COMP 3322
Modern Technologies on World Wide Web

2nd semester 2017-2018

AngularJS (O2)

Dr. C Wu

Department of Computer Science
The University of Hong Kong

More about Express.js

- Working with client-side frameworks
  - AngularJS: a client-side model–view–controller (MVC) framework
  - Bootstrap: an HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites (http://www.w3schools.com/bootstrap/)
  - etc.

The MEAN software stack: MongoDB, Express.js, AngularJS, and Node.js

Overview

- AngularJS (or Angular, or Angular.js) is a JavaScript-based open-source front-end web application framework
  - mainly maintained by Google, started 2009.
  - used on the websites of NBC, Intel, Sprint, ABC News, etc.
  - designed for developing single-page applications

- Single-page application (SPA) is a web application that fits on a single web page
  - the page does not completely reload or transfer to another page

    to provide a user experience similar to that of a desktop application
  - all HTML, JavaScript, CSS codes are retrieved with a single page load, or dynamically loaded in response to user actions (e.g., through AJAX requests)
  - examples at http://www.awwwards.com/websites/single-page/

MVC

- AngularJS implements the MVC pattern to separate presentation, data, and logic components
- Model-View-Controller (MVC) is a software design pattern
  - *model* captures behaviour of an application in terms of data and logic
  - *view* can be any output representation of data
  - *controller* controls model and view
- In AngularJS, the HTML page is the *view*, which is embedded with AngularJS custom tag attributes (directives). AngularJS binds input or output data to a *model,* represented by standard JavaScript variables. The values of those JavaScript variables can be manipulated by JavaScript function (*controller*).

An example AngularJS application

AngularJS is a JavaScript framework, added to an HTML page in <script></script>

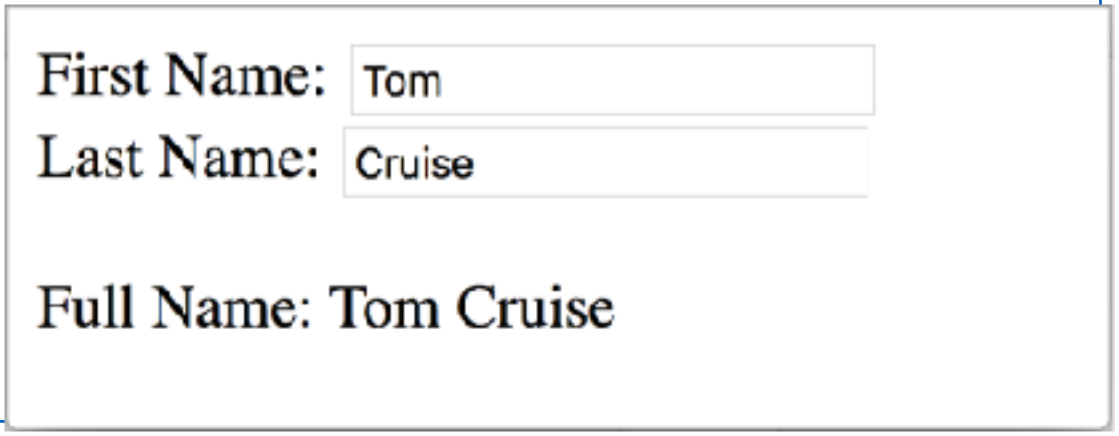
```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
</body>
</html>
```

directive

AngularJS extends HTML attributes with **directives**, and binds data to HTML with **expressions**

expression

an AngularJS module defines an AngularJS application



First Name: Tom
Last Name: Cruise

Full Name: Tom Cruise

Directives

- **ng-app** directive defines an AngularJS application. In this example, the <div> element is where the AngularJS application will run

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
</body>
</html>
```

Directives

- **ng-controller** directive defines the controller, which handles application variables

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
</body>
</html>
```

Directives

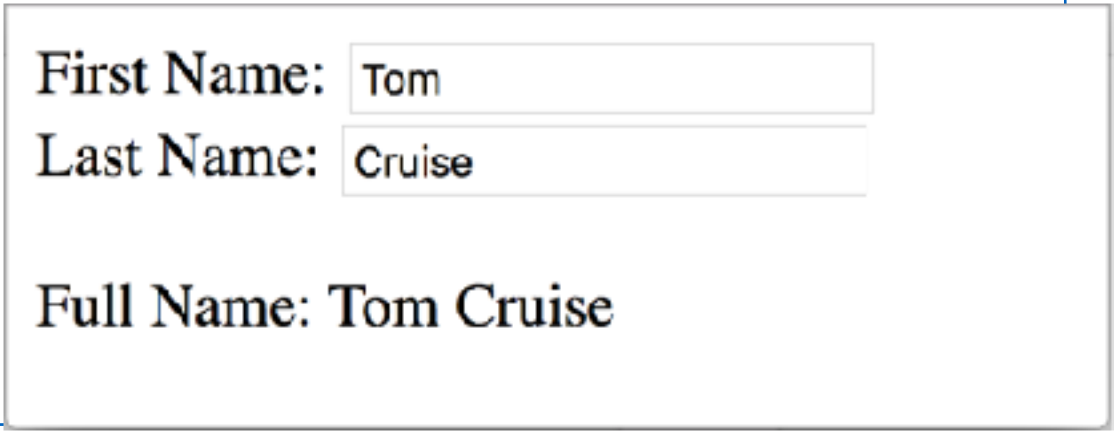
- **ng-model** directive binds the value of HTML elements (input, select, textarea) to application data (variable). In this example, two ng-model directives bind values of the two input textboxes to variable **firstName** and variable **lastName**, respectively.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name:  <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
</body>
</html>
```

Directives

- **ng-bind** directive binds application data (variable, expression) to the content (innerHTML) of an HTML element. If the value of the given variable or expression changes, the content of the specified HTML element changes accordingly.

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
<p> Full Name: <span ng-bind="firstName"></span> <span ng-bind="lastName"></span> </p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
</body>
</html>
```



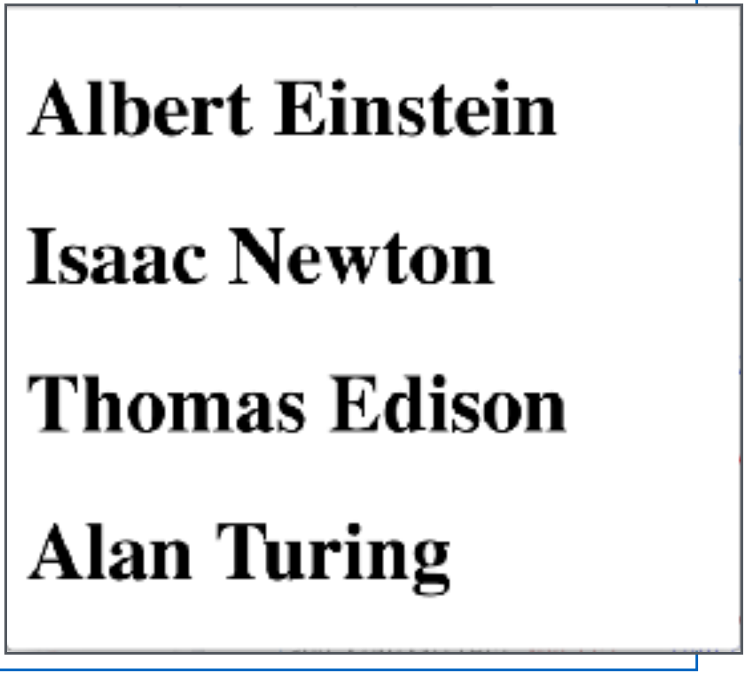
First Name: Tom
Last Name: Cruise

Full Name: Tom Cruise

Directives

- **ng-repeat** directive repeats an HTML element for each item in a collection (an array or an object)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>

<body ng-app="myApp" ng-controller="myCtrl">

<h1 ng-repeat="x in records">{{x}}</h1>

<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.records = [
        "Albert Einstein",
        "Isaac Newton",
        "Thomas Edison",
        "Alan Turing",
    ]
});
</script>

</body>
</html>
```

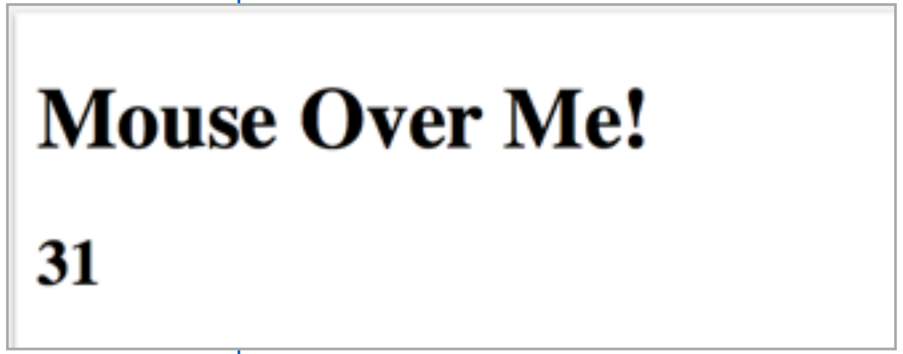


```
Albert Einstein

Isaac Newton

Thomas Edison

Alan Turing
```

Directives

- Events directives: ng-blur, ng-change, ng-click, etc.
  - AngularJS uses its own HTML events directives to add AngularJS event listeners to HTML elements, allowing to run AngularJS functions at these events

```
<div ng-app="myApp" ng-controller="myCtrl">

<h1 ng-mousemove="count = count + 1">Mouse over me!</h1>

<h2>{{ count }}</h2>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>
```



Mouse Over Me!

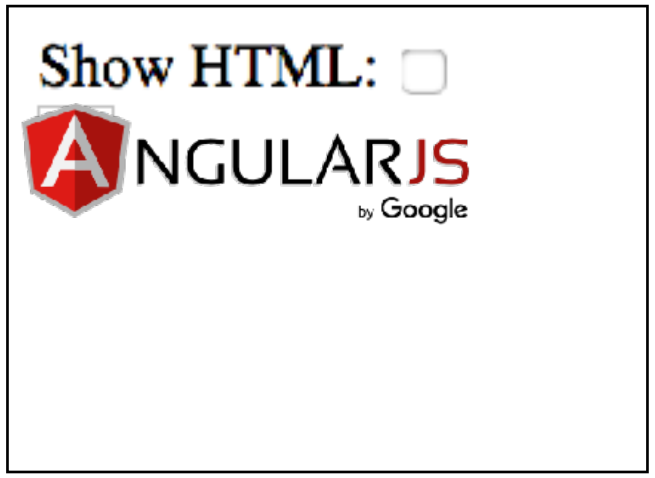
31

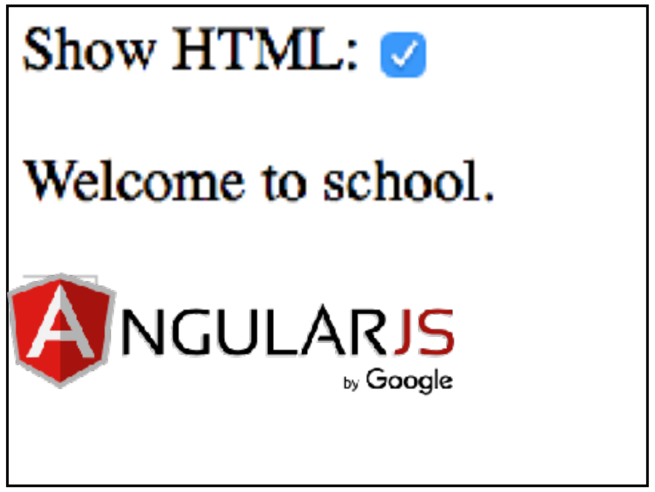
Directives

- Directives for binding application data to the attributes of HTML DOM elements: ng-show, ng-hide, ng-src, etc.

```
<body ng-app="">

Show HTML: <input type="checkbox" ng-model="checkedOrNot">

<div ng-show="checkedOrNot">
<p>Welcome to school.</p>
</div>

<div ng-init="myPic = 'pic.jpg'">
<img ng-src="{{myPic}}">
</div>

</body>
```

evaluate an expression in the current scope


Show HTML: ☐

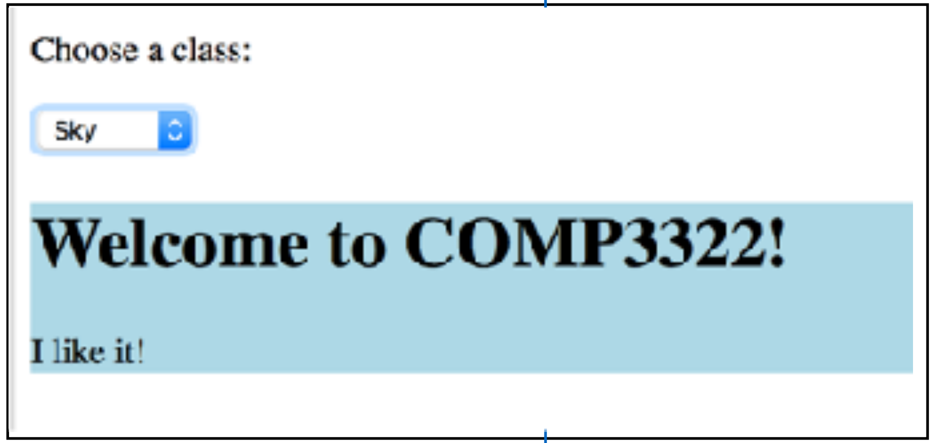

Show HTML: ✓

Welcome to school.

Directives

- ng-class for dynamically binding one or more CSS classes to an HTML element

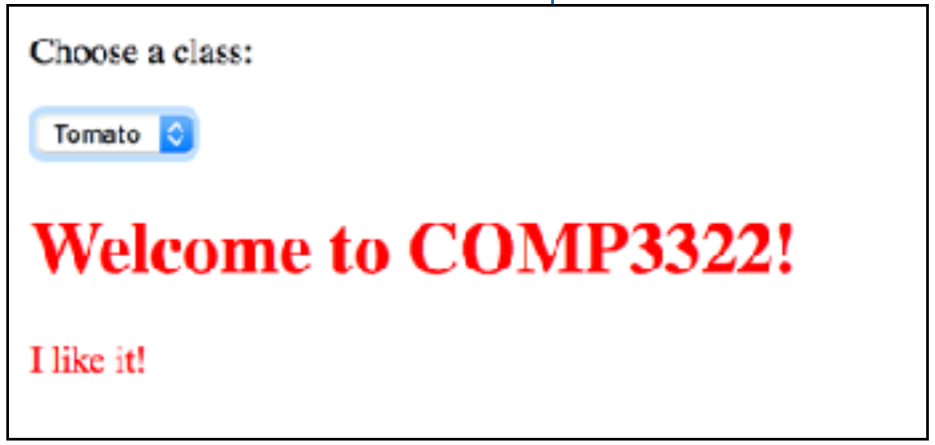
```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<style>
.sky {
    background-color:lightblue;
}
.tomato {
    color:red;
}
</style>
<body ng-app="">

<p>Choose a class:</p>

<select ng-model="home">
<option value="sky">Sky</option>
<option value="tomato">Tomato</option>
</select>

<div ng-class="home">
  <h1>Welcome to COMP3322!</h1>
  <p>I like it!</p>
</div>

</body>
</html>
```


Choose a class:

Sky

Welcome to COMP3322!

I like it!


Choose a class:

Tomato

Welcome to COMP3322!

I like it!

See examples and more AngularJS directives at
http://www.w3schools.com/angular/angular_ref_directives.asp

Expressions

- AngularJS **expressions** are much like JavaScript expressions
  - an expression can contain numbers, strings, objects, arrays, operators, and variables
- AngularJS expression can be written inside double braces **{{ expression }}**, or inside a directive such as **ng-bind="expression"**
  - AngularJS resolves the expression and writes result data where the expression is

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>
```

the {{ firstName + " " + lastName}} expression is bound with ng-model="firstName" and ng-model="lastName"

```
<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
```

Data binding

- **Data binding** in AngularJS is the synchronization between the model and the view
  - *model* of an AngularJS application is a collection of data available for the application
  - *view* is the HTML page where the AngularJS application is displayed

- Several ways of binding the model (data) and the view
  - use **ng-bind** directive, which binds specific data with the innerHTML of an HTML element

```
<p> Full Name: <span ng-bind="firstName"></span> <span ng-bind="lastName"></span></p>
```

  - use double braces **{{ }}** to display application data (bind expression with data)

```
Full Name: {{firstName + " " + lastName}}
```

  - use **ng-model** directive to bind application data (model) and the value of HTML elements (input, select, textarea) (view)

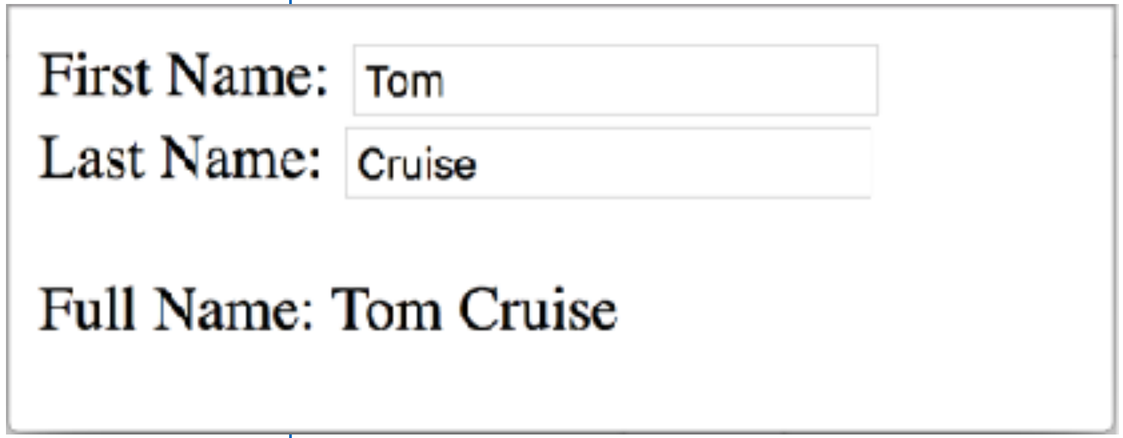
```
First Name: <input type="text" ng-model="firstName"><br>
```

bind “firstName” with value of the textbox

Two-way binding

- **ng-model** directive provides two-way binding between the model and the view
  - when data in the *model* changes, the *view* reflects the change; and when content in the *view* changes, the *model* is updated as well (this happens immediately and automatically, such that the model and the view are consistent at all times)

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
```



First Name: Tom
Last Name: Cruise

Full Name: Tom Cruise

AngularJS' two-way data binding is its most notable feature, largely relieving the server backend from web page templating responsibilities

AngularJS application

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js">
</script>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>


<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
</body>
</html>
```

an AngularJS **module** defines an AngularJS app; controllers, etc., can be added to this app

[ ] can be used to specified dependent modules

"myApp" corresponds to the HTML element in which the application will run

Controller

- AngularJS controllers control the data in an AngularJS application
  - **ng-controller** directive defines a controller in the HTML
  - in the script, a controller is a JavaScript object, created by a JavaScript object constructor

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
```

$scope is the object owning application variables and functions

add controller myCtrl's constructor function to the application using the .controller() method

Controller (cont'd)

- Due to immediate synchronization of the model and the view (data binding) in AngularJS, the controller can simply focus on controlling the model (data), but be completely separated from the view, while the view will reflect any changes made to the model in the controller

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
});
</script>
```

$scope is the object owning application variables and functions

add controller myCtrl's constructor function to the application using the .controller() method

Scope

- **$scope** is a JavaScript object with properties and methods (application variables and functions), which are available for both the view and the controller
  - When adding/changing properties and methods in **$scope** in the controller, the view (HTML) immediately reflects the changes; likewise, any value alteration in the view are reflected in the **$scope** object as well

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Tom";
    $scope.lastName= "Cruise";
   $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    };
});
</script>
```

Similar to variables, the **fullName()** method defined in the **$scope** object can be used (invoked) in the view

AngularJS MVC revisited

- An AngularJS application consists of
  - *view*, which is the HTML page
  - *model*, which is the data available for the application (exposed in $scope)
  - *controller*, which is the JavaScript function that creates/changes/removes/controls the data

Services

- An AngularJS **service** is a function or an object, that provides specific functionalities in an AngularJS application
- AngularJS has about 30 built-in services (e.g., $http, $location, $interval) or developers can build their own services

an example self-built service:

```
<div ng-app="myApp" ng-controller="myCtrl">

<p>The hexadecimal value of 255 is: {{hex}} </p>

</div>

<script>
var app = angular.module('myApp', []);

app.service('hexafy', function() {
    this.myFunc = function (x) {
        return x.toString(16);
    }
});

app.controller('myCtrl', function($scope, hexafy)
{
    $scope.hex = hexafy.myFunc(255);
});
</script>
```

HTTP service

- The **$http** service is a core AngularJS service that facilitates communication with the remote HTTP server via the browser's XMLHttpRequest object
  - the service sends a request to the server, and lets the application handle the response

```
<div ng-app="myApp" ng-controller="myCtrl">

<p>Welcome message: {{myWelcome}}</p>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http({
        method : "GET",
        url : "welcome.html"
    }).then(function(response) {
        $scope.myWelcome = response.data;
    }, function(response) {
        $scope.myWelcome = response.statusText;
    });
});
</script>
```

a function that takes a configuration object as argument

success callback

error callback

response is an object with these properties:

.config: the configuration object used to generate the request
.data: a string or an object representing the response body
.headers: a function used to get header value
.status: status code of the response
.statusText: status phrase of the response

See more at https://docs.angularjs.org/api/ng/service/$http

HTTP service (cont'd)

Use shortcut method:

```
<div ng-app="myApp" ng-controller="myCtrl">

<p>Welcome message: {{myWelcome}}</p>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
    $http.get("welcome.html")
    .then(function(response) {
        $scope.myWelcome = response.data;
     }, function(response) {
        $scope.myWelcome = response.statusText;
    });
});
</script>
```

Shortcut methods in **$http**

- .delete()
- .get()
- .head()
- .jsonp()
- .patch()
- .post()
- .put()

if using .post():

```
$http.post("welcome.html", {key1:value1, key2:value2})
```

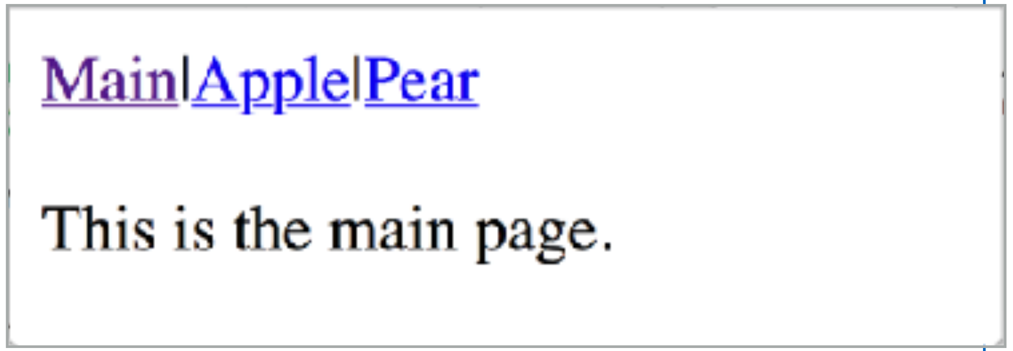
See more at https://docs.angularjs.org/api/ng/service/$http

Routing

- The **ngRoute** module routes an application to different pages without reloading the entire page, to implement a single page application

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-route.js"></script>
<body ng-app="myApp">

<a href="#/!">Main</a>|<a href="#!apple">Apple</a>|
<a href="#!pear">Pear</a><br>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.html"
    })
    .when("/apple", {
        templateUrl : "apple.html"
    })
    .when("/pear", {
        templateUrl : "pear.html"
    });
});
</script>
```

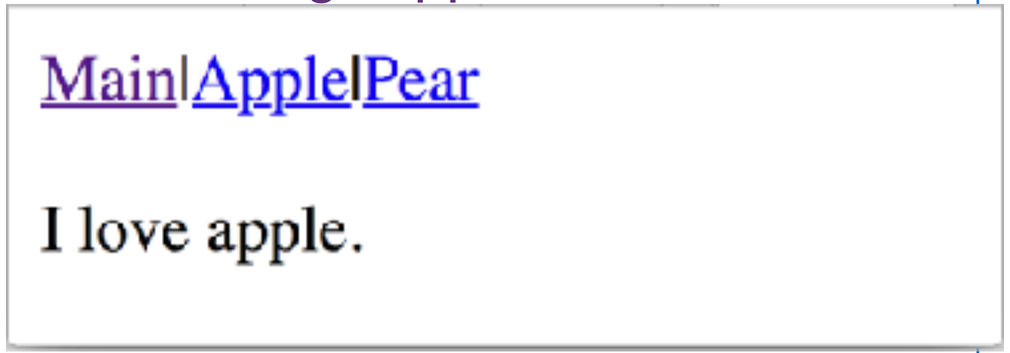
After initial page load



Main|Apple|Pear

This is the main page.

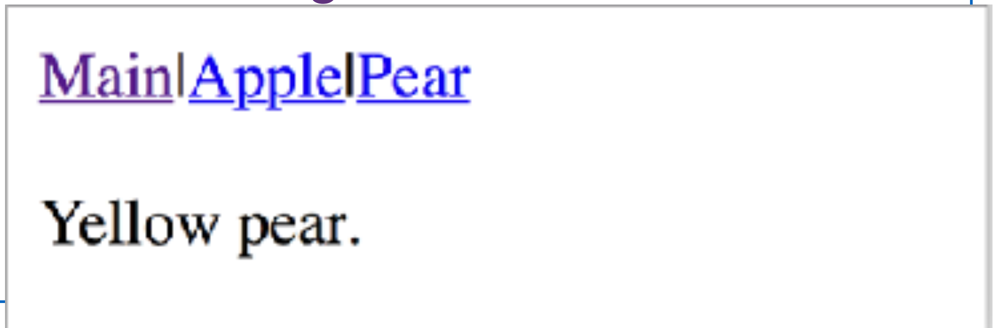
After clicking “Apple”



Main|Apple|Pear

I love apple.

After clicking “Pear”



Main|Apple|Pear

Yellow pear.

Routing

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-route.js"></script>

<body ng-app="myApp">

<a href="#/!">Main</a>|
<a href="#!apple">Apple</a>|
<a href="#!pear">Pear</a>
<br>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.html"
    })
    .when("/apple", {
        templateUrl : "apple.html"
    })
    .when("/pear", {
        templateUrl : "pear.html"
    });
});
</script>
```

include the AngularJS Route module

use # because we don't want the browser to actually go to main.html or apple.html or pear.html

ng-view directive specifies a container to put the content provided by the routing.

There are other ways to include the directive: `<div class="ng-view"></div>`

or `<ng-view></ng-view>`

Each application can only include one **ng-view** directive

Routing

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular-route.js"></script>

<body ng-app="myApp">

<a href="#/!">Main</a>|
<a href="#!apple">Apple</a>|
<a href="#!pear">Pear</a>
<br>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.html"
    })
    .when("/apple", {
        templateUrl : "apple.html"
    })
    .when("/pear", {
        templateUrl : "pear.html"
    });
});
</script>
```

add **ngRoute** as a dependent module in the application; then the application has access to the route module, which provides the **$routeProvider** object

configure different routes in **config** method of the application

Routing

Instead of referring to an HTML page using templateUrl, use the template property to write HTML content directly in the container

```
<body ng-app="myApp">
<a href="#/!">Main</a>|<a href="#!apple">Apple</a>|<a href="#!pear">Pear</a>|
<a href="#!others">Others</a><br>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        template : "<p>This is the main page.</p>"
    })
    .when("/apple", {
        template : "<p>I love apple.</p>"
    })
    .when("/pear", {
        template : "<p>Yellow pear.</p>"
    })
    .otherwise({
        template : "<p>Nothing has been selected.</p>"
    });
});
</script>
</body>
```

the otherwise method specifies the default route when none of the others is matched

MEAN

- AngularJS is the front-end part of the **MEAN** stack, which consists of **M**ongoDB database, **E**xpress.js web application server framework, **A**ngular.js, and **N**ode.js runtime environment.
  - a free and open-source JavaScript software stack for building dynamic web sites and web applications

We will use the MEAN stack to implement a web service in Lab 7, using AngularJS as the web front-end

References

- http://www.w3schools.com/angular/
- https://angularjs.org
- http://mean.io, https://meanjs.org