Matias Niemelä - **GOTO Copenhagen 2014**

BUILD A STRONG ANGULARJS FOUNDATION

ABOUT THE INSTRUCTOR(S)

Name: **Matias Niemelä**
Email: **matias@yearofmoo.com**
Twitter: **@yearofmoo**
Website: www.yearofmoo.com

Name: **Lukas Ruebbelke (not here)**
Email: **simpul@gmail.com**
Twitter: **@simpulton**
Website: http://onehungrymind.com/

DOWNLOAD DETAILS

What are we building?
http://ng-tube.com/

Make sure to have the following installed:
- git
- nodejs (with npm)
- grunt (via npm)
- karma (via npm)

Install the application code:
- Read the README here: https://github.com/angularjs-foundation/video-exercises#angularjs-foundation---hands-on-exercises

THE AGENDA

M1: HELLO ANGULARJS

M2: MODULE

M3: CONTROLLER

M4: VIEW

M5: SERVICES

M6: DIRECTIVES

M1: HELLO ANGULARJS

AGENDA

What is AngularJS
The Big Picture
Hello World
Exercise
Karma, Jasmine and a Hello World Test
Exercise
Review

THE HISTORY OF ANGULARJS

Developed in 2009 by **Misko Hevery**.
Publicly released as version 0.9.0 in October 2010
Currently at version 1.2.x

ANGULARJS IN 1 MINUTE

An intuitive framework that makes it **easy** to organize your code.

Testable code makes it **easier** to sleep at night.

Two-way data binding **saves** you **hundreds** of lines of code.

Templates that are HTML means you **already** know how to write them.

Data structures that are just JavaScript make integration **really easy**.

MVC WEBSITES IN 1 MINUTE

AngularJS is a client-side MVC framework

Model + View + Controller

Model = Data

View = HTML Template Code

Controller = JavaScript code without the DOM

WHY CLIENT-SIDE MVC?

JavaScript applications need structure

M + V + C = a logical separation of the main components

Easier to test the code properly

COMMUNITY STATS

Over 4000 closed issues

Large community of JavaScript developers

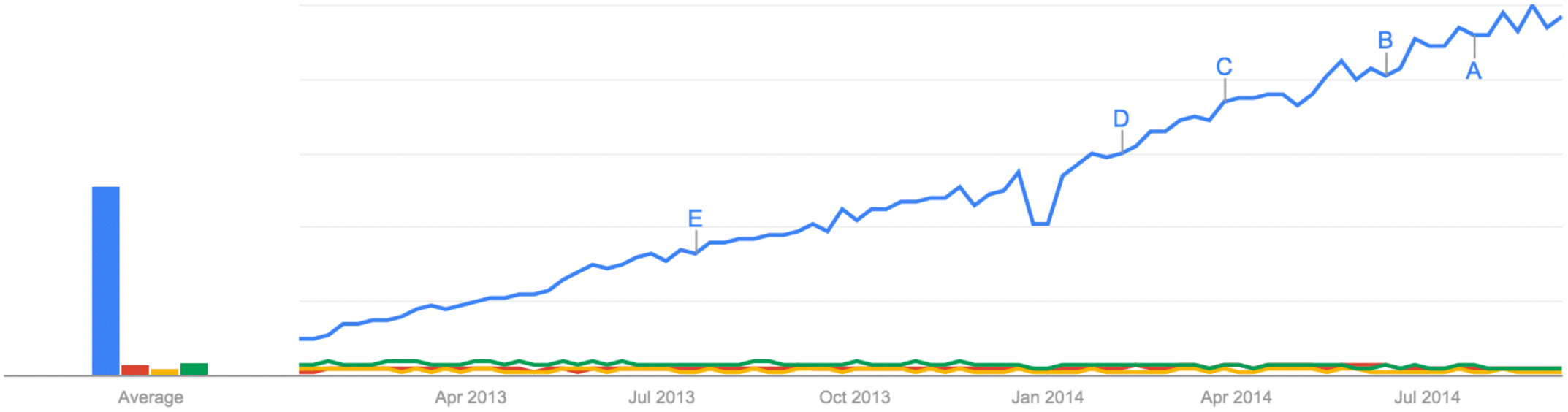
Version 1.2.25 is the current stable version

Version 1.3.0-rc.2 is the current edge version

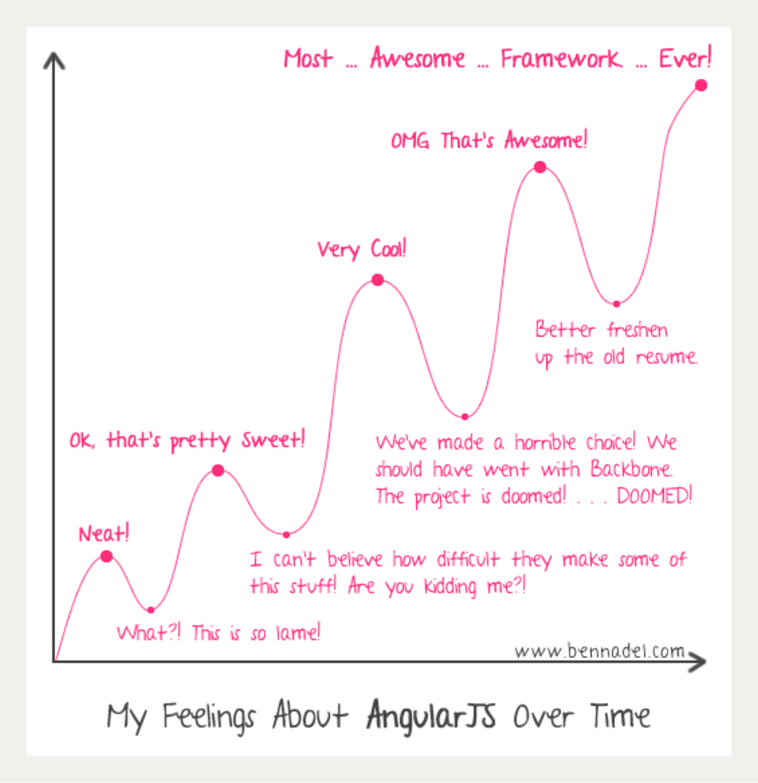
(stable vs unstable doesn’t mean what you expect it to)

**** we will be using 1.3.0-rc.2 for this workshop ****

THE RISE OF ANGULARJS

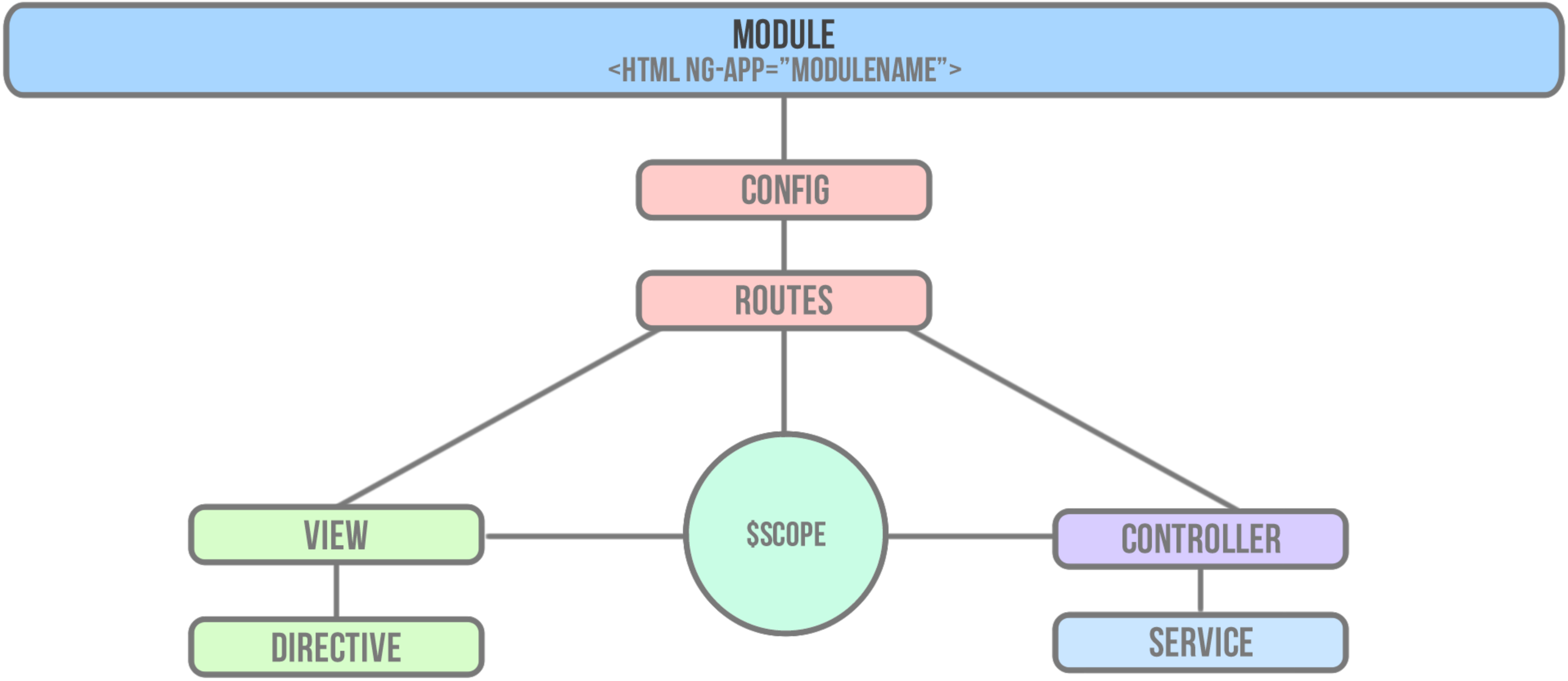


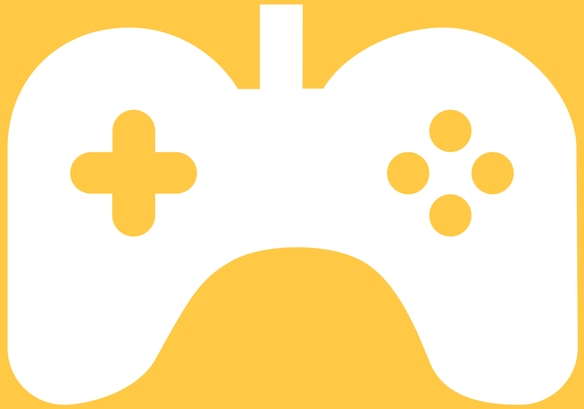
THE ANGULARJS LEARNING CURVE



THE ANGULARJS BIG PICTURE

The **majority** of the work you are going to do with **AngularJS** will fall within a **core** set of **AngularJS** components that follow MVC.


MODULE
<HTML NG-APP=”MODULENAME”>
CONFIG
ROUTES
VIEW
$SCOPE
CONTROLLER
DIRECTIVE
SERVICE



HELLO WORLD

THE SCOPE

The **scope** is the memory of the application

Anything placed on the scope can be rendered in the template.

TEMPLATE-BASED DATA

Any valid form of **JavaScript data** can be placed **on the scope**

```
<body ng-app>
   <div ng-init="name='John'">
      <div>My name is: {{ name }}</div>
   </div>

   <div ng-init="name=function() { return 'Rita'; }">
      <div>My name is: {{ name(); }}</div>
   </div>
</body>
```

HOW DOES IT UPDATE ITSELF?

How does it update the UI?

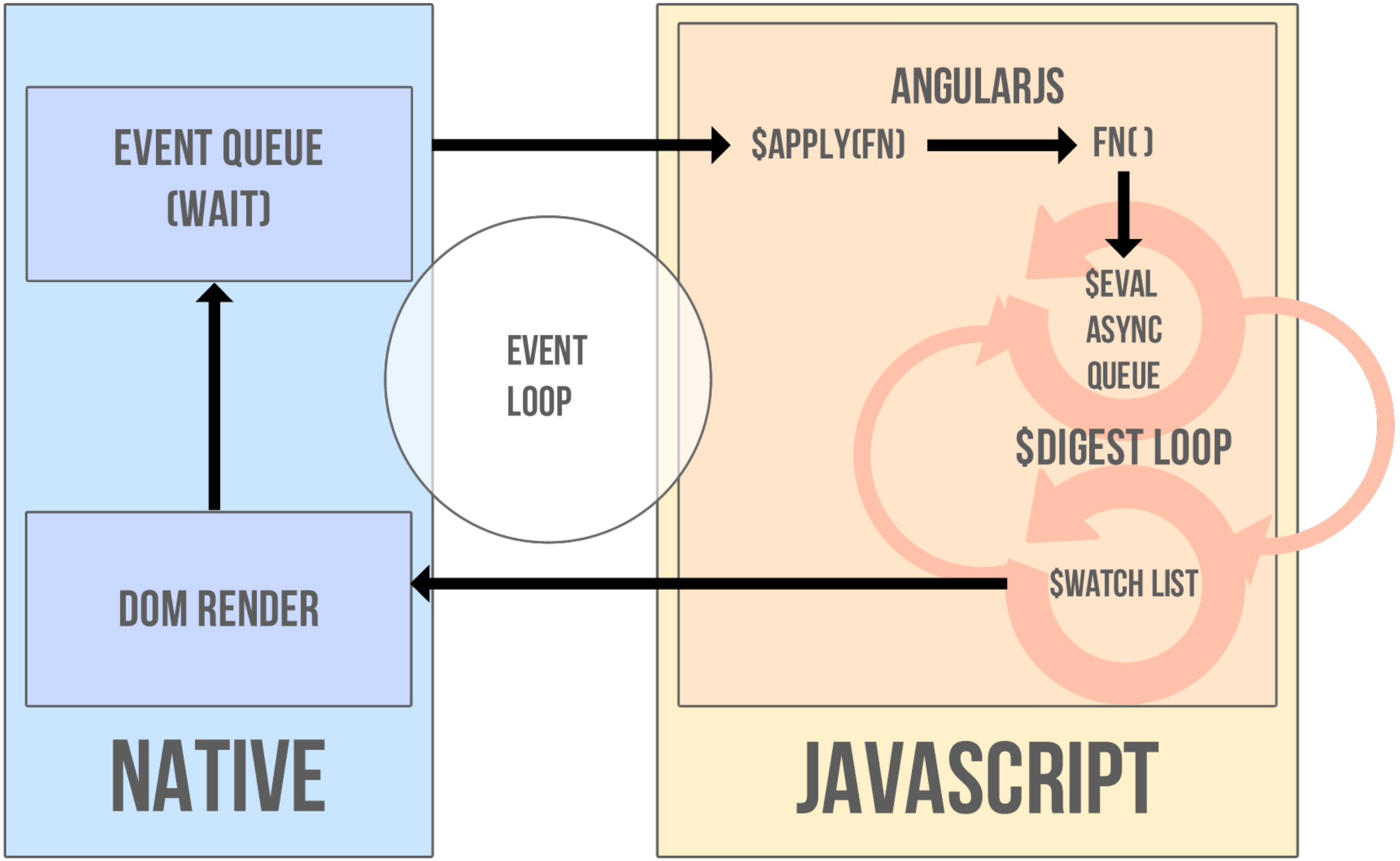
When data changes, Angular looks at the differences and then performs the necessary updates

$DIGEST AND $APPLY

$digest examines the scope to see if anything has changed and then updates the template code afterwards

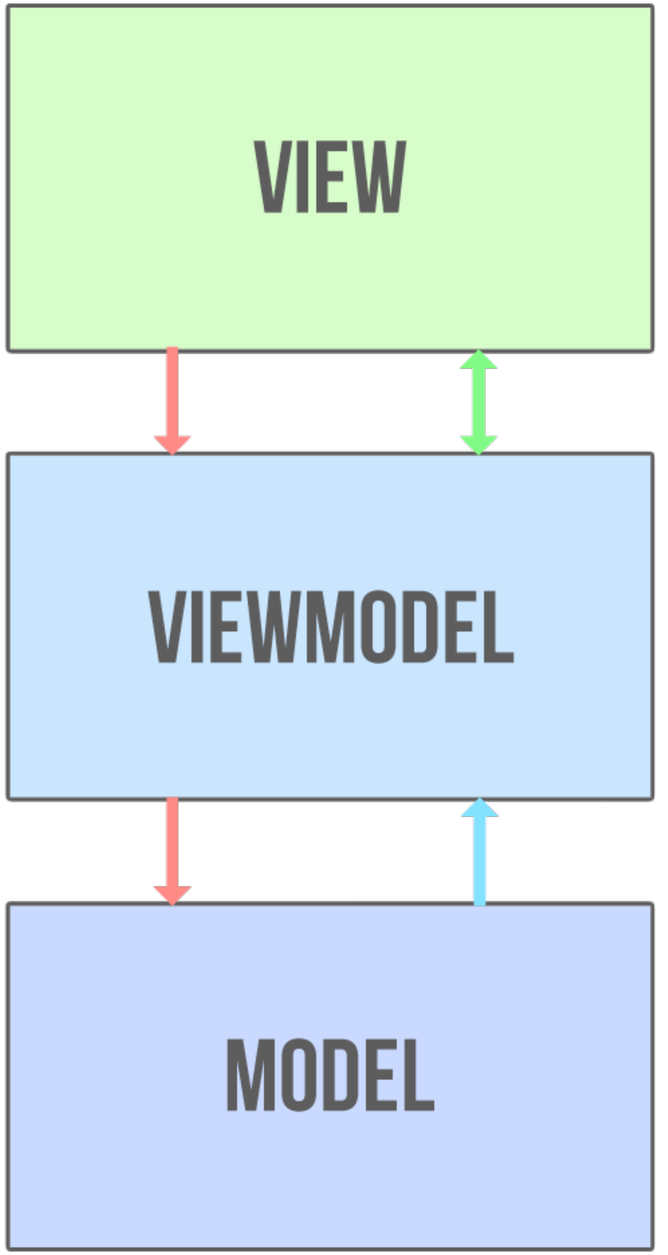
$digest is called internally by the main AngularJS template code and components.

$apply() is used to notify that something has happened outside of the **AngularJS** domain


EVENT QUEUE
(WAIT)
DOM RENDER
NATIVE
EVENT
LOOP
ANGULARJS
$APPLY(FN)
FN( )
$EVAL
ASYNC
QUEUE
$DIGEST LOOP
$WATCH LIST
JAVASCRIPT

MODEL VIEW WHATEVER

Choose **whatever** pattern helps you be more productive.



MVVM

COMMANDS

DATA BINDINGS

CHANGE NOTIFICATION

TESTING JAVASCRIPT

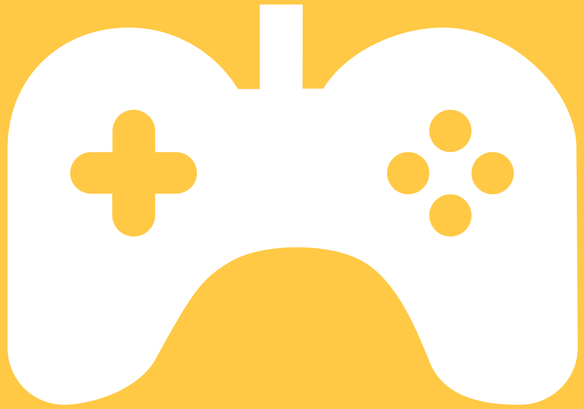
```
describe(“I am testing this feature”, function() {
  it(“should behave this way”, function() {
    var result = someOperation();
    var expectedResult = “expected value”;
    expect(result).toEqual(expectedResult);
  });
});
```

KARMA

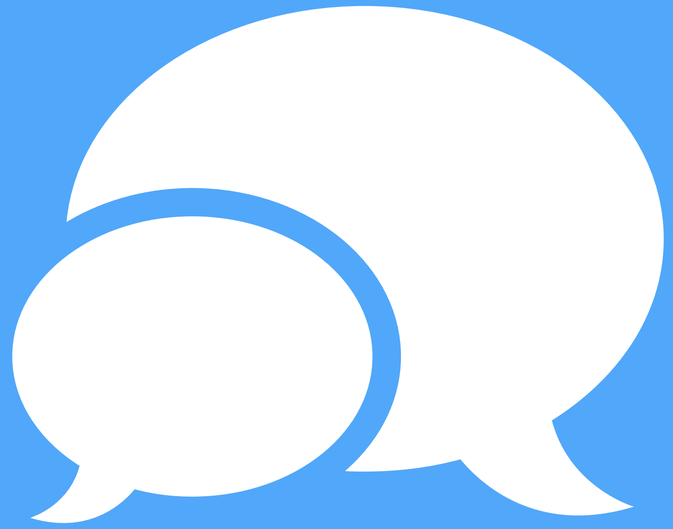
We use a tool called **karma** to run our front-end JavaScript tests

Karma will run each of the tests and tell us which have failed

```
INFO [Chrome 34.0.1847 (Mac OS X 10.9.2)]: Connected on socket GDCZKcqZFpHoYVyUrQPV with id 9967315
Chrome 34.0.1847 (Mac OS X 10.9.2) ntApp should define a version higher than 1.0 FAILED
        Expected 1 to be greater than 1.
        Error: Expected 1 to be greater than 1.
            at null.<anonymous> (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/test/unit/ntAppSpec.js:5:21)
            at Object.invoke (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/app/lib/angular/angular.js:3869:17)
            at workFn (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/app/lib/angular-mocks/angular-mocks.js:2147:20)
Chrome 34.0.1847 (Mac OS X 10.9.2) ntApp should have a working isWeekend() scope function FAILED
        Expected false to be true.
        Error: Expected false to be true.
            at null.<anonymous> (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/test/unit/ntAppSpec.js:17:36)
            at Object.invoke (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/app/lib/angular/angular.js:3869:17)
            at workFn (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/app/lib/angular-mocks/angular-mocks.js:2147:20)
        Expected false to be true.
        Error: Expected false to be true.
            at null.<anonymous> (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/test/unit/ntAppSpec.js:23:36)
            at Object.invoke (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/app/lib/angular/angular.js:3869:17)
            at workFn (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/app/lib/angular-mocks/angular-mocks.js:2147:20)
Chrome 34.0.1847 (Mac OS X 10.9.2) ytCore should have a working ytCore module FAILED
        Expected false to be true.
        Error: Expected false to be true.
            at null.<anonymous> (/Volumes/Data/projects/angularjs-foundation/video-exercises/hands-on-exercises/M2/test/unit/ytCoreSpec.js:9:20)
Chrome 34.0.1847 (Mac OS X 10.9.2): Executed 3 of 3 (3 FAILED) ERROR (0.055 secs / 0.047 secs)
```



JASMINE TEST



REVIEW

M2: THE MODULE

AGENDA

Module
Build Your First Module
Exercise
module.constant, module.run + the DI
Exercise
Multiple Modules
Exercise
Review

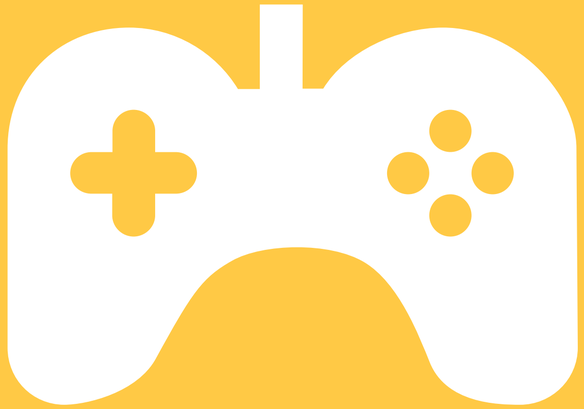
THE ANGULARJS MODULE

A **module** is used to house the JavaScript code for your application

A **module** consists of controllers, directives, factories, initializers and configurations.

WHAT A MODULE LOOKS LIKE

```
<html ng-app="MyModule">
  …
  <script type="text/javascript">
    var myModule = angular.module("MyModule", [])
  </script>
</html>
```



YOUR FIRST MODULE

MODULE.VALUE, MODULE.CONSTANT

Registers a **member** value on the module.

```
myModule.value(
   “applicationName”, “AngularJS Foundation”);
```

Registers a **constant** value on the **module (**Excellent strategy for storing **configuration** or application wide variables that do not change)

```
myModule.constant(“VERSION”, “2.0”);
```

Contant values are usually uppercase.

MODULE.RUN

Executed when all modules have been loaded
Excellent for executing system wide code that only needs to be run once

```
myModule.run(function(VERSION, $rootScope) {
   //this code is run when the application starts
   $rootScope.version = VERSION;
});
```

DEPENDENCY INJECTION (DI)

AngularJS uses **DI** to resolve dependencies within the application based on the parameter values that are placed in the function.

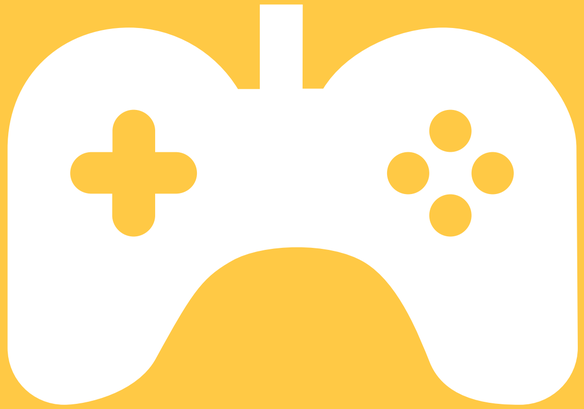
```
myModule.run(function(VERSION, $rootScope) {
   //VERSION is injected
   //$rootScope is injected
});
```

DEPENDENCY INJECTION (DI)

While the parameters work in development, we should use the array syntax to avoid code compression errors.

myModule.run([‘VERSION’, ‘$rootScope’, function(VERSION, $rootScope) {

```
   //VERSION is injected
   //$rootScope is injected
}]);
```

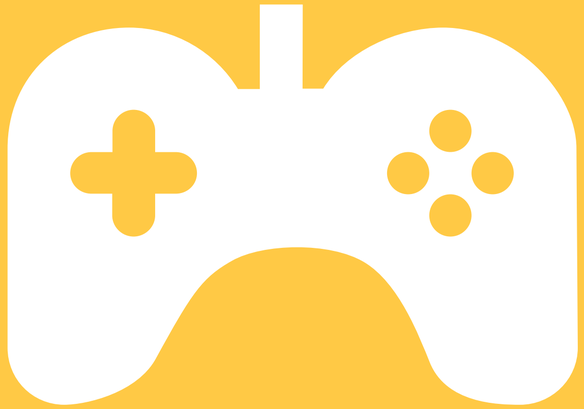


MODULE.RUN

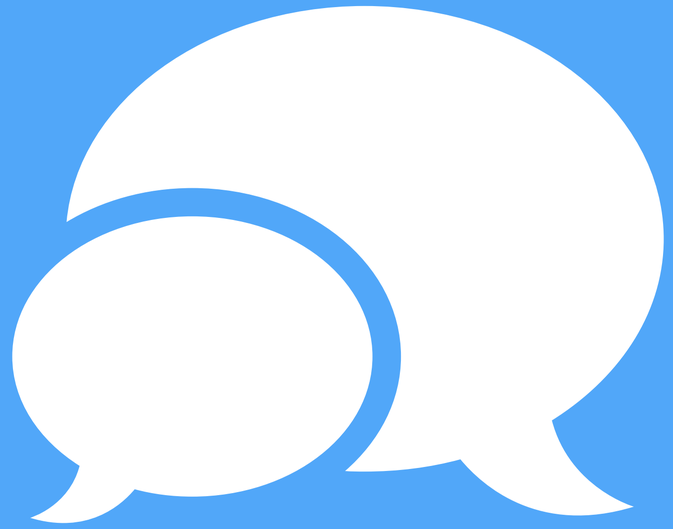
MULTIPLE MODULES

```
angular.module(‘Module1’, [])
   .constant(‘VERSION’, ‘2.0’);

angular.module(‘Module2’, [‘Module1’]);
     .run([‘VERSION’, ‘$rootScope’,
   function(VERSION,   $rootScope) {
     $rootScope.version = VERISON;
   }])
```



MULTIPLE MODULES



REVIEW

M3: THE CONTROLLER

AGENDA

Controllers and $scope
Build Your First Controller
Exercise
Methods and Properties on $scope
Exercise
Routes with Controllers
Exercise
Review

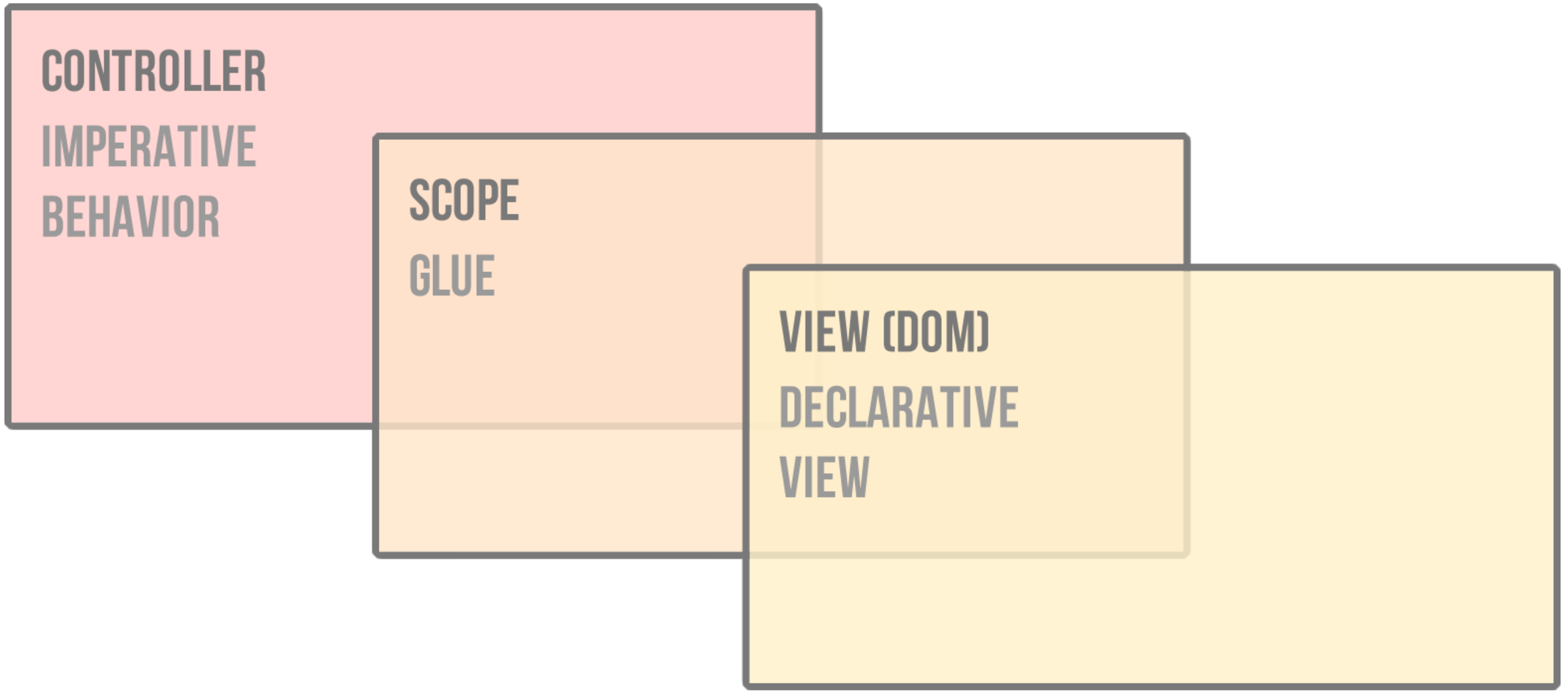
THE ANGULARJS CONTROLLER

$scope is the glue between the **Controller** and the **View**

The **Controller** is responsible for constructing the model on **$scope** and providing commands for the **View** to act upon

$scope provides context

$rootScope is the topmost **Scope** object with all other children **$scope** objects prototypically inheriting from it

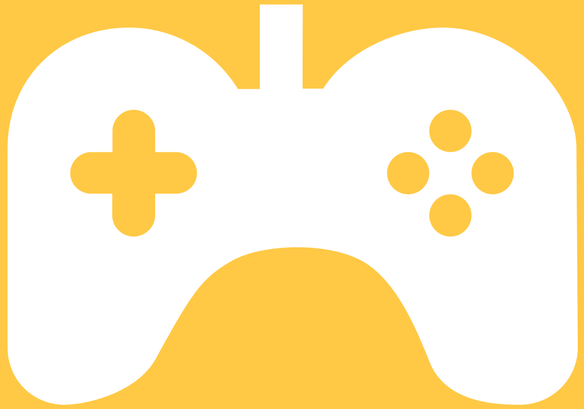

CONTROLLER
IMPERATIVE
BEHAVIOR
SCOPE
GLUE
VIEW (DOM)
DECLARATIVE
VIEW

CONTROLLER

A **Controller** is like **run()** and it is defined on the module, but unlike run(), a controller can be reused multiple times and it contains it's own scope

```
ngModule.controller("HomeCtrl",
   ['$scope', function($scope) {

   $scope.title = "Hello There";
}]);
```



YOUR FIRST CONTROLLER

METHODS & PROPERTIES ON $SCOPE

When properties are added to **$scope**, they are available for binding with the **View**

When methods are added to **$scope**, they are available to be called from the **View**

AngularJS also allows you to bind to methods as well as simple values

METHODS & PROPERTIES ON $SCOPE

```
ngModule.controller(“HomeCtrl”,
   [‘$scope’, function($scope) {


   $scope.title = “Hello There”;
   $scope.isReady = function() { return true; };
}]);
```

CONTROLLER INSIDE OF HTML

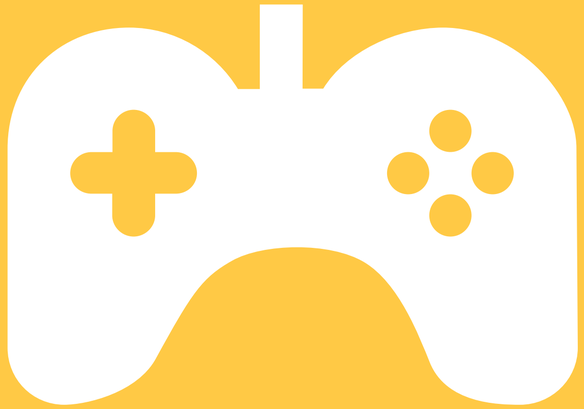
We can then use the controller directly inside of the HTML code.

```
<div ng-controller="HomeCtrl">
   <strong>{{ title }}</strong> <br />
   READY = <strong>{{ isReady }}</strong>
</div>
```

CONTROLLER-AS

We can also alias the controller as its own object to keep the scope clean

```
ngModule.controller(“HomeCtrl”, function() {
  this.title = “Hello There”;
  this.isReady = function() { return true; };
});
<div ng-controller=“HomeCtrl as home”>
  <strong>{{ home.title }}</strong> <br />
  READY = <strong>{{ home.isReady }}</strong>
</div>
```



METHODS AND PROPERTIES ON $SCOPE

ROUTES WITH CONTROLLERS

Use the ngRoute module to setup URL-based routes for your application

Routes help you keep state as to where the user is when browsing

Routes by default use a hash bang (#!) URL, but HTML5 URLs are also supported.

$routeProvider is used to define routes

<div **ng-view**>… is used as the container element.

NG-ROUTE

First include **angular-route.js** into your application and add **ngRoute** as a module dependency. Then define some routes.

```
angular.module(‘MyApp’, [‘ngRoute’])
    .config([‘$routeProvider’, function($routeProvider) {
        $routeProvider.when(‘/‘, {
            templateUrl : ‘/home_tpl.html’,
            controller: ‘HomeCtrl as home’
        });
    }])
```

NG-ROUTE

Remember to also add <div ng-view> to your index.html file

```
<body>
    <div ng-view></div>

    <script type="text/javascript" src="./angular.js"></script>
    <script type="text/javascript" src="./angular-route.js"></script>
    <script type="text/javascript" src="./app.js"></script>
</body>
```

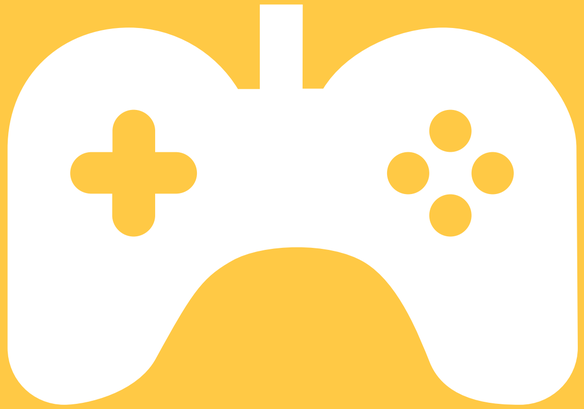
NG-ROUTE

By default, hash-based URLs are used, so add the prefix into your links

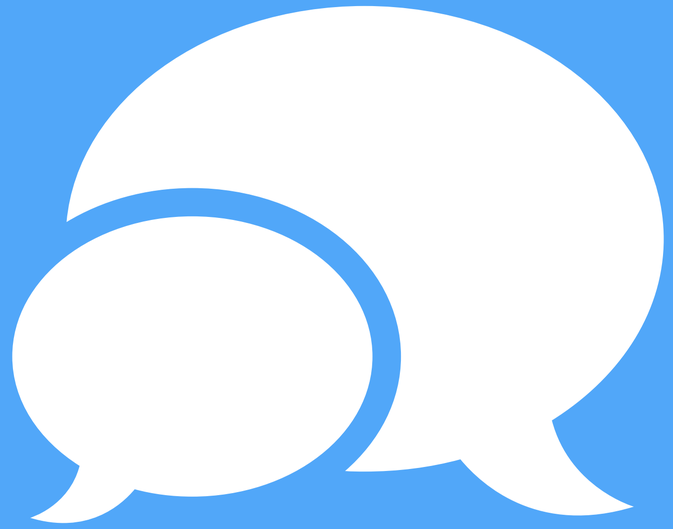
```
<a href="#/">Home Page</a>
<a href="#/users">Users Page</a>
```

This can be changed via **$locationProvider** inside of **config()**

```
ngModule.config(['$locationProvider', function($locationProvider) {
    $locationProvider.html5Mode(true);
}])
```



ROUTES AND CONTROLLERS



REVIEW

M4: THE VIEW

AGENDA

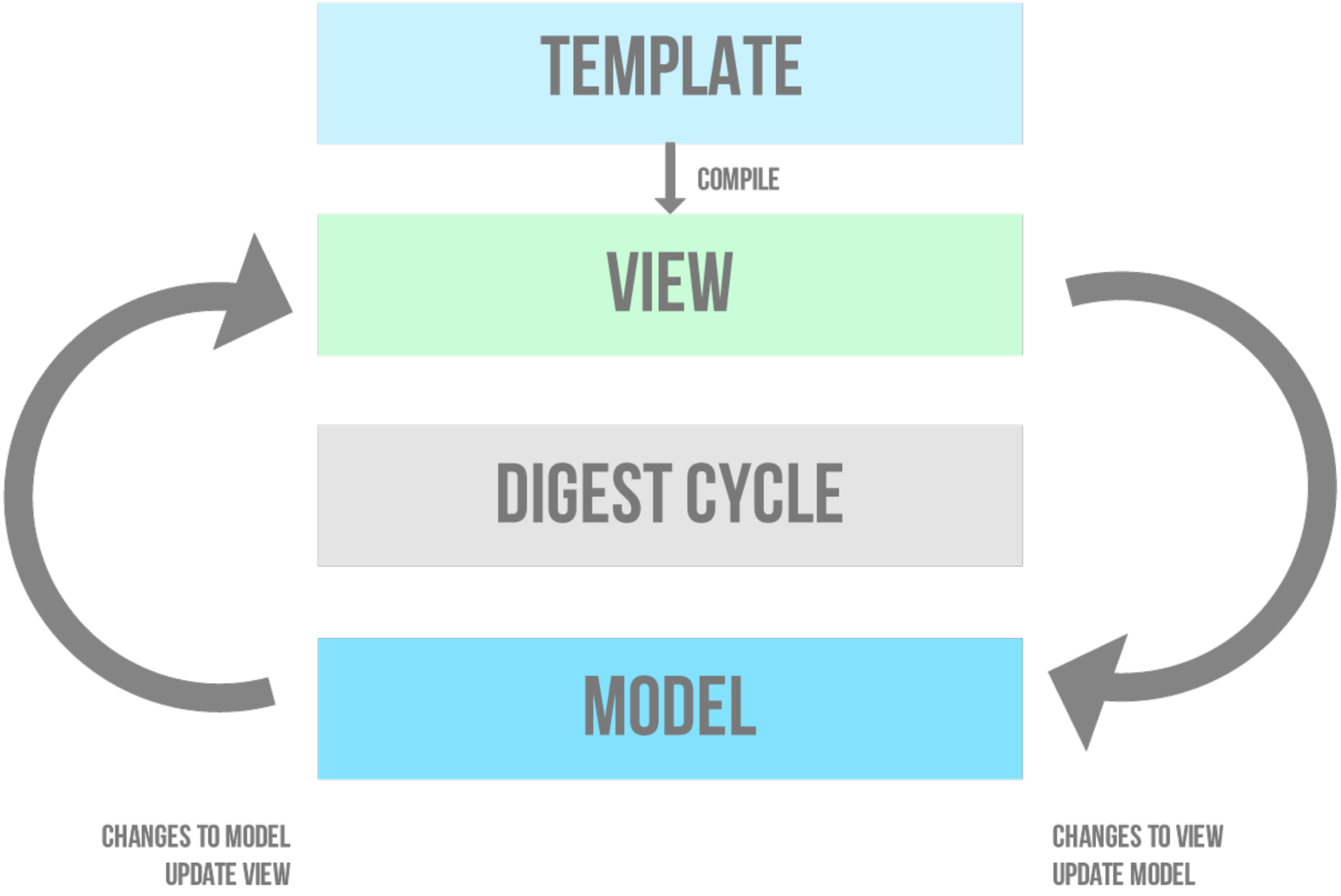
Views in AngularJS
AngularJS Directives Tour
Layout Directives
Exercise
Event Directives
Exercise
Misc' Directives
Exercise
Review

THE ANGULARJS VIEW

The **View** is when a template is compiled with a scope/controller.

The scope handles all interaction and data flow between the controller and the DOM.

It is wrong to update the DOM manually otherwise.


TEMPLATE
COMPILE
VIEW
DIGEST CYCLE
MODEL
CHANGES TO MODEL
UPDATE VIEW
CHANGES TO VIEW
UPDATE MODEL

ANGULAR DIRECTIVES

Directives extend the capabilities of HTML to do more

How do you have a list of items in HTML without using JavaScript?

Directives also reduce a ton of JavaScript code

Finally directives make the template code easy to follow and understand without the need to read the underlying JavaScript code

Angular comes packed with a bunch of predefined directives. All Angular directives are prefixed with **ng-**

ANGULAR DIRECTIVES

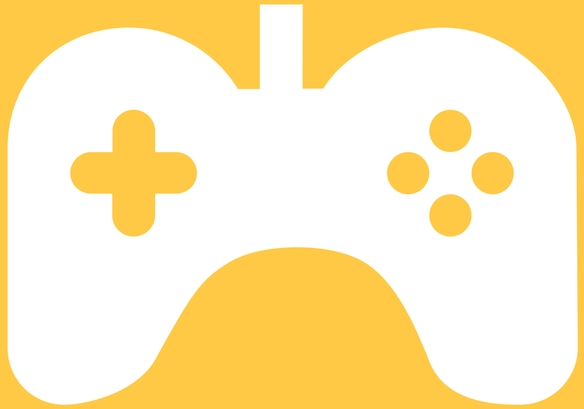
```
<!- quite easy to see what's going on here ->
<div ng-controller="ItemsCtrl as items">
  <div ng-repeat="item in items.entries">
    <h2>{{ item }}</h2>

    <button ng-click="items.remove(item)">
      Remove
    </button>
  </div>
</div>
```

LAYOUT DIRECTIVES

== Directives that control the **layout/structure** of the page

- ngIf
- ngShow / ngHide
- ngRepeat
- ngSwitch
- ngInclude

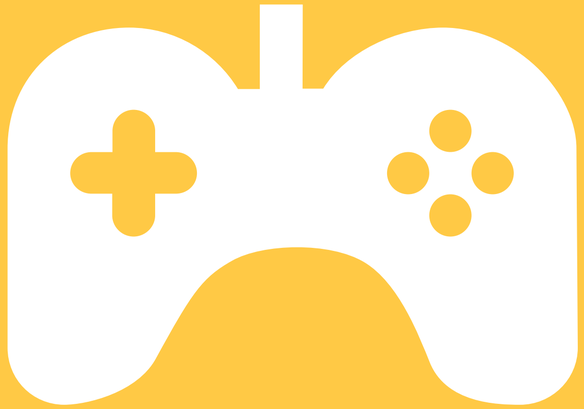


LAYOUT DIRECTIVES

INTERACTION DIRECTIVES

== Directives that handle event listens and trigger callbacks on the **Controller**

- ngModel + ngModelOptions
- ngBlur / ngFocus
- ngClick
- ngSubmit
- ngMessages

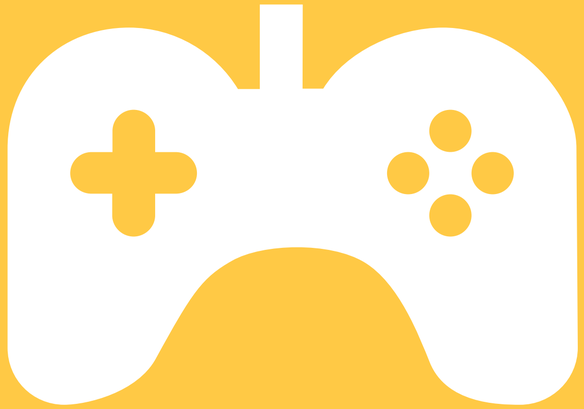


EVENT DIRECTIVES

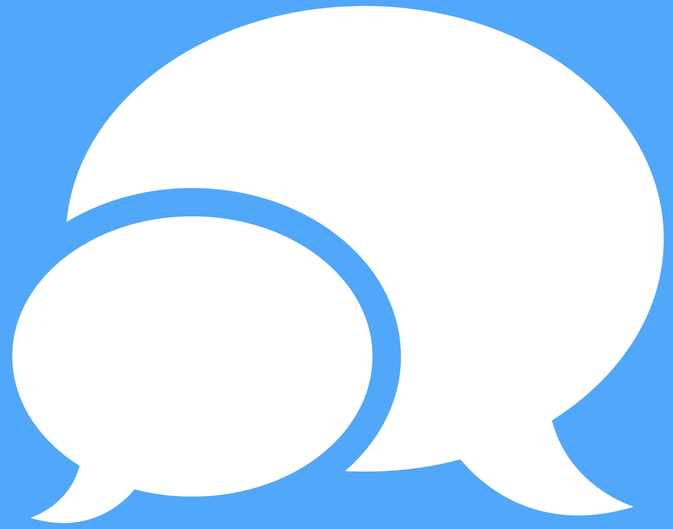
MISC' DIRECTIVES

Directives that alter the styling and other attributes for elements

- ngClass
- ngStyle
- attr="{{ val }}"



MISC' DIRECTIVES



REVIEW

M5: SERVICES

AGENDA

Services in AngularJS
Build Your First Service
Exercise
The $http Service
Exercise
Communicating with Controllers
Exercise
Review

ANGULARJS SERVICES

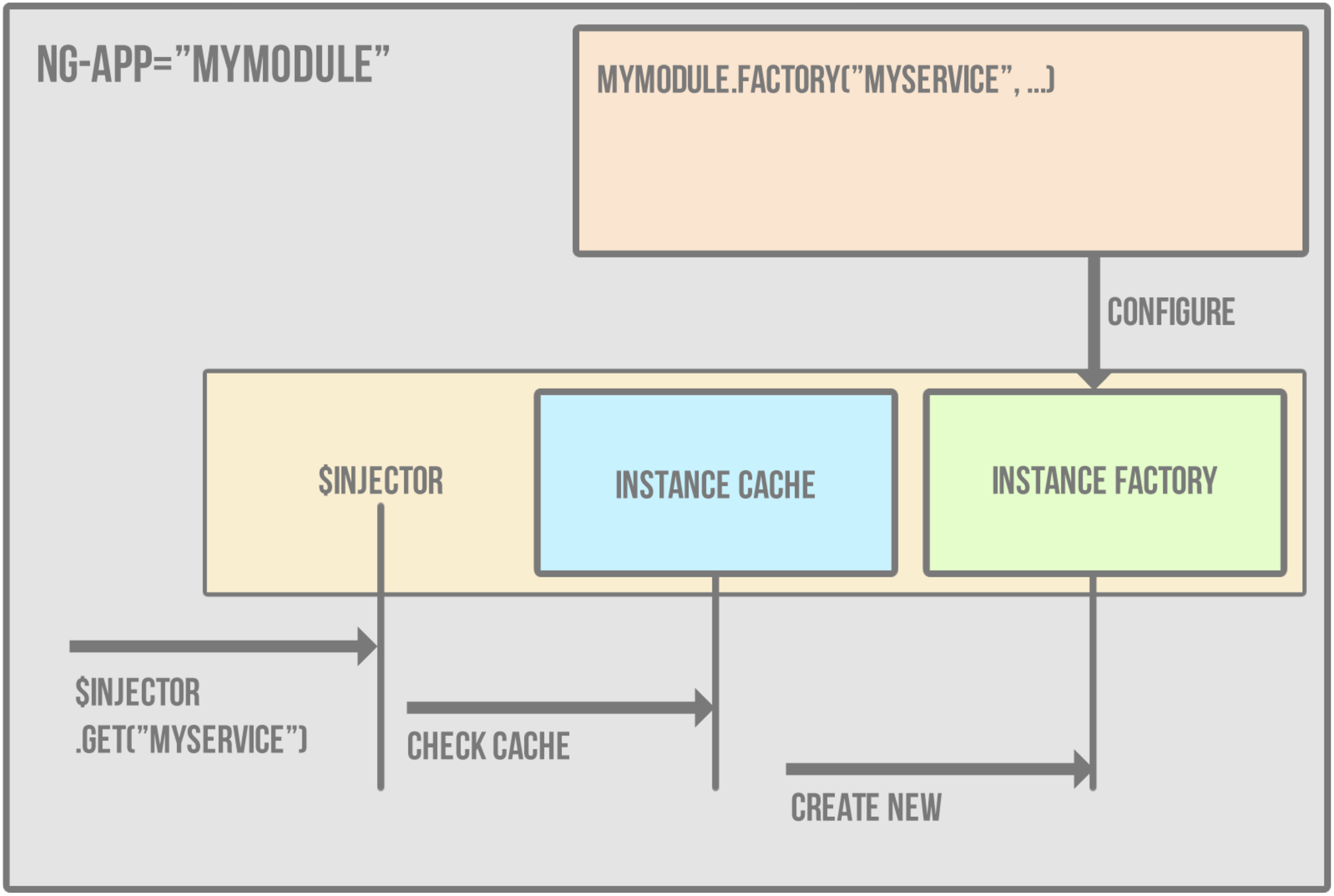
Services carry out common tasks specific to the web application

Services are available via **Dependency Injection**

Services are only initialized once

Services are **instantiated lazily**

All Angular services have a **$** as a prefix


NG-APP="MYMODULE"
MYMODULE.FACTORY("MYSERVICE", ...)
CONFIGURE
$INJECTOR
INSTANCE CACHE
INSTANCE FACTORY
$INJECTOR
.GET("MYSERVICE")
CHECK CACHE
CREATE NEW

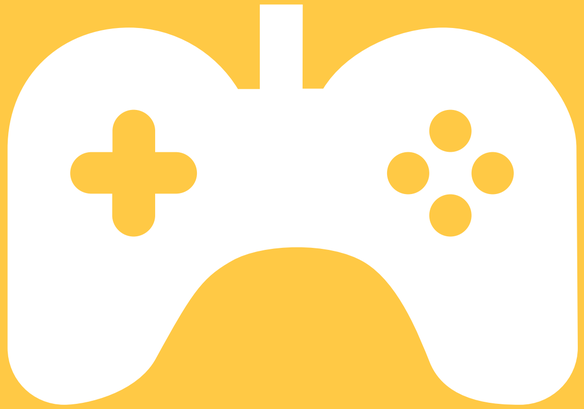
EXAMPLE SERVICE

```
ngModule.factory('isValidEmailAddress', function() {
   return function(email) {
      return /\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]
{2,4}\b/.test(email);
   };
});
```

EXAMPLE SERVICE

```
ngModule.controller('LoginCtrl',
   ['isValidEmailAddress', function(isValidEmailAddress) {

   this.isEmailValid = function(email) {
      return isValidEmailAddress(email);
   };
}]);
```



YOUR FIRST SERVICE

THE $HTTP SERVICE

The **$http** service is a core Angular service that facilitates communication with the remote HTTP servers via the browser's **XMLHttpRequest** object or via **JSONP**.

When $http is fired then a promise is returned. A promise is a callback function wrapped in an object.

THE $HTTP SERVICE

```
ngModule.factory('emailAvailableChecker',
   ['$http', function($http) {

   return function(email) {
      return $http.get('/api/email-available?email=' + email);
   };
}]);
```

THE $HTTP SERVICE

```
ngModule.controller(“RegisterCtrl”,
   [‘emailAvailableChecker’, function(emailAvailableChecker) {

   this.register = function(email) {
      var status;
      emailAvailableChecker(email).then(
         function() { status = ‘available’; },
         function() { status = ‘unavailable’; });
   };
}]);
```

$HTTP SHORTCUT METHODS

$http.get

$http.head

$http.post

$http.put

$http.delete

$http.jsonp

PROMISES

The returned value of calling **$http** is a **promise**

You can use the **then** method to register callbacks

These callbacks receive a single argument - an object representing the response

HTTP PROMISES

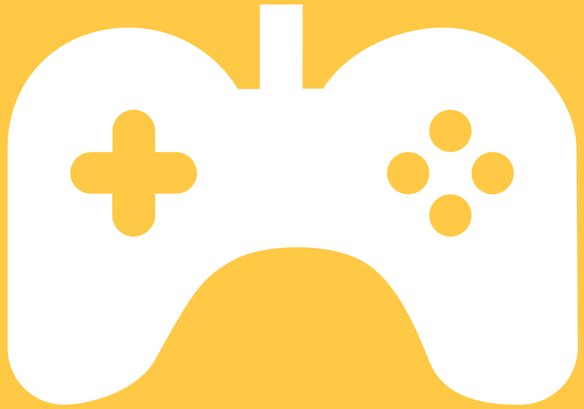
```
// standard then function
$http.get('/api/countries.json').then(function(data) {
   // data.countries == ['Denmark', 'Canada', 'Finland', 'USA'];
});

// success function
$http.get('/api/countries.json').success(function(data) {
   // data == ['Denmark', 'Canada', 'Finland', 'USA'];
});
```

HTTP PROMISES

```
// standard then function
$http.get('/api/countries.json').then(null, function(data) {
   // data.error == "Page Not Found";
});

// error function
$http.get('/api/countries.json').error(function(data) {
   // data == "Page Not Found"
});
```



$HTTP

COMMUNICATION BETWEEN CONTROLLERS

Controllers should **not** execute shared functions

Use a **service** or **use scope events** to facilitate communication

The **$scope** can be used to emit and listen on events from other scopes

$broadcast sends events downwards

$emit sends events upwards

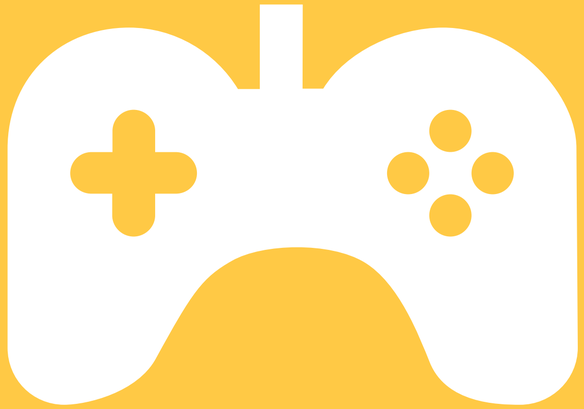
$on listens for events

SCOPE-BASED EVENTS

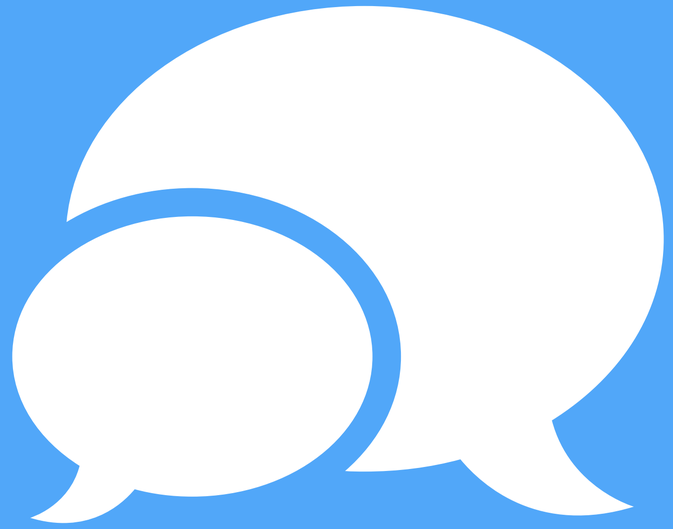
Use **$rootScope** for **broadcasting** events and listen via the local **$scope**

```
ngModule.controller('CtrlA', ['$rootScope', function($rootScope) {
    $scope.$broadcast('hello', "let's hang out");
}]);

ngModule.controller('CtrlB', ['$scope', function($scope) {
    $scope.$on('hello', function(event, property) {
        //property == "Let's hang out"
    });
}]);
```



COMMUNICATING WITH CONTROLLERS



REVIEW

M6: DIRECTIVES

AGENDA

AngularJS Directives Simplified
Build Your First Directive
Exercise
Directive Definition Object
Exercise
Link Function
Exercise
Controller Function
Exercise
Review

WHY A DSL?

> People find DSLs valuable because a well-designed DSL can be much easier to program with than a traditional library. This improves programmer productivity, which is always valuable. **In particular it may also improve communication with domain experts, which is an important tool for tackling one of the hardest problems in software development.**

MARTIN FOWLER

FIXED LANGUAGES ARE BROKEN

> “[Without the ability to reprogram the language] programmers resort to repetitive, error-prone workarounds. **Literally millions of lines of code have been written to work around missing features in programming languages.**”

STUART HOLLOWAY

DIRECTIVES 101

We've been using directives all along, but nothing custom.

Custom directives can be created using .directive() module function.

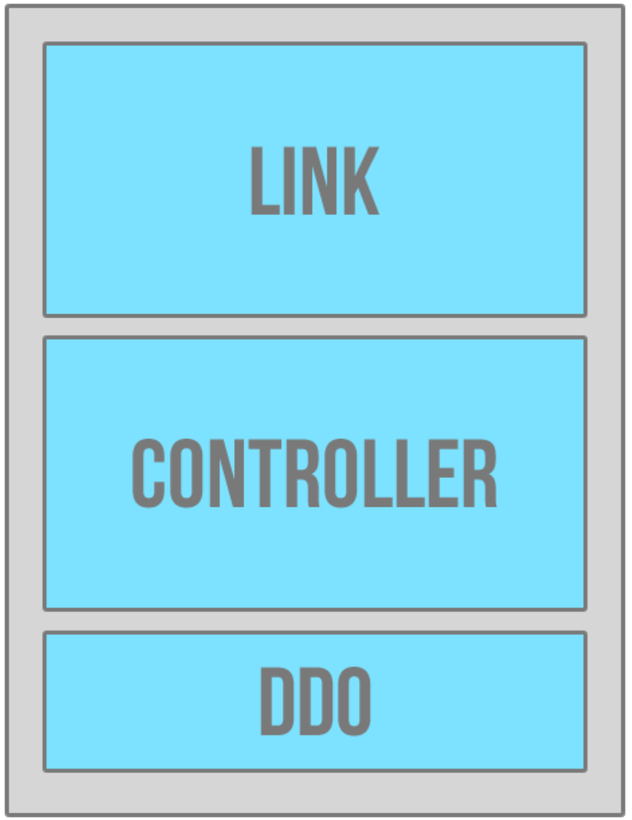
```
//<div say-hello>Click me to say hello</div>
ngModule.directive('sayHello', function() {
   return function(scope, element, attrs) {
      element.on('click', function() {
         alert('Hello!");
      });
   };
});
```

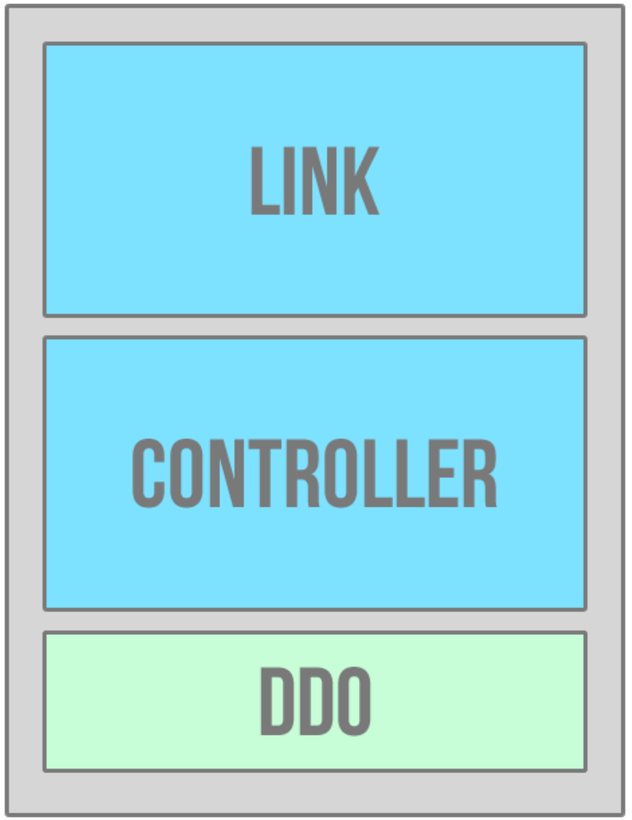
THE DIRECTIVE BREAKDOWN

Directives usually consist of a **definition object (DDO), link** function**,** and a **controller** function

But there are other properties as well

Having a firm grasp on these three things will make advanced directives much more approachable


LINK
CONTROLLER
DDO


LINK
CONTROLLER
DDO

DIRECTIVE DEFINITION OBJECT

The Directive Definition Object (DDO) tells the compiler how a directive is supposed to be assembled

```
ngModule.directive('sayHello', function() {
   return {
      /* this object is the DDO and
         it contains all kinds of data */
   };
});
```

DIRECTIVE DEFINITION OBJECT

Common properties are the **link** function, **controller** function, **restrict**, **template** and **templateUrl**

```
ngModule.directive('sayHello', function() {
   return {
      link : function() { … },
      controller : function() { … },
      restrict: bool,
      templateUrl : 'remote-file.html'
   };
});
```

RESTRICT

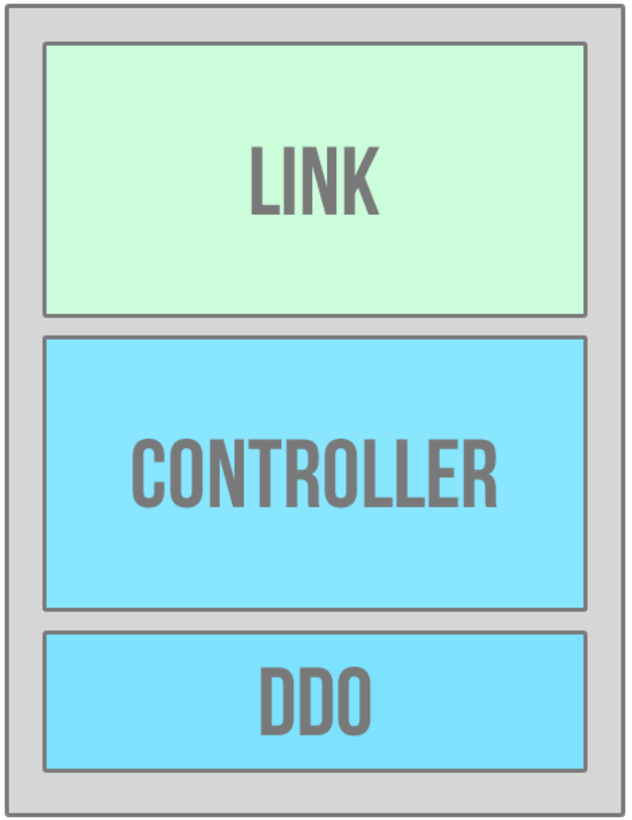
Restricts the definition of the directive
  E = Element, C = Class, M = Comment, A = Attribute
  If restrict is not provided then **A** is used by default

```
// <say-hello>Click here to say hello</say-hello>
ngModule.directive('sayHello', function() {
   return {
      restrict: 'E',
      link : function() {
         /* same DOM code as before */ },
   };
});
```

TEMPLATE URL

Replaces the current contents of the DOM element with the content of the template loaded from the templateUrl

```
// <div say-hello>… REMOTE TEMPLATE HTML CODE …</div>
ngModule.directive('sayHello', function() {
  return {
    templateUrl : 'remote-template.html',
    link : function() {
      /* same DOM code as before */ },
  };
});
```


LINK
CONTROLLER
DDO

LINK FUNCTION

The link function is where DOM manipulation happens

The link function comes with **scope**, **element** and **attars**

scope is the the same **$scope** in the controller function

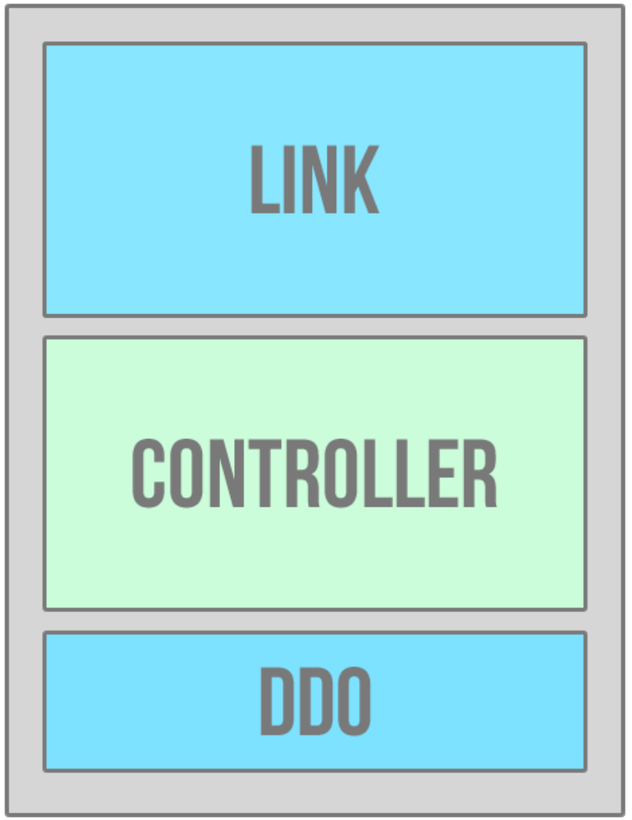
attrs is a list of attributes declared on the element

If a function is returned instead of a DDO then Angular treats that function the same as the link function

LINK FUNCTION

```
ngModule.directive(‘sayHello’, function() {
   return function() {
      //say hello code
   }
});

ngModule.directive(‘sayHello’, function() {
   return {
      link : function() {
         //say hello code
      }
   };
});
```


LINK
CONTROLLER
DDO

CONTROLLER FUNCTION

Directives can also have a controller function.

```
ngModule.directive('sayHello', function() {
   return {
      controller : ['$scope', function($scope) {
         $scope.sayHello = function() {
            alert('Hello');
         };
      }]
   };
});
```

CONTROLLER FUNCTION

We can also use the **controllerAs** feature with directives:

```
ngModule.directive('sayHello', function() {
   return {
      controllerAs: 'hello',
      controller : function() {
         this.say = function() { alert('Hello'); };
      }
   };
});
```

DIRECTIVE TRANSCLUSION

Using something called transclusion we can wrap the inner content of a directive with our own template code.

We can go from this:

```
<greeting-message>John</greeting-message>
```

To this:

```
<greeting-message>Hello John, nice to meet you</greeting-message>
```

DIRECTIVE TRANSCLUSION

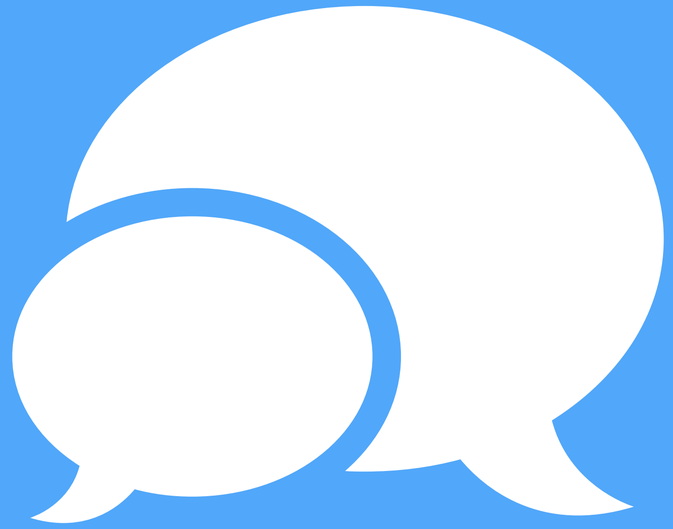
Using something called transclusion we can wrap the inner content of a directive with our own template code.

```
ngModule.directive('greetingMessage', function() {
  return {
    restrict: 'E',
    transclude: true,
    template: 'Hello <ng-transclude></ng-transclude>,
nice to meet you'
  }
});
```

DIRECTIVE TRANSCLUSION

We can also place the template code into a remote file and use templateUrl:

```
ngModule.directive('greetingMessage', function() {
   return {
      restrict: 'E',
      transclude: true,
      templateUrl: 'greeting-template.html'
   }
});
```



REVIEW

BONUS: ANIMATIONS

AGENDA

Animations
Build Your First Animation
Exercise
The Animation Naming Convention
CSS Transitions
Exercise
CSS Animations
Exercise
JavaScript Animations
Exercise
Review

ANGULARJS ANIMATIONS

Include **ngAnimate** as an application sub-module

Does not actually do the animations but provides an API to integrate them however you choose via CSS or JavaScript

Three types of animations: **CSS Transitions**, **CSS Animations** and **JavaScript**

Uses CSS classes to figure out what to animate

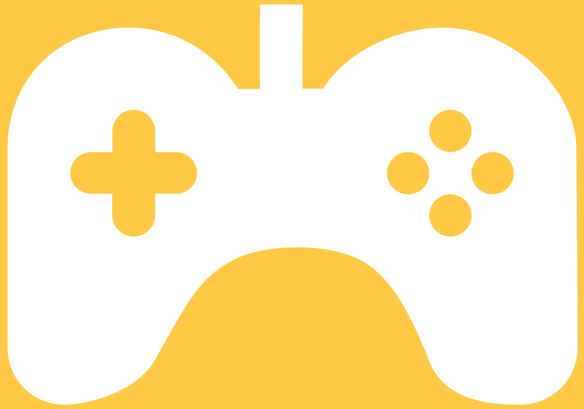
THE NG-ANIMATE MODULE

Remember to include **angular-animate.js** into your app

And use **ngAnimate** as a module dependency

Otherwise none of the animations will trigger.

```
var ngModule = angular.module('myApp', ['ngAnimate']);
```



YOUR FIRST ANIMATION

THE ANIMATION NAMING CONVENTION

The pattern is **[class][event][destination]** and stack on each other

```
/* .my-animation.CLASS-add.CLASS-add-active */
.repeat-item.ng-enter.ng-enter-active { }

/* .my-animation.CLASS */
.my-elm.ng-hide { }
```

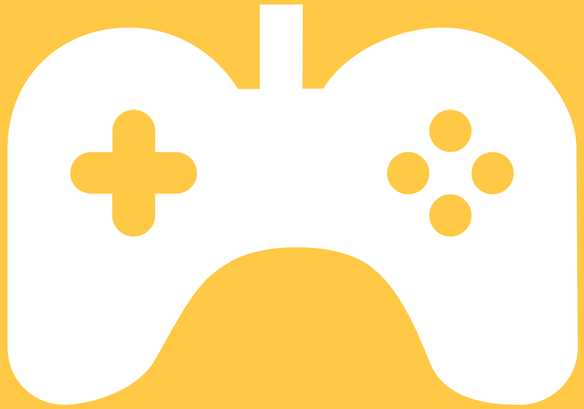
CSS TRANSITIONS

The easiest and fastest way to attach animations

Support by every browser except for IE9 and below

As long as the matching CSS class is present then **AngularJS** will pick up the animation

```
<div class="my-class" ng-if="bool">…</div>
.my-class.ng-enter { transition:1s linear all; opacity:0; }
.my-class.ng-enter.ng-enter-active { opacity:1; }
.my-class.ng-leave { transition:1s linear all; opacity:1; }
.my-class.ng-leave.ng-leave-active { opacity:0; }
```



CSS TRANSITIONS

CSS ANIMATIONS

More extensive than transitions

Supported by every browser except IE9 and below

Does not require the destination class styling aka **-active** class

```
<div class="my-class" ng-if="bool">…</div>
.my-class.ng-enter {
    animation: enter_animation 1s;   -webkit-animation: enter_animation 1s;
}
.my-class.ng-leave {
    animation: leave_animation 1s;   -webkit-animation: leave_animation 1s;
}
```

CSS ANIMATIONS

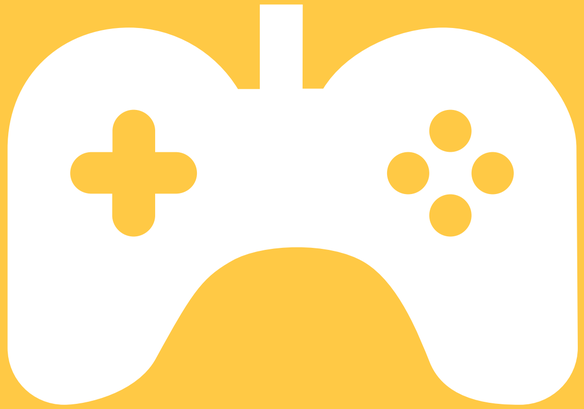
CSS Animations invoke the @keyframes declaration

```
@keyframes enter_animation {
    from { opacity:0; }
    to { opacity:1; }
}

@keyframes leave_animation {
    from { opacity:1; }
    to { opacity:0; }
}
```

```
@-webkit-keyframes enter_animation {
        from { opacity: 0; }
        to { opacity: 1; }
}

@-webkit-keyframes leave_animation {
        from { opacity: 1; }
        to { opacity: 0; }
}
```



CSS ANIMATIONS

JAVASCRIPT ANIMATIONS

Allows for more control over your animations

Define your animations with the **animation** service

Naming convention is still class based

Make sure to call the **done()** function when the animation is over

THE $ANIMATE SERVICE

If you use **$animate** in your own directive code then you can trigger animations when DOM operations are being rendered during the life cycle of your directive.

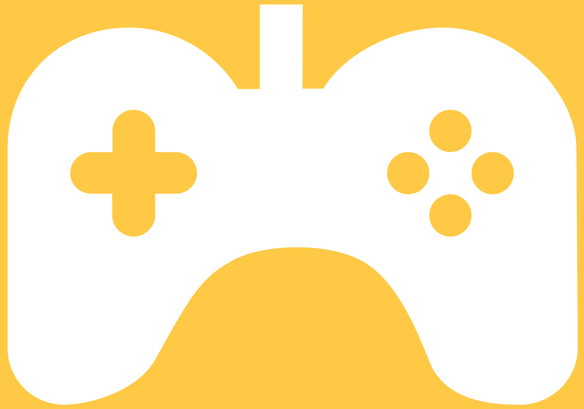
```
ngModule.directive('.my-class', ['$animate', function($animate) {
    return function(scope, element, attrs) {
        element.on('click', function() {
            $scope.$apply(function() {
                $animate.addClass(element, 'active');
            });
        });
    }
}]);
```

THE $ANIMATE SERVICE

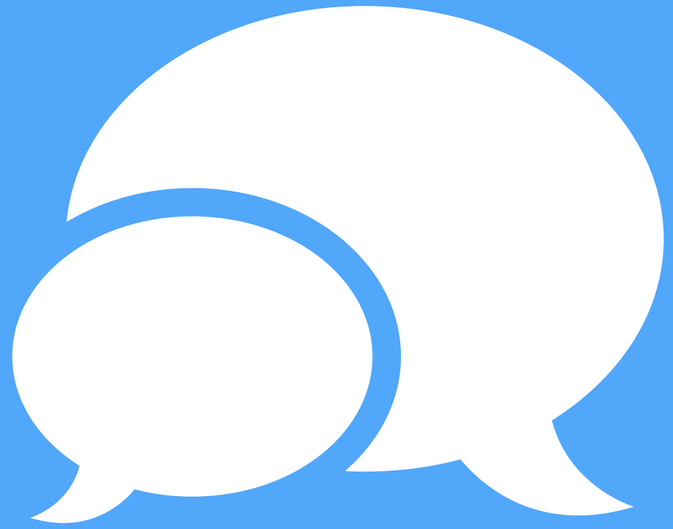
Each of these methods will trigger an animation on the provided element

- $animate.enter(element, parent, after);
- $animate.move(element, parent, after);
- $animate.leave(element);
- $animate.addClass(element, className);
- $animate.removeClass(element, className);
- $animate.setClass(element, classesToAdd, classesToRemove);

Remember, $animate will not trigger any animations unless the ngAnimate module is added.



JAVASCRIPT ANIMATIONS



REVIEW

WHERE TO GO FROM HERE

There's a lot to learn from AngularJS. This course went over just the basics.

Be sure to visit these websites to find out more stuff:

- http://www.yearofmoo.com/
- http://onehungrymind.com/
- http://ng-newsletter.com/
- http://toddmotto.com/
- http://egghead.io/

STAY IN TOUCH!

Name: **Matias Niemelä**
Email: **matias@yearofmoo.com**
Twitter: **@yearofmoo**
Website: www.yearofmoo.com

Name: **Lukas Ruebbelke (not here)**
Email: **simpul@gmail.com**
Twitter: **@simpulton**
Website: http://onehungrymind.com/