a better choice.

Angular Animation

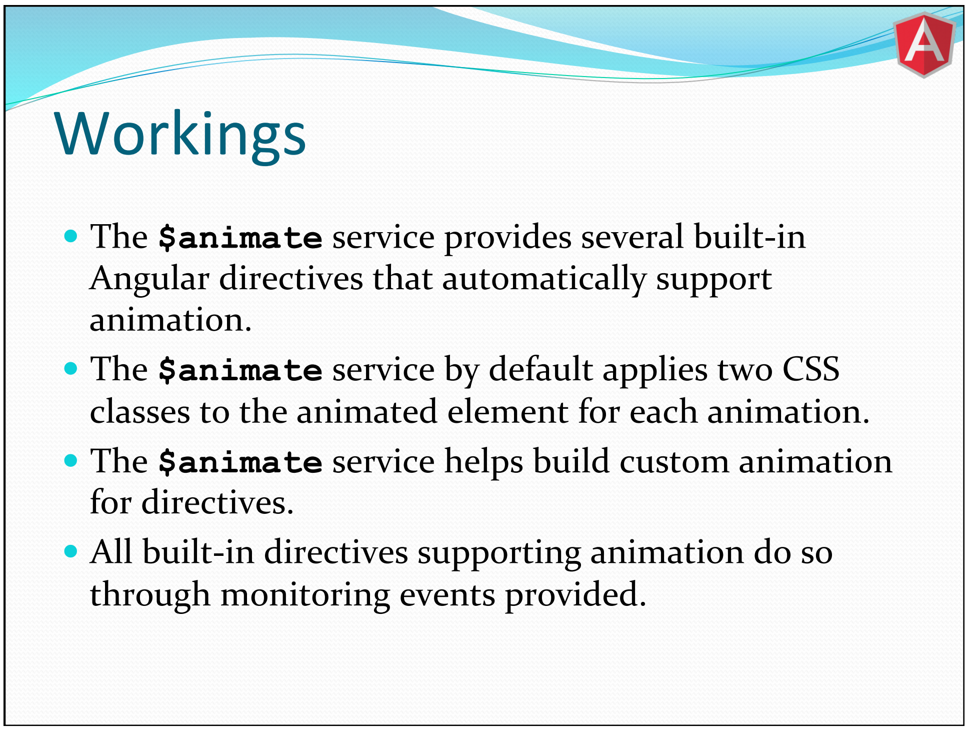
Animation

- The version of Angular (1.2.0 +) includes support for animations.
- Since animation support is built into the framework, animations can be created in a simple manner.
- The Angular team created the **`ngAnimate`** module to provide Angular apps hooks into CSS and JavaScript.
- Animations can be created in multiple ways in an Angular app
  - Using CSS3 animations
  - Using CSS3 transitions
  - Using JavaScript animations



Installation

- As of 1.2.0, animations are no longer a part of the core of Angular, they exist in their own module.
- The module needs to be installed and referenced to be used in an Angular app.
- The library needs to be referenced in the HTML after the reference to Angular.
- The module can be downloaded from code.angularjs.org



Workings

- The **`$animate`** service provides several built-in Angular directives that automatically support animation.
- The **`$animate`** service by default applies two CSS classes to the animated element for each animation.
- The **`$animate`** service helps build custom animation for directives.
- All built-in directives supporting animation do so through monitoring events provided.

Directives and Events

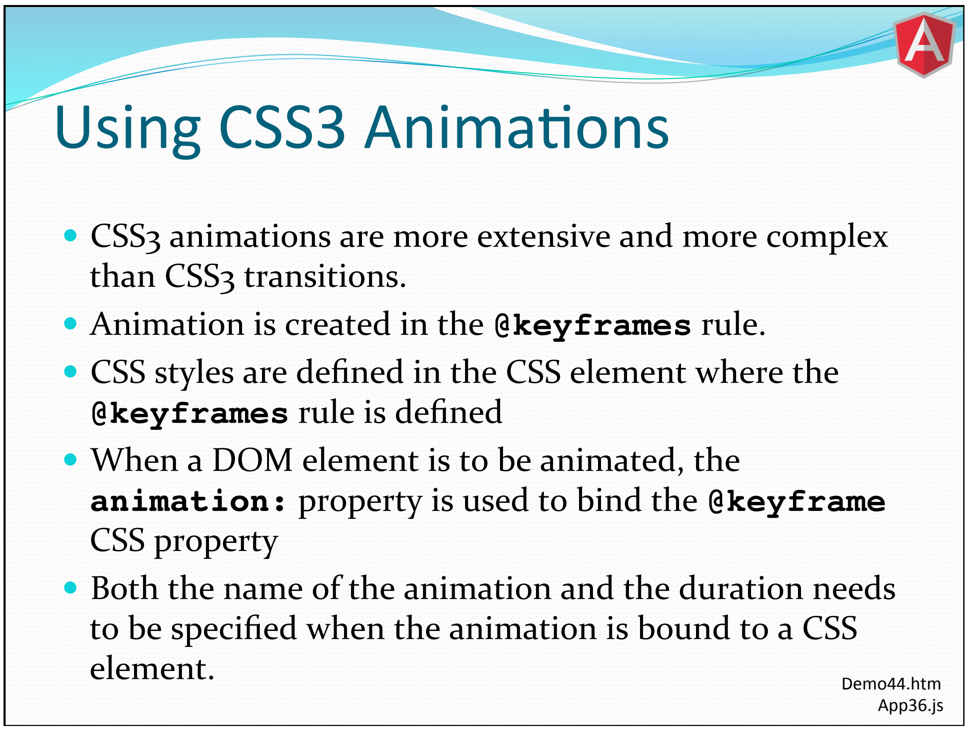
- The **`$animate`** service attaches specific classes based on the events emitted by the directive.
- Built-in directives and the events fired

Directive	Events
`ngRepeat`	`enter, leave, move`
`ngView`	`enter, leave`
`ngInclude`	`enter, leave`
`ngShow`	`add, remove`
`ngHide`	`add, remove`

Using CSS3 Transitions

- CSS3 transitions provide the easiest approach to include animations in an app.
- CSS3 transitions are effects that allow an element to gradually change from one style to another.
- Browsers that don't support CSS3 transitions gracefully fall back to the non-animated version of the app.
- CSS3 transitions are completely class-based.
  - Animation takes place in the browser if the required classes exist.

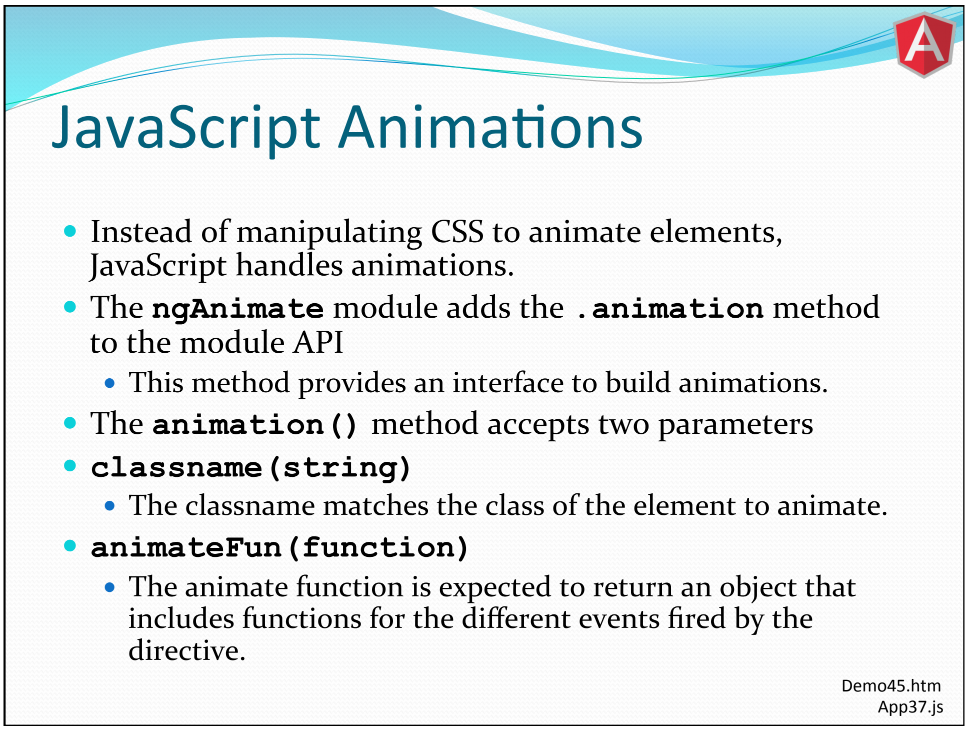
Demo43.htm
App35.js



Using CSS3 Animations

- CSS3 animations are more extensive and more complex than CSS3 transitions.
- Animation is created in the **`@keyframes`** rule.
- CSS styles are defined in the CSS element where the **`@keyframes`** rule is defined
- When a DOM element is to be animated, the **`animation:`** property is used to bind the **`@keyframe`** CSS property
- Both the name of the animation and the duration needs to be specified when the animation is bound to a CSS element.

Demo44.htm
App36.js



JavaScript Animations

- Instead of manipulating CSS to animate elements, JavaScript handles animations.
- The **`ngAnimate`** module adds the **`.animation`** method to the module API
  - This method provides an interface to build animations.
- The **`animation()`** method accepts two parameters
- **`classname(string)`**
  - The classname matches the class of the element to animate.
- **`animateFun(function)`**
  - The animate function is expected to return an object that includes functions for the different events fired by the directive.

Demo45.htm
App37.js

Summary

- AngularJS comes packed with animation support via the **`ngAnimate`** module.
- **`ngAnimate`** needs to be included as a dependency within the application.
- Animations are entirely class-based.

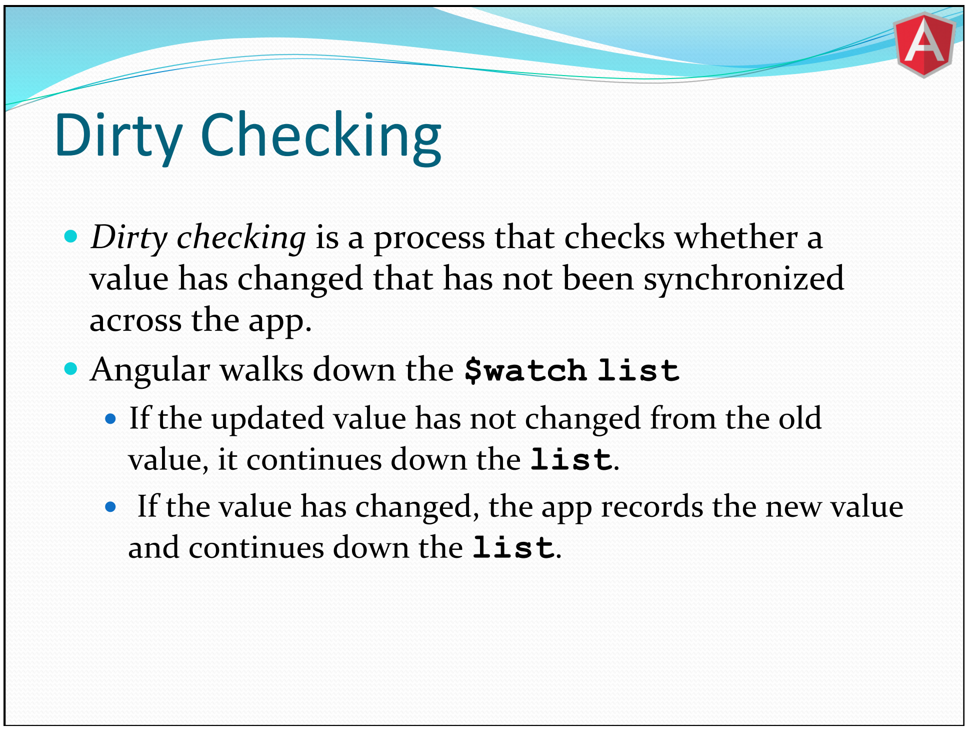
Under the Hood

Inner workings!

- Whenever the browser detects an event, the function registered with the **`addEventListener`** is called.
- With the inclusion of Angular, the normal browser flow is extended to create an Angular context.
- The Angular context refers to the code that runs in the Angular event loop.
- This event loop is referred to as the **`$digest`** loop

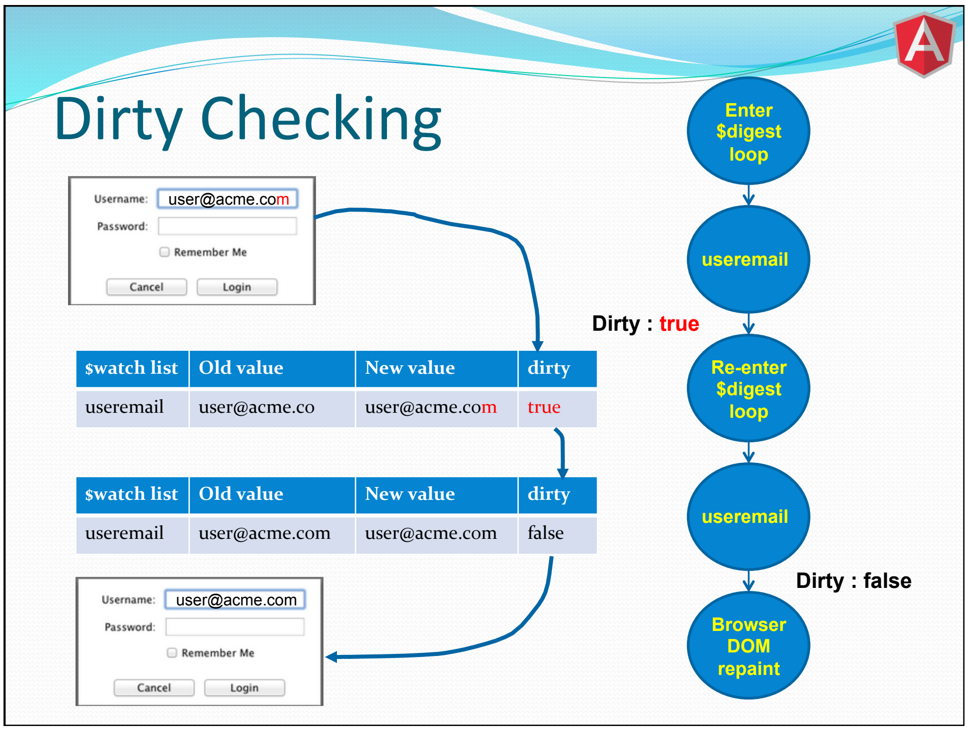
`$watch` list

- When an input element is bound to the **`$scope`** object, the user may change the value.
- If this data is bound to the view, in order to update the view Angular needs to track this change.
- This is done by adding a watch function to the **`$watch list`**.
- Properties that are on the **`$scope`** object are only bound if they are used in a view.
- For all UI elements that are bound to a **`$scope`** object, a **`$watch`** is added to the **`$watch list`**.
- These **`$watch lists`** are resolved in the **`$digest`** loop through a process called **`dirty checking`**.



Dirty Checking

- *Dirty checking* is a process that checks whether a value has changed that has not been synchronized across the app.
- Angular walks down the **`$watch list`**
  - If the updated value has not changed from the old value, it continues down the **`list`**.
  - If the value has changed, the app records the new value and continues down the **`list`**.



Dirty Checking

Username:	user@acme.com
Password:	

☐ Remember Me

Cancel Login

$watch list	Old value	New value	dirty
useremail	user@acme.co	user@acme.com	true

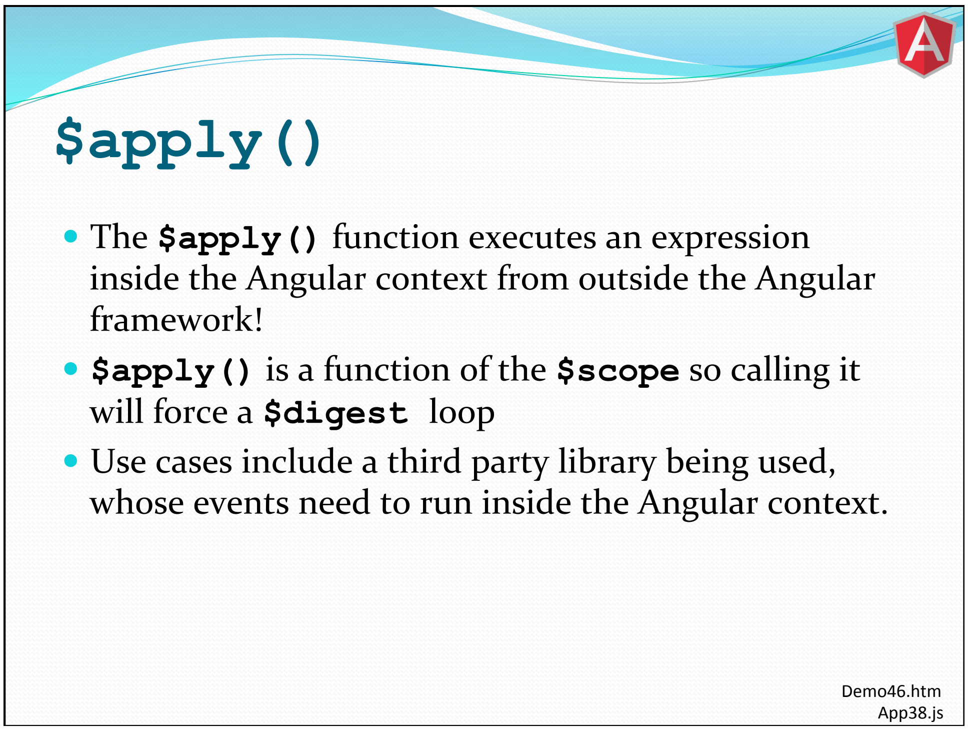
$watch list	Old value	New value	dirty
useremail	user@acme.com	user@acme.com	false

Username:	user@acme.com
Password:	

☐ Remember Me

Cancel Login

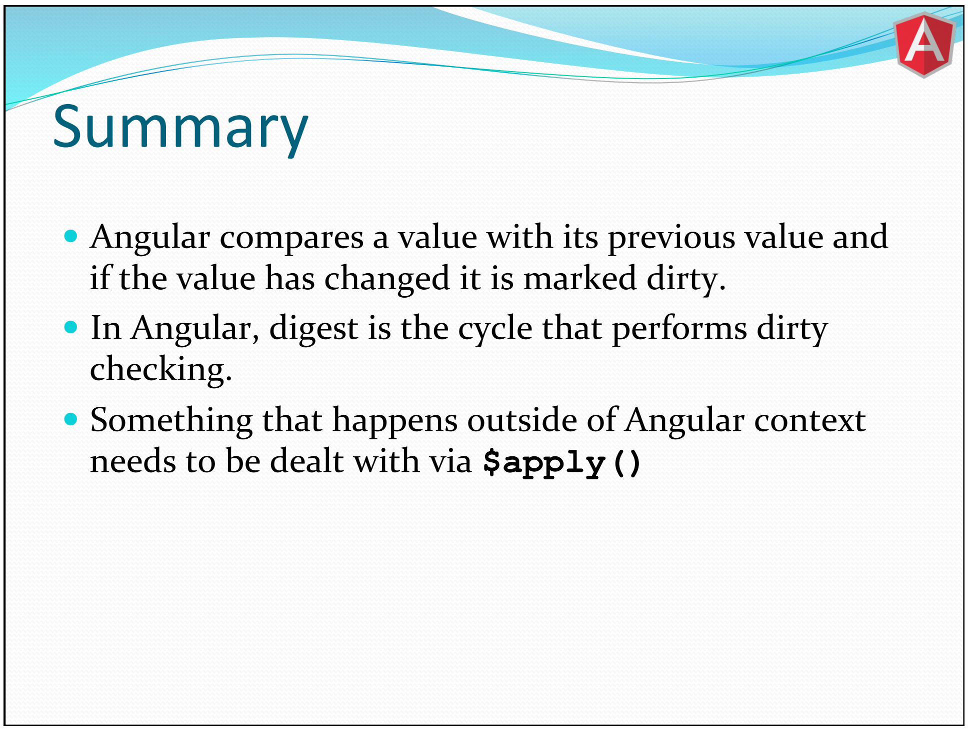
Compiled by Lakshman M N



`$apply()`

- The **`$apply()`** function executes an expression inside the Angular context from outside the Angular framework!
- **`$apply()`** is a function of the **`$scope`** so calling it will force a **`$digest`** loop
- Use cases include a third party library being used, whose events need to run inside the Angular context.

Demo46.htm
App38.js



Summary

- Angular compares a value with its previous value and if the value has changed it is marked dirty.
- In Angular, digest is the cycle that performs dirty checking.
- Something that happens outside of Angular context needs to be dealt with via **`$apply()`**

Extending Angular

AngularUI

- AngularJS is packed with a lot of features that can be used to build an app without additional libraries.
- There are however some libraries that can be included to maximize the power of the applications.
- The AngularUI library has been segregated into several modules.
  - This helps pick and choose components needed rather than including the entire suite.

ui-router

- A very useful component of the AngularUI library is the **`ui-router`**.
- It is a routing framework that embraces the state-machine nature of a routing system rather than a simple URL route.
- It allows defining of states and transitioning the application into those states.
- The **`ui-router`** needs to be installed and injected as a dependency into the application.
- It can be downloaded from https://github.com/angular-ui/ui-router

ui-router

- Unlike the built-in **`ngRoute`** service, the **`ui-router`** can nest views since it is based on states rather than a simple url.
- In place of the **`ng-view`** directive, the **`ui-view`** directive is employed.
- The **`config()`** method is used to define a route.
  - Instead of setting our routes on **`$routeProvider`**, the states are set on the **`$stateProvider`**

http://localhost/AngularJS/Demo47.htm
App39.js
Content.html

ui-utils

- The UI utils library is a powerful utility package that provides many custom extensions.
- The library is to be downloaded and installed

  http://angular-ui.github.io/ui-utils/
- Each component of the **ui-utils** library is built as a separate module.
- The required module needs to be included independently.

mask

- A lot of input fields require users to provide data in specific formats.
- A clean UI can be presented for users to provide appropriate input.
- The **ui-utils** library needs to be referenced and the **ui.mask** module is to be added as a dependency.
- The **ui-mask** accepts a single format string based on the following rules
  - **A** – any letter
  - **9** – any number
  - ***** - any alphanumeric character

Demo48.htm
App40.js

ui-event

- The **`ui-event`** module is very useful to handle events not natively supported by AngularJS.
- The **`ui-event`** module is a wrapper around native events.
  - This helps in responding to any event fired by the browser.

```
<img src="sunshine.jpg" ui-
  event="{dblclick:'showImg()'}“>
```

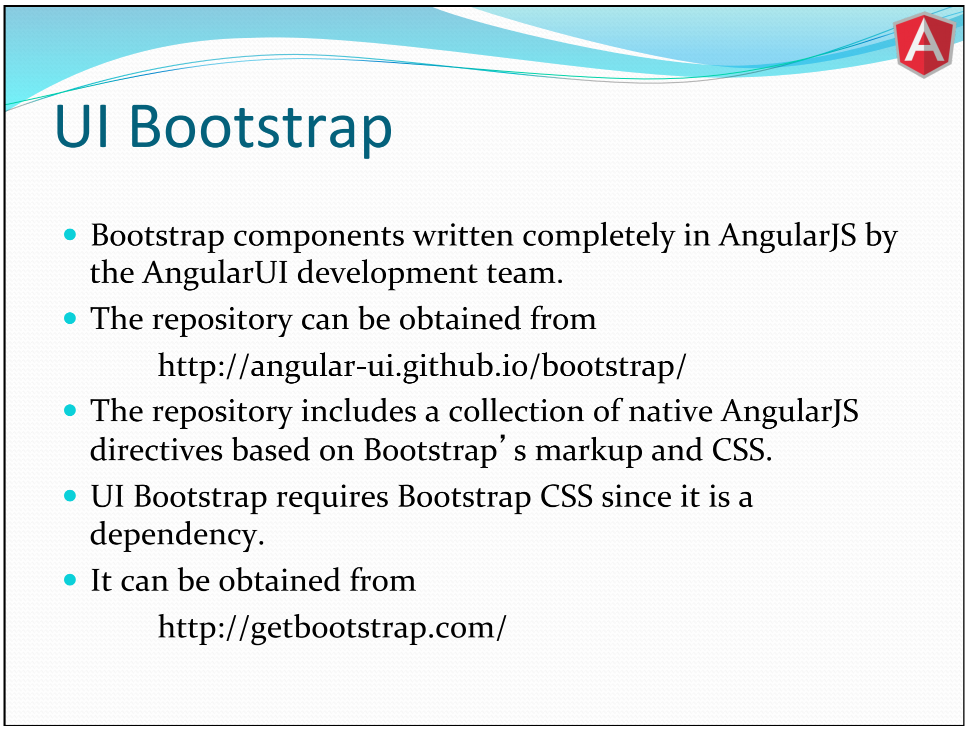
Demo49.htm
App41.js

ui-format

- The format library is a wrapper around different approaches to handle working with string tokens.
- The token replacement can be used with the format library either with an array or a key-value/javascript object.

```
{{ "Are you on the $0, $1, or $2 floor?"
  | format:tokens}}
```

Demo50.htm
App42.js

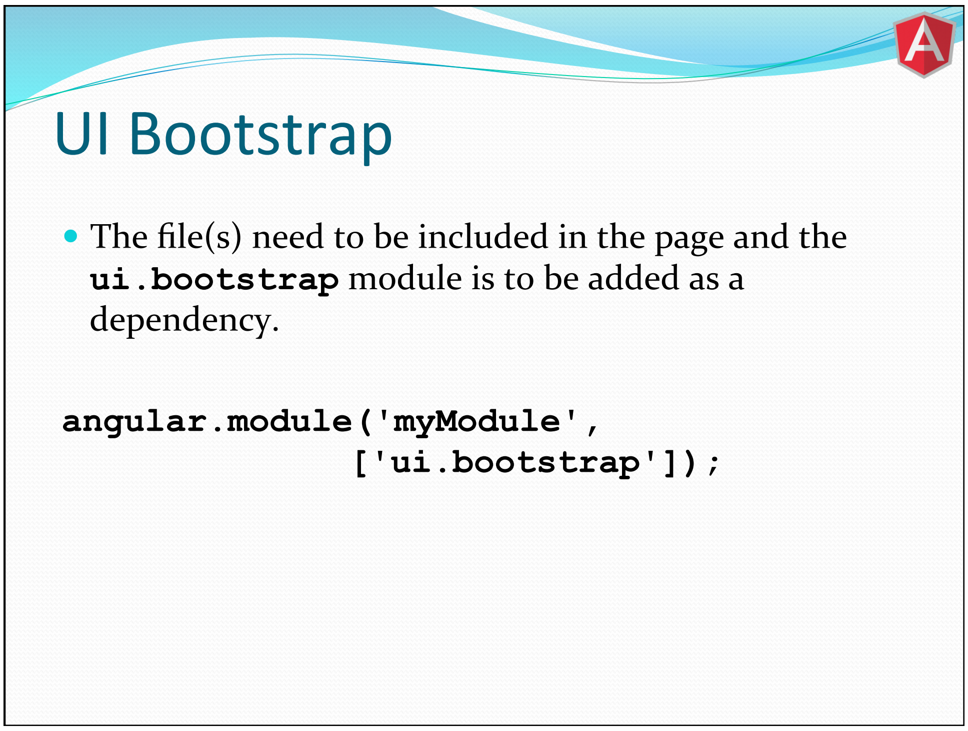


UI Bootstrap

- Bootstrap components written completely in AngularJS by the AngularUI development team.
- The repository can be obtained from

  http://angular-ui.github.io/bootstrap/
- The repository includes a collection of native AngularJS directives based on Bootstrap's markup and CSS.
- UI Bootstrap requires Bootstrap CSS since it is a dependency.
- It can be obtained from

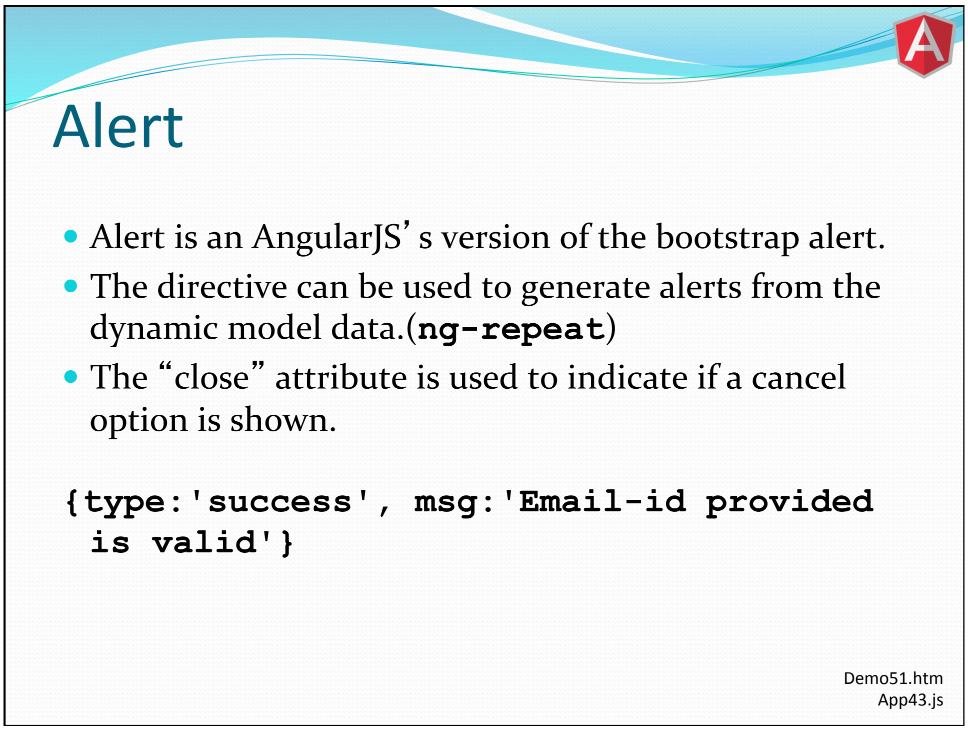
  http://getbootstrap.com/



UI Bootstrap

- The file(s) need to be included in the page and the **`ui.bootstrap`** module is to be added as a dependency.

```
angular.module('myModule',
               ['ui.bootstrap']);
```

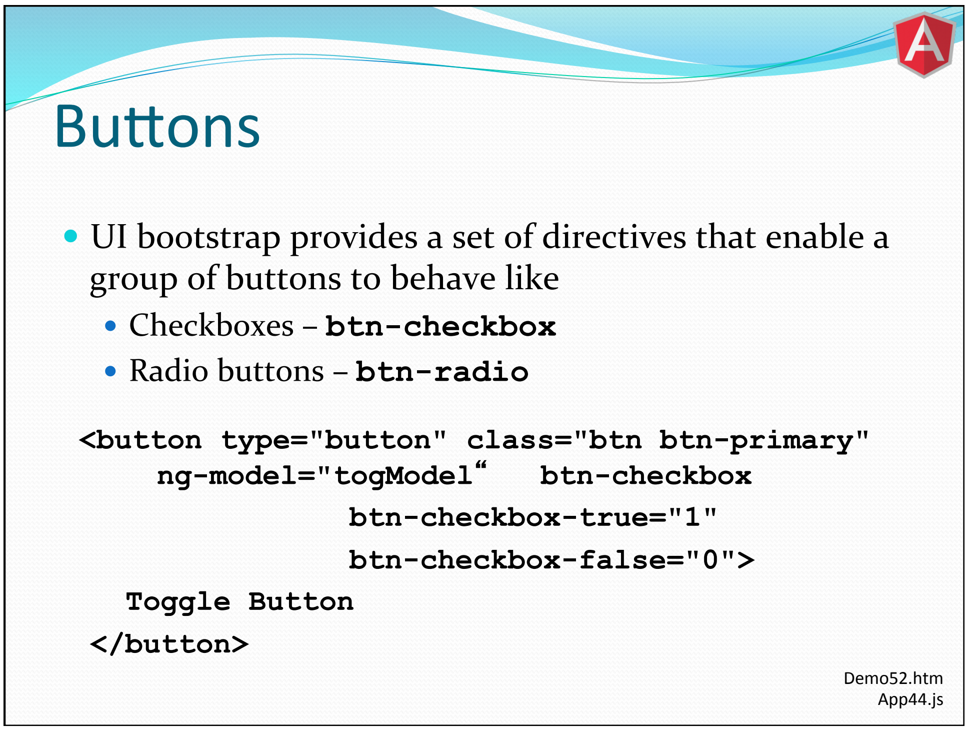


Alert

- Alert is an AngularJS' s version of the bootstrap alert.
- The directive can be used to generate alerts from the dynamic model data.(**`ng-repeat`**)
- The "close" attribute is used to indicate if a cancel option is shown.

```
{type:'success', msg:'Email-id provided
  is valid'}
```

Demo51.htm
App43.js



Buttons

- UI bootstrap provides a set of directives that enable a group of buttons to behave like
  - Checkboxes – **`btn-checkbox`**
  - Radio buttons – **`btn-radio`**

```
<button type="button" class="btn btn-primary"
     ng-model="togModel"   btn-checkbox
                  btn-checkbox-true="1"
                  btn-checkbox-false="0">
   Toggle Button
</button>
```

Demo52.htm
App44.js

Carousel

- Carousel creates a carousel that is on the lines of bootstrap’ s image carousel.
- A **`<carousel>`** element is employed with a **`<slide>`** element in it.
- The slides are cycled through at a specified interval.

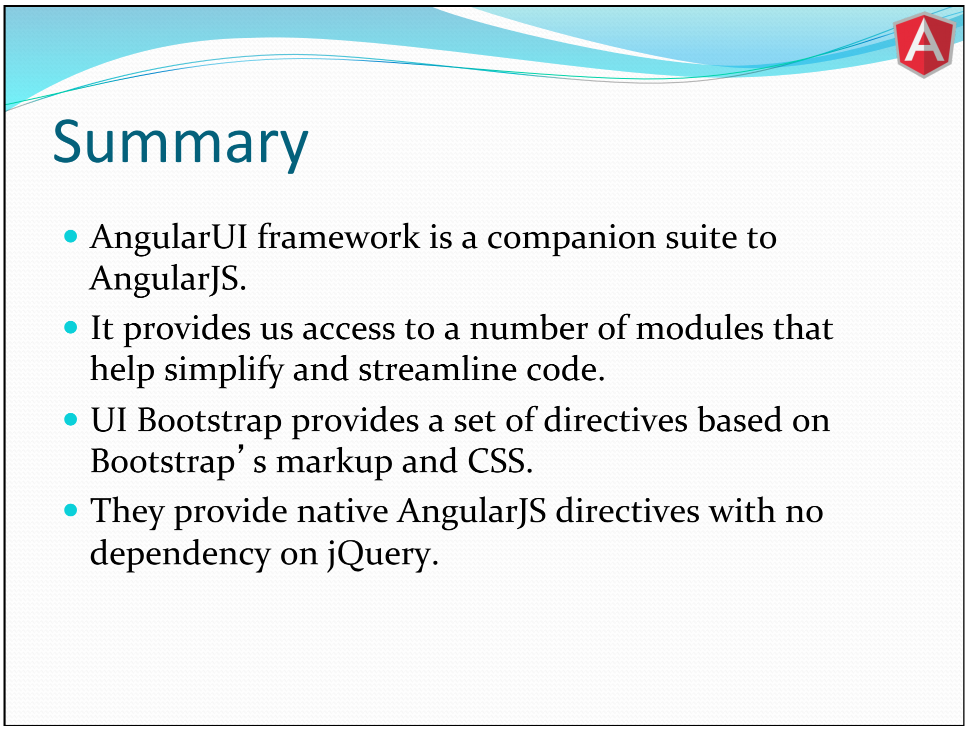
Demo53.htm
App45.js

Progressbar

- A directive that intends to provide visible feedback on the progress of a task.

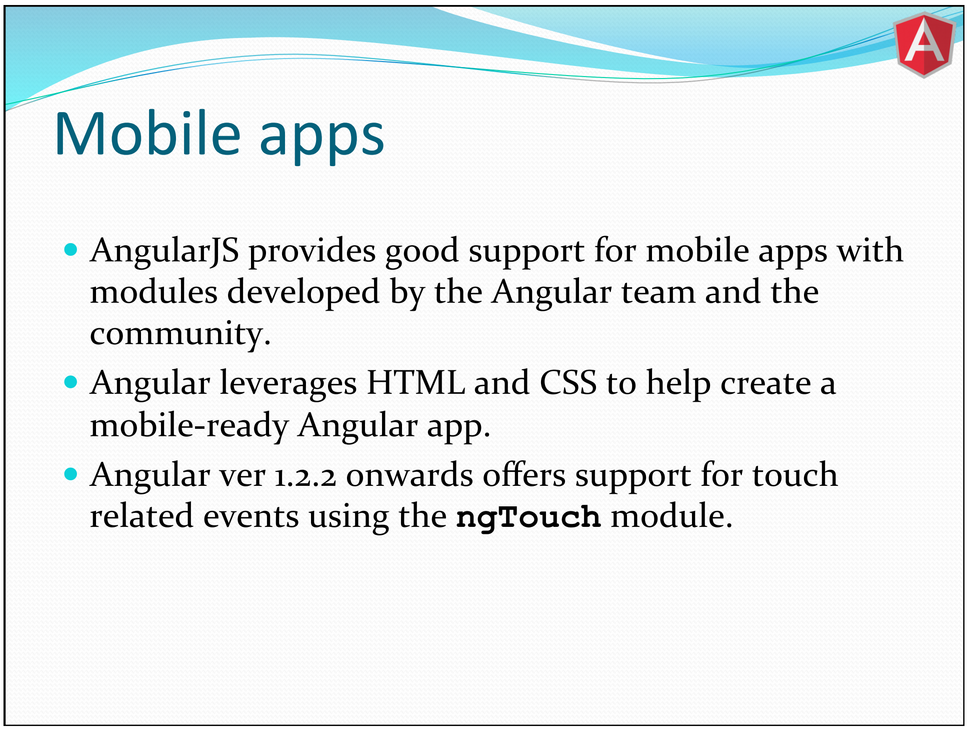
```
<progressbar value=“40”></progressbar>

<progressbar class=“progress-striped”
     value=“35” type=“success”>35%
</progressbar>
```

Demo54.htm
App46.js



Summary

- AngularUI framework is a companion suite to AngularJS.
- It provides us access to a number of modules that help simplify and streamline code.
- UI Bootstrap provides a set of directives based on Bootstrap' s markup and CSS.
- They provide native AngularJS directives with no dependency on jQuery.



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.

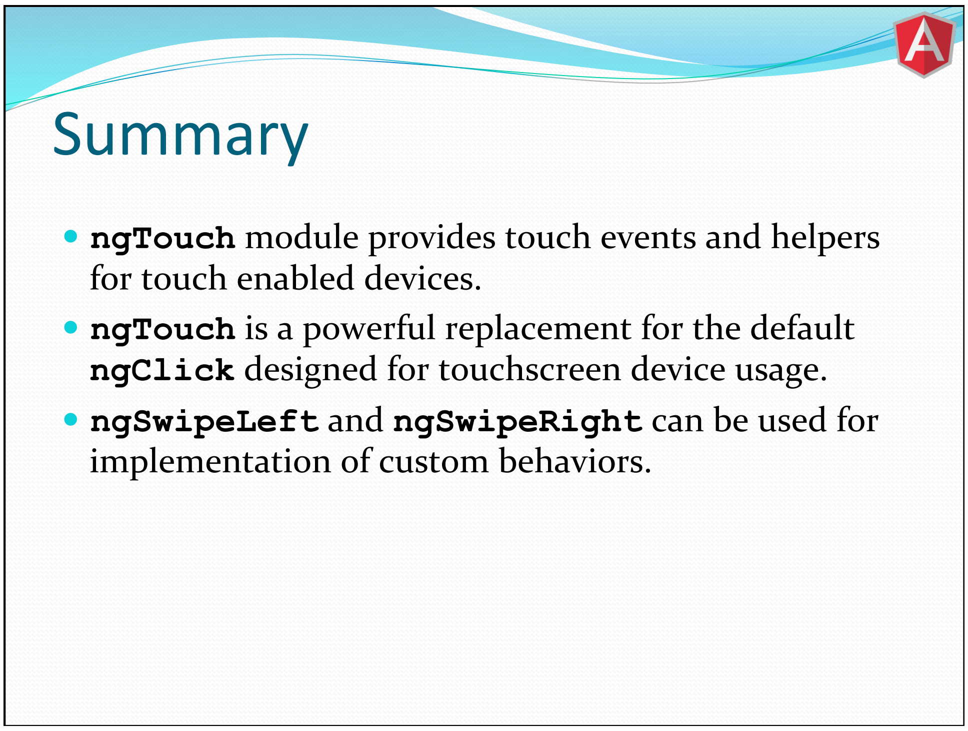
ngTouch

- **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.

swipe directives

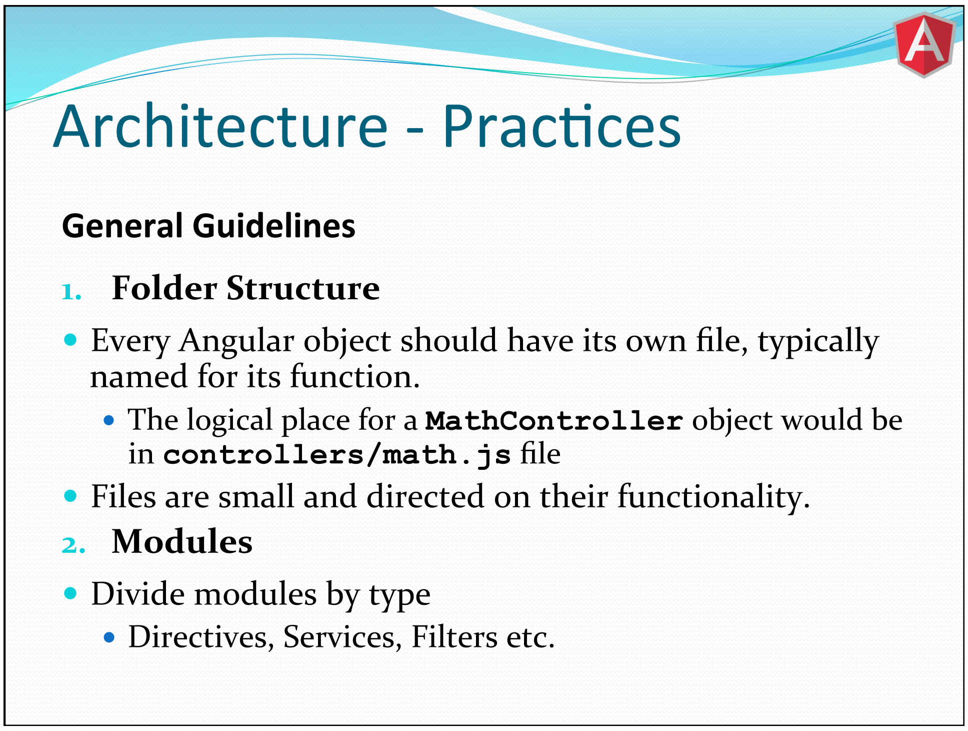
- **ngTouch** introduces two new directives called swipe directives.
- These directives are used to capture user swipes, either left or right across the screen.
- These directives help in introducing contextual features in applications.
- **ngSwipeLeft** directive is used to detect a swipe from right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

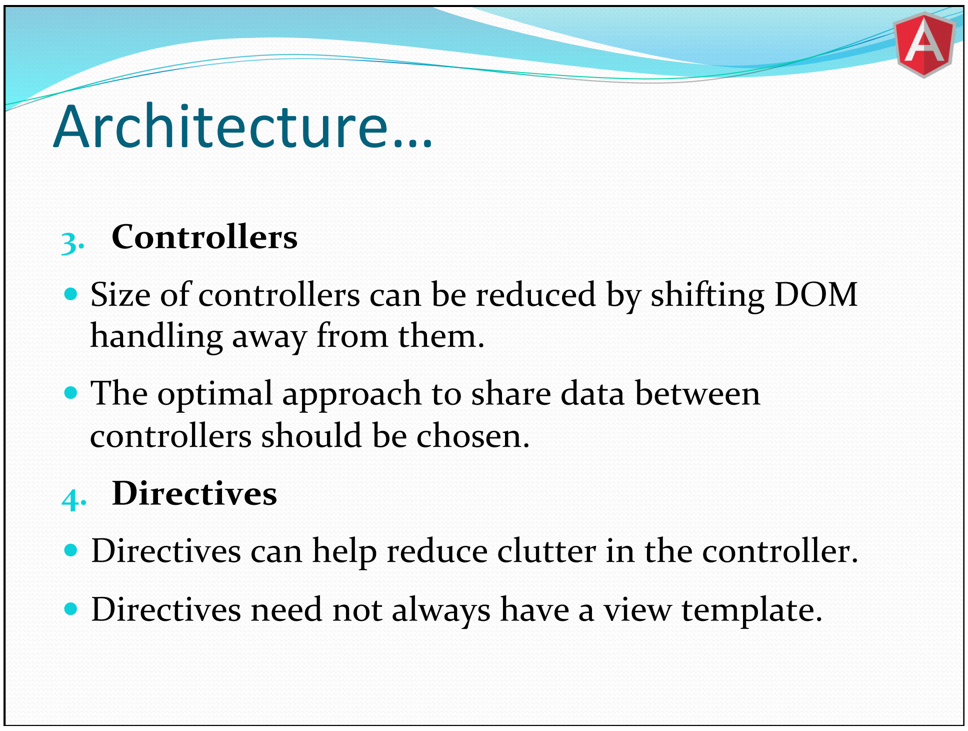
General Guidelines

1. **Folder Structure**

- Every Angular object should have its own file, typically named for its function.
  - The logical place for a **`MathController`** object would be in **`controllers/math.js`** file
- Files are small and directed on their functionality.

2. **Modules**

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



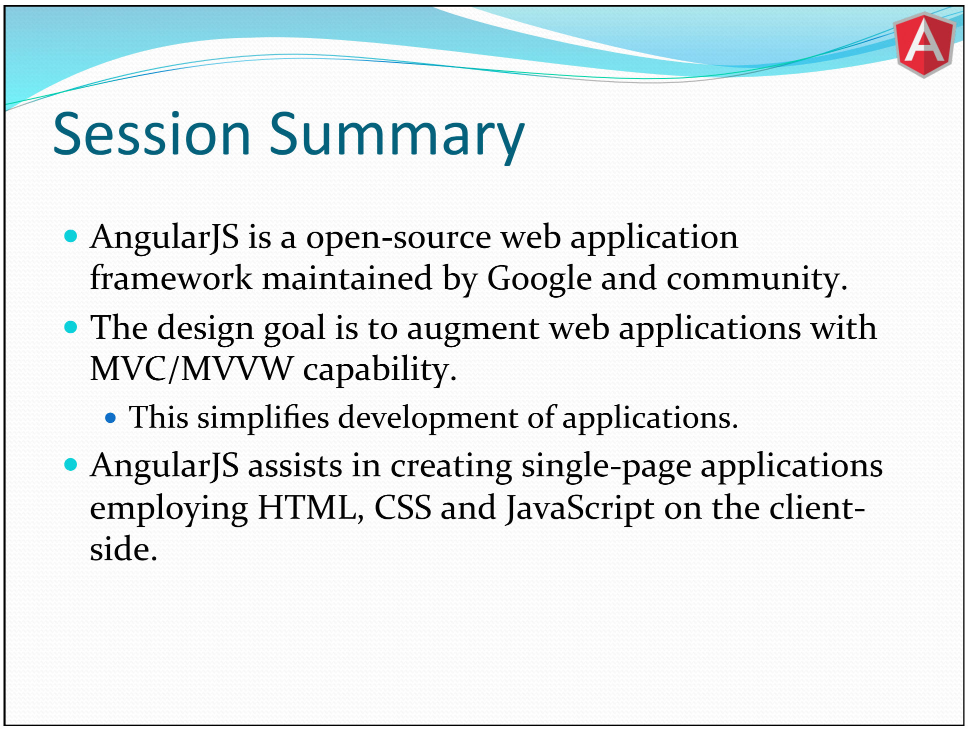
Architecture…

3. **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.

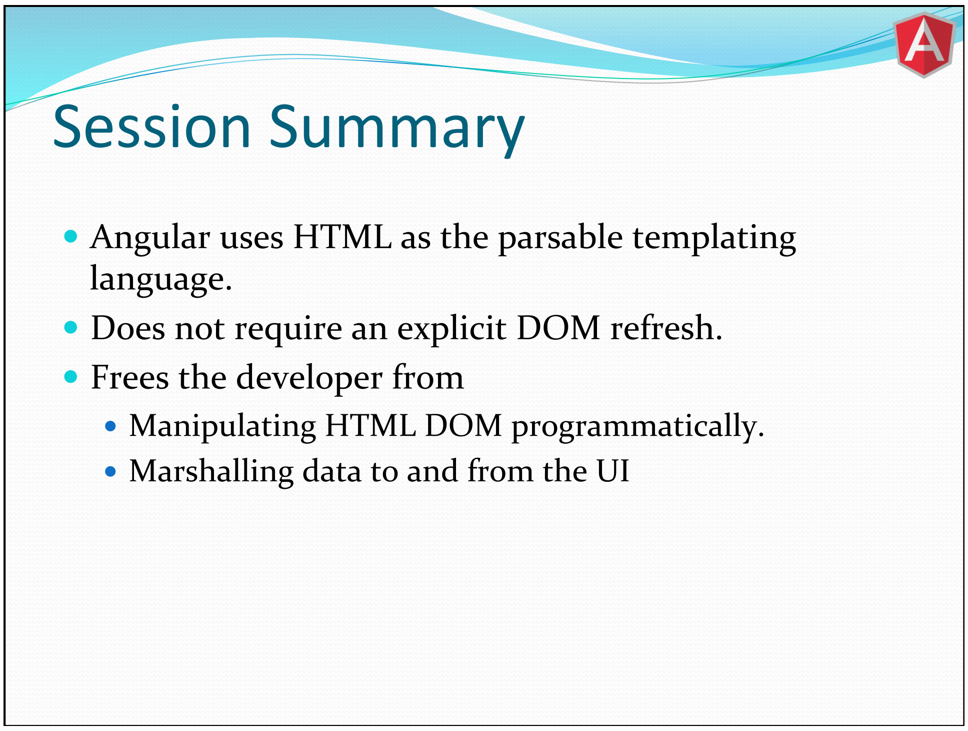
4. **Directives**

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



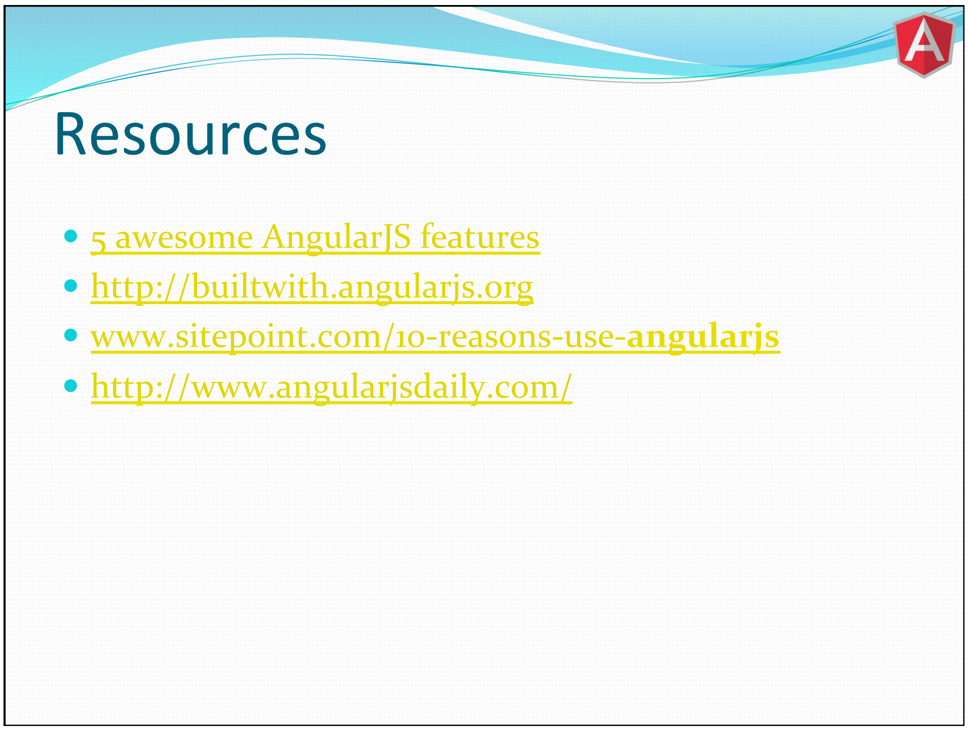
Session Summary

- 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.



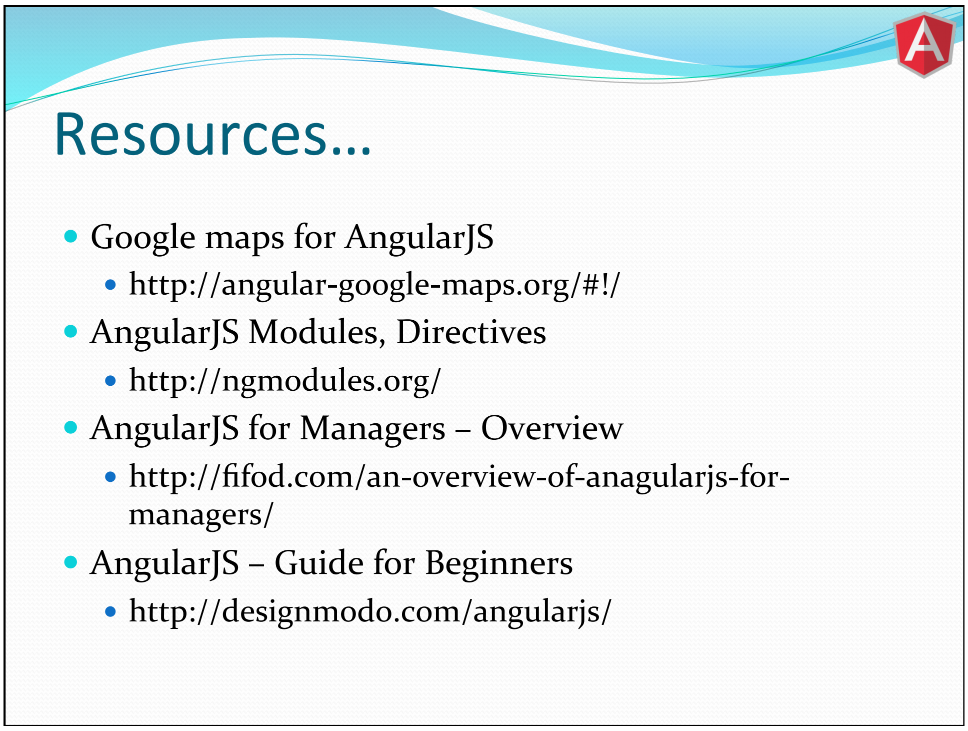
Session Summary

- 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



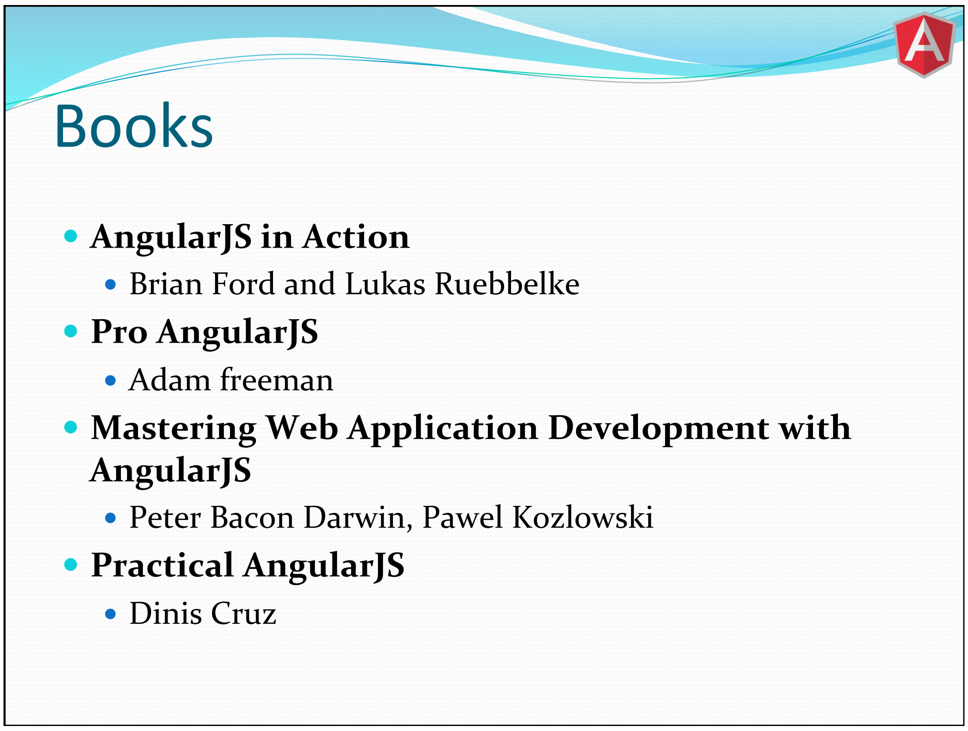
Resources

- 5 awesome AngularJS features
- http://builtwith.angularjs.org
- www.sitepoint.com/10-reasons-use-**angularjs**
- http://www.angularjsdaily.com/



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