AngularJS:

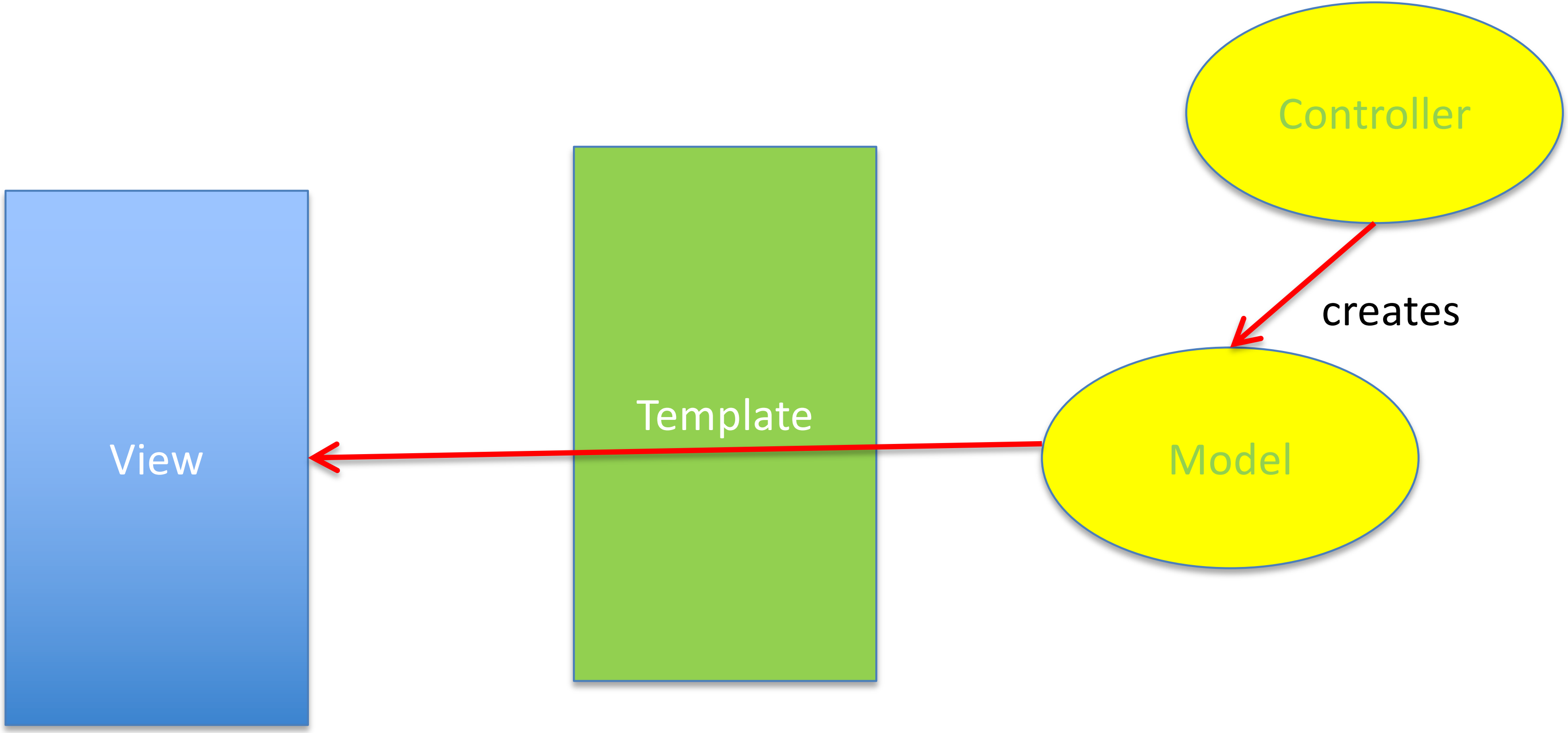
MODULES,
VIEWS, CONTROLLERS,
TEMPLATE, SCOPE

Model – View – Controller

The **view** is a projection of the **model** through the HTML **template**.

Whenever the model changes, AngularJS refreshes the appropriate binding points, which updates the view.

The model is generated by a **controller**.



Controllers

- AngularJS apps are controlled by controllers
- **Controllers** provide the **logic** behind your app.

- Use **ng--controller** to define the controller

- Controller **is a JavaScript Object (function), created by**
  standard **JS object constructor**
    - It contains data
    - It specifies the behavior
    - It should contain only the business logic needed for a single view.

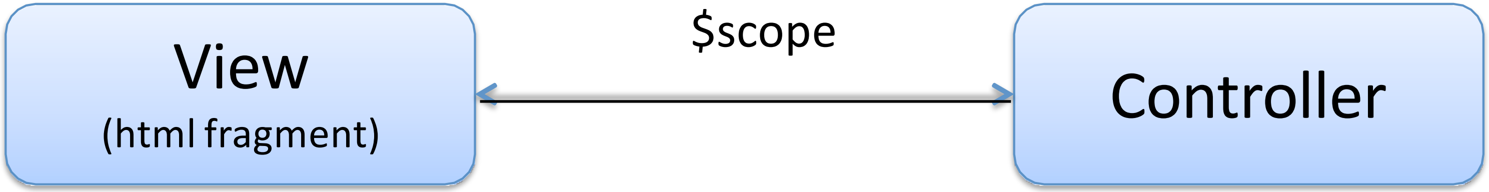
When to use Controllers

- Use controllers
  - set up the initial state of $scope object
  - add behavior to the $scope object
- Do not
  - Manipulate DOM (use **databinding, directives**)
  - Format input (use **form controls**)
  - Filter output (use **filters**)
  - Share code or state (use **services**)

Templates

- A Template is an HTML file, containing placeholders to be filled with the data.
- Such placeholders can be **directives** or **expressions**
- Once a controller fills the templates with the model (data), the **view** is generated

View, Controller and Scope



View

(html fragment)

$scope

Controller

$scope is an object that can *be used to communicate* between View and Controller

Modules

A module is a (reusable) component containing various elements: template and controller.

- If you have a lot of controllers, you are **polluting JS namespace**
- Modules can be loaded in any order
- We can build our **own filters** and **directives**!

Template for Controllers in Modules

```
// Create new module 'myApp' using angular.modul method.
// The module is not dependent on any other module
var myModule = angular.module('myModule',[]);

myModule.controller('MyCtrl', function ($scope) {
   // Your controller code here!
});
```

Example

```
<html ng-app="coursesApp">
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
  <script src="app.js">
</script>
</head>
<body ng-controller="CourseListController">
  <h1> Courses at UniTN </h1>
  <ul>
   <li ng-repeat="course in courses">
    <span>{{course.name}}</span>
    <p>{{course.teacher}}</p>
   </li>
  </ul>
</body>
</html>
```

Load the controller

Use the controller

Variable carried by the scope

Placeholders for the data

index.html

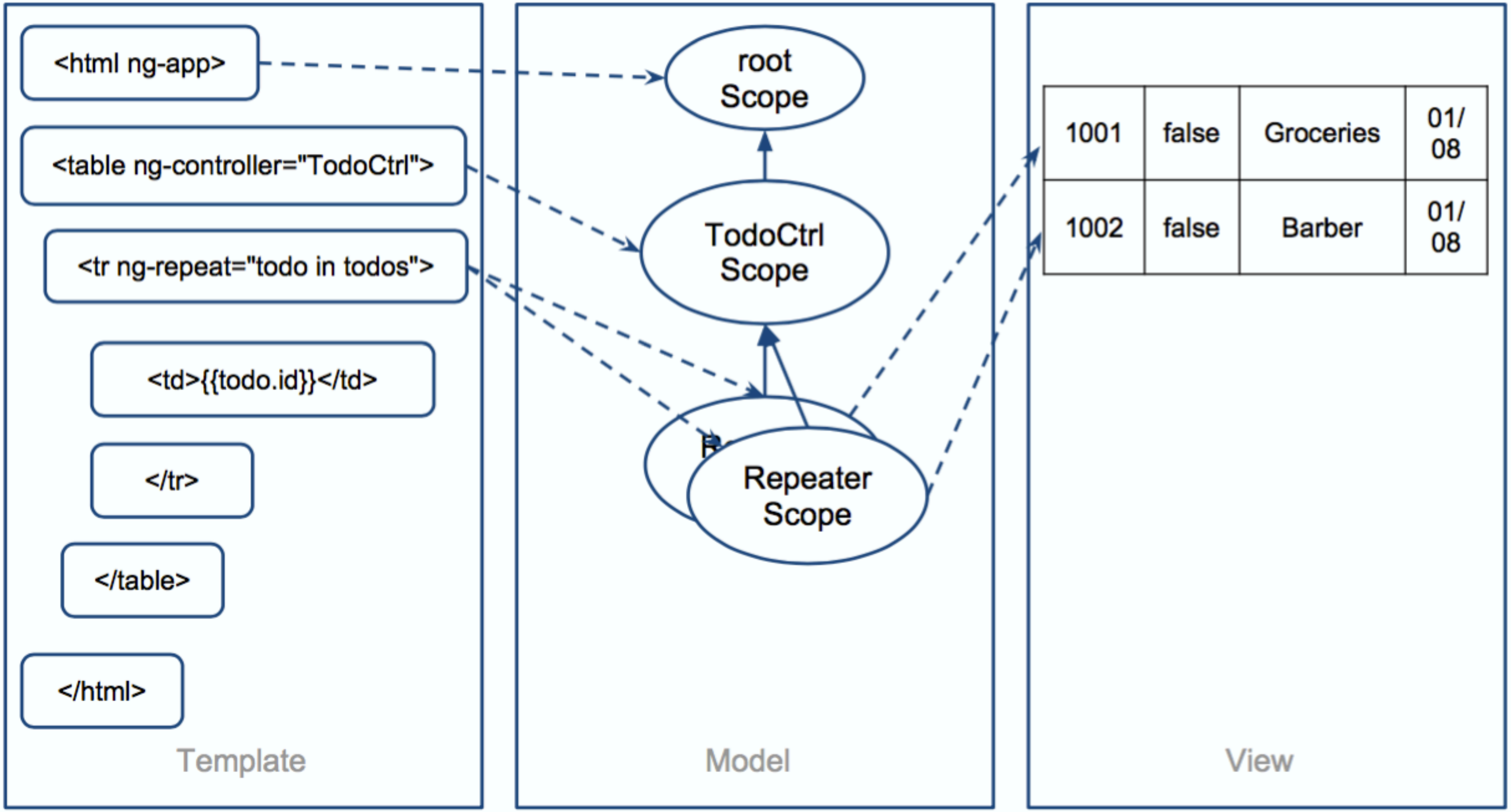
Example- part 2

```
/ /Define the `coursesApp` module
var coursesApp = angular.module('coursesApp', []);

/ /Define the controller
coursesApp.controller('CourseListController', function
    CourseListController($scope) {
        $scope.courses = [
{ name: 'Advanced computing architectures’, teacher: 'Roberto Passerone’},
{ name: 'Affective computing’,  teacher: 'Niculae Sebe’},
{  name: 'Web architectures’, teacher: 'Marco Ronchetti’ }
  ];
});
```

app.js

Scope



1001	false	Groceries	01/ 08
1002	false	Barber	01/ 08

Warning!

- Due to security reasons Chrome will not load local files by default.
- Launch the Google Chrome browser from the command line window with the additional argument '–allow-file-access-from-files'.
- Also, you might get "cross origin requests are only supported for HTTP"
- install a web server and access files on http://localhost

Warning!

- It is easy to make mistakes, difficult to detect them.
- Always keep the Javascript console open in the browser!

Other ways of referencing vars

in views you can bind an alias to your controller making it easy to reference $scope variables

```
<body ng-controller="ParentCtrl as ptr">
<input ng-model="name" /> {{ptr.name}}
<div ng-controller="ChildCtrl as chl">
<input ng-model="name" />
{{chl.name}} - {{ptr.name}}
</div>
```

This is useful e.g. when you nest controllers and you dont want to reference something from a different controller.

Default way to refer to current controller: $ctrl

phone in $ctrl.phones

Redefinable with the "Controller as" construct:

ng-controller="ParentCtrl as ptr"

$scope not used any more since AngularJS 1.5 !

Rearranging files to improve portability (One Feature per File)

Feature

module definition

app/

Component definition object

- course-list/
  - course-list.module.js
  - course-list.component.js
  - course-list.template.html
- app.module.js
- app.css
- index.html

Component template

Rearranging files to improve portability (One Feature per File)

course-list.module.js

```
// Define the ` courseList ` module
angular.module(courseList ', []);
```

phone-list.component.js

```
angular. module('phoneList’).
        component('phoneList', {
            templateUrl: 'phone-list/phone-list.template.html’,
            controller: ... });
```

app.module.js (Former pp.js)

```
// Define the ` 'phoneList App` module
angular.module(' 'phoneList App’, [
 // ...which depends on the ` courseList `
module ' courseList ' ]);
```

Adding filtering
and using 2-way data binding

```
<div class="q">
Search: <input ng-model="$ctrl.query" /> </div>

 <ul class=»list">
<li ng-repeat="course in $ctrl.courses | filter:$ctrl.query">
<span class="first-row">{{course.name}}</span>
<p class=»second-row">{{course.teacher}</p> </li>
</ul>
```

Angular JS services

Services

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application.

AngularJS has about 30 built-in services.

https://docs.angularjs.org/api/ng/service

Services

$anchorScroll
$animate
$animateCss
$cacheFactory
$compile
$controller
$document
$exceptionHandler
$filter
$locale
$location

$interpolate
$rootElement
$rootScope
$sce
$sceDelegate
$templateCache
$templateRequest

$interval
$log
$parse
$q
$timeout
$window

$timeout

Index.html (fragment only)

```
<div ng-app="myApp">
   <div ng-controller="myController">
      <div>5 seconds delay message : {{test1}}</div>
   </div>
</div>
```

Shows message after 5 seconds

app.js

```
var myAppModule = angular.module('myApp', []);
myAppModule.controller('myController', function($scope, $timeout){
    $timeout( function(){
                         $scope.test1 = "Hello World!";
                         }, 5000 );
}
```

$interval

Index.html (fragment only)

```
<div ng-app="myApp">
        <div ng-controller="myCtrl">
          <div>message : {{theTime}}</div>
        </div>
 </div>
```

Shows time every second

app.js

```
var myAppModule = angular.module('myApp', []);
myAppModule.controller('myCtrl', function($scope, $interval) {
        $scope.theTime = new Date().toLocaleTimeString();
        $interval(function () {
                $scope.theTime = new Date().toLocaleTimeString();
                }, 1000);
        });
```

Services as DOM object replacement

- For many services, it seems like you could use objects that are already in the DOM

(e.g. $location service vs window.location object).

- Since AngularJS constantly supervises your application, it is better to you use the service instead of DOM object (to handle changes and events properly).

$location

Index.html (fragment only)

```
<div ng-app="myApp">
        <div ng-controller="customersCtrl">
          <div>location : {{myUrl}}</div>
        </div>
</div>
```

Shows current URL

app.js

```
var myAppModule = angular.module('myApp', []);
myAppModule.controller('customersCtrl', function($scope, $location) {
          $scope.myUrl = $location.absUrl();
          });
```

Services as network support

- XHR support! (XmlHttpRequest == Ajax)

  $http

  $httpBackend

  $httpParamSerializer

  $httpParamSerializerJQLike

  $jsonpCallbacks

  $xhrFactory

$http

Index.html (fragment only)

```
<div ng-app="myApp" ng-controller="myHttpCtrl">
        <p>Today's welcome message is:</p>
        <h1>{{myWelcome}}</h1>
</div>
```

Shows message retrieved via XHR

app.js

```
var app = angular.module('myApp', []);
app.controller('myHttpCtrl', function($scope, $http) {
        $http.get("welcome.txt")
        .then(function(response) {
            $scope.myWelcome = response.data;
            });
        });
```

message.txt

```
This is today's welcome message.
```

$http service

- The $http service is a function which takes a single argument that is used to generate an HTTP request and returns a promise that is resolved (request success) or rejected (request failure) with a response object.

$http service

this callback will be called asynchronously when the response is available

```
$http({ method: 'GET', url: '/someUrl' }).
then(function successCallback(response) { … },
function errorCallback(response) { …});
```

called asynchronously if an error occurs or server returns response with an error status.

Example

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

Typical pattern

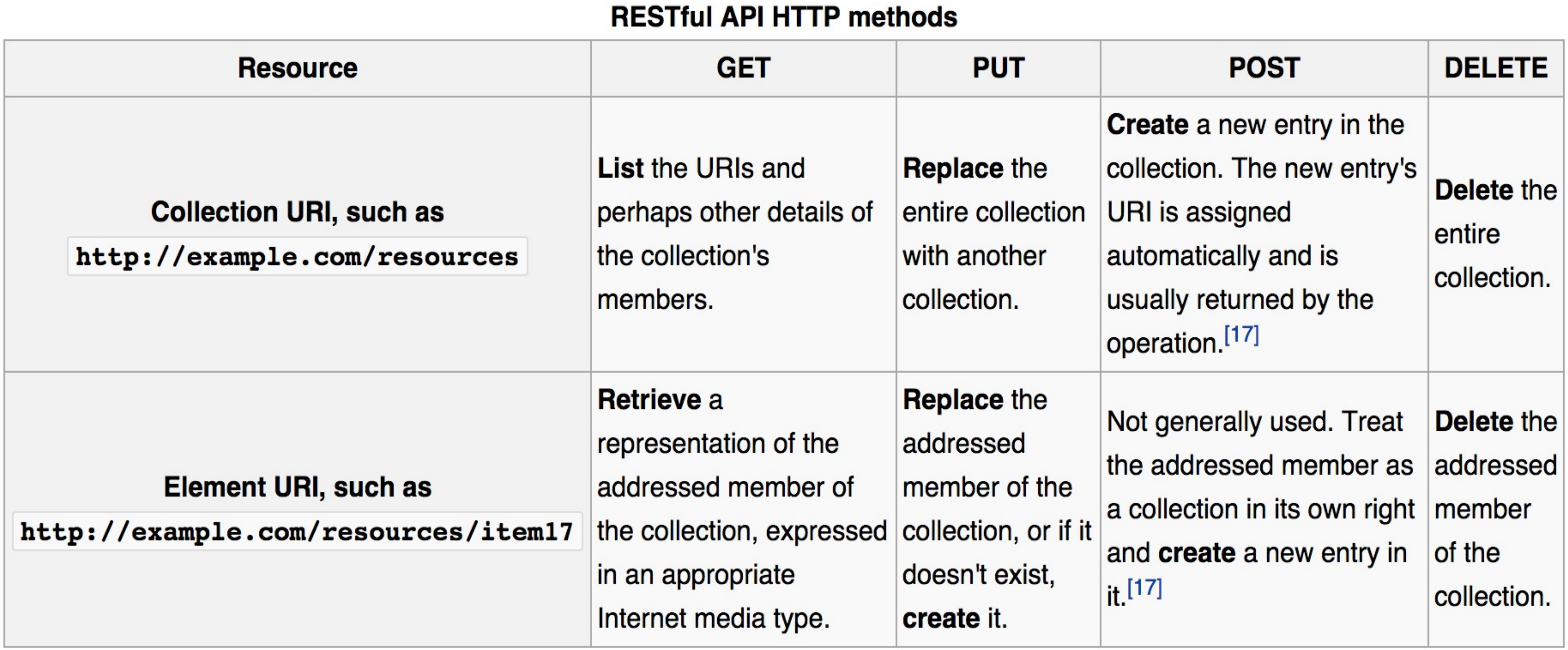
The controller uses http to retrieve data from the back end, and inject them into the view.

Methods

The .get method is a shortcut method of the $http service. There are several shortcut methods:

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

RESTful API HTTP methods (wikipedia)



RESTful API HTTP methods

Resource	GET	PUT	POST	DELETE
Collection URI, such as `http://example.com/resources`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	**Create** a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.[17]	**Delete** the entire collection.
Element URI, such as `http://example.com/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it doesn't exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[17]	**Delete** the addressed member of the collection.

READ UPDATE CREATE DELETE

HTTP Patch

The HTTP PATCH request method applies partial modifications to a resource.

The HTTP PUT method only allows complete replacement of a document.

Unlike PUT, PATCH is not idempotent (successive identical patch requests may have different effects)..

PATCH (like PUT) may have side-effects on other resources.

AngularJS:

Managing Events

on-mouseenter

Index.html (fragment only)

```
<div  ng-app="myApp" ng-controller="myCtrl" >
  <p >Click the button to run a function:</p>
  <button ng-mouseenter="myFunc()" >OK</button>
  <p>You passed over the button {{count}} times.</p>
</div>
```

Count how many times user passed over the button

app.js

```
angular.module('myApp', [])
  .controller('myCtrl', ['$scope', function($scope) {
    $scope.count = 0;
    $scope.myFunc = function() {
      $scope.count++;
    };
  }]);
```

Event list

- ng-click
- ng-dblclick
- ng-mousedown
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-mouseup

- ng-copy
- ng-cut
- ng-paste
- ng-focus
- ng-blur
- ng-keydown
- ng-keypress
- ng-keyup
- ng-change

AngularJS:

Routing

Routing

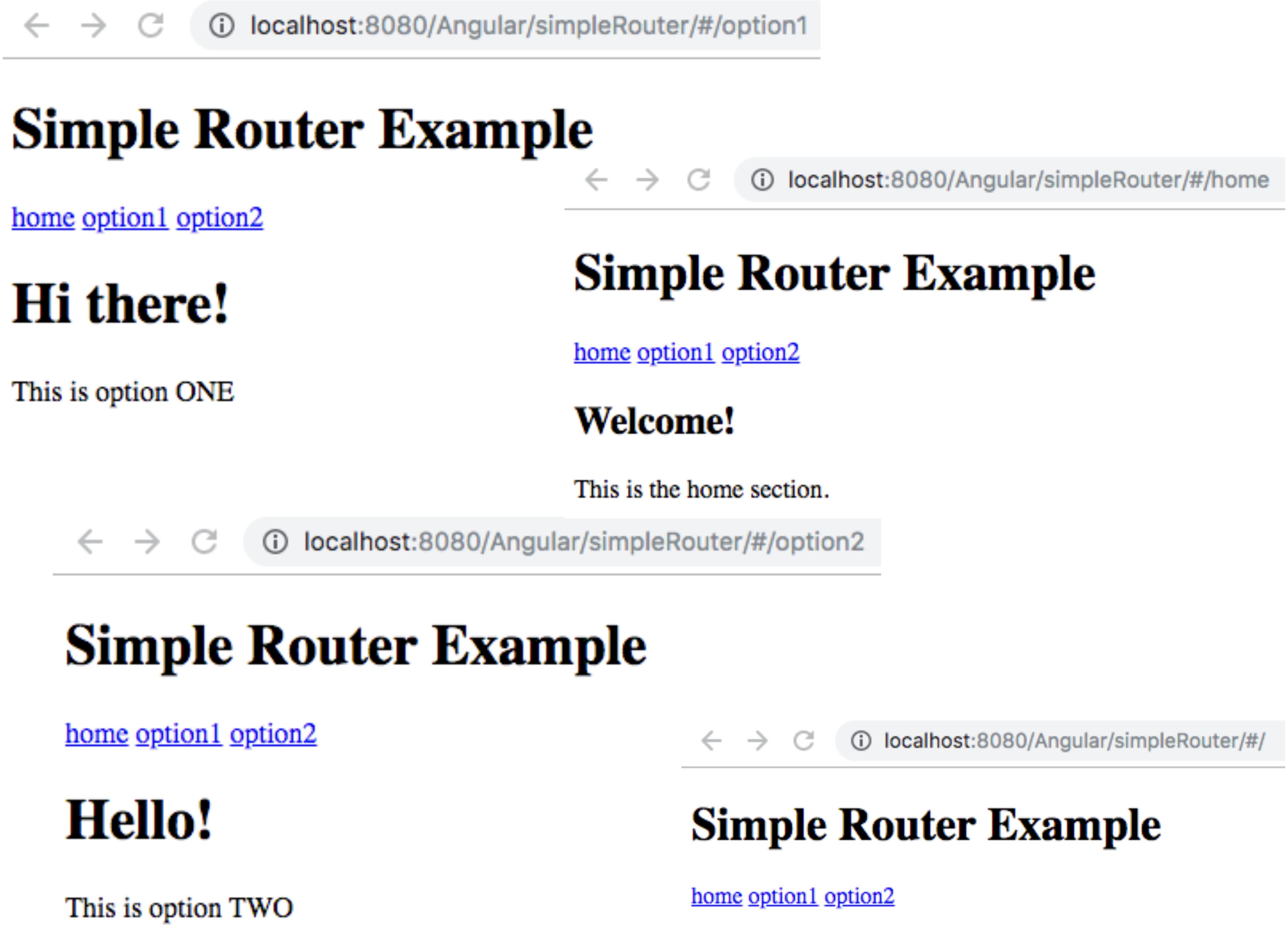
- Since **we are building a SPA** app, everything happens in **one page**
  - How should **back--button** work?
  - How should **linking** between "pages" work?
  - How about **URLs?**
- **Routing** comes to rescue!

app.js

```
angular.module("DemoApp", ['ngRoute'])
    .controller("DemoController", function($scope) {
        $scope.title = "Simple Router Example";
    })
    .config(['$routeProvider', function($routeProvider) {
        $routeProvider.
            when('/home', {
                template: '<h2>Welcome!</h2> This is the home section.',
            }).
            when('/option1', {
                templateUrl: 'option1.html',
            }).
            when('/option2', {
                  templateUrl: 'option2.html',
                  }).
            otherwise({
                redirectTo: '/'
            });
    }]);
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
<title>Routing</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.min.js"></script>
  <script src="app.js"></script>
</head>
<body ng-app="DemoApp" ng-controller="DemoController">
<h1>{{title}}</h1>
<a href="#home">home</a>
<a href="#option1">option1</a>
<a href="#option2">option2</a>
<div ng-view></div>
</body>
</html>
```



localhost:8080/Angular/simpleRouter/#/option1

Simple Router Example

home option1 option2

Hi there!

This is option ONE

localhost:8080/Angular/simpleRouter/#/home

Simple Router Example

home option1 option2

Welcome!

This is the home section.

localhost:8080/Angular/simpleRouter/#/option2

Simple Router Example

home option1 option2

Hello!

This is option TWO

localhost:8080/Angular/simpleRouter/#/

Simple Router Example

home option1 option2

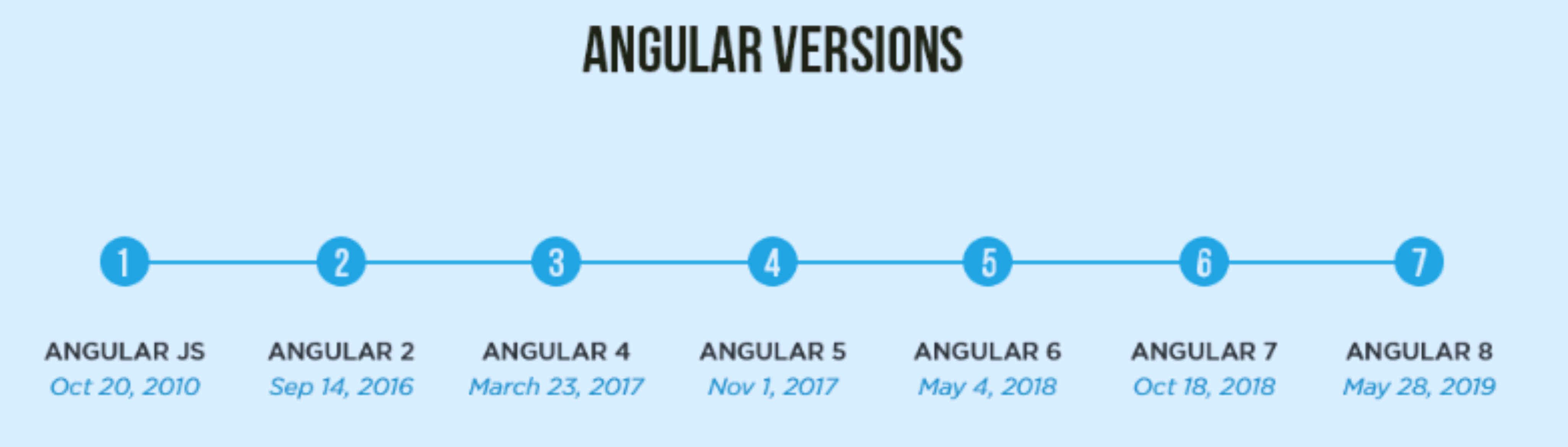
A full (commented) routing example

https://www.guru99.com/angularjs-views.html

Wrapping UP

- AngularJS is a modular JavaScript SPA framework
- Lot of great features, but learning curve can be hard
- Great for CRUD (create, read, update, delete) apps, but not suitable for every type of apps
- Works very well with some JS libraries (JQuery)

Angular versions



- See https://www.clariontech.com/blog/angular-framework-from-its-first-steps-to-adulthood