



# ANGULAR JS

Getting started

# A BIT OF HISTORY

Angular was developed by **Misko Hevery** and **Adam Abrons** in **2009**.

Angular is currently maintained by Google.

Licensed under the MIT license.

# WHAT IS ANGULAR JS?

Angular JS is a framework that aims to extend HTML and allow describing dynamic applications in a declarative manner

# VIEWS

In Angular JS dynamic HTML views are created using three key components: **expressions**, **directives** and **data bindings**

# EXPRESSIONS

Expressions are JS-like strings that are compiled by Angular and displayed to the user

```
1 + 1 = {{1+1}}  
Hello, {{“John”}}  
{{ true ? “True” : “False” }}
```



# DIRECTIVES

Directives allow to define new HTML elements and alter the behaviour of existing elements using custom attributes

```
<main-menu></main-menu>  
<div ng-show="false">Hello!</div>
```

# BINDINGS

Bindings are used to store values in memory and display them to the user

```
<input type="text" ng-model="name" />  
<div>  
    Hello, {{name}}!  
</div>
```

# SIMPLE EXAMPLE

```
<html>
  <head>
    ...
  </head>

  <body ng-app="myApp">
    <div ng-controller="MainController">
      <input type="text" ng-model="name" />
      <div>Hello, {{name}}!</div>
    </div>
  </body>
</html>
```



# CONTROLLERS

Controllers are used to define and populate a new scope in the view

```
<div ng-controller="MainController">  
  <input type="text" ng-model="name"  
    ng-change="onNameChange()" />  
  <div>Hello, {{name}}!</div>  
</div>
```

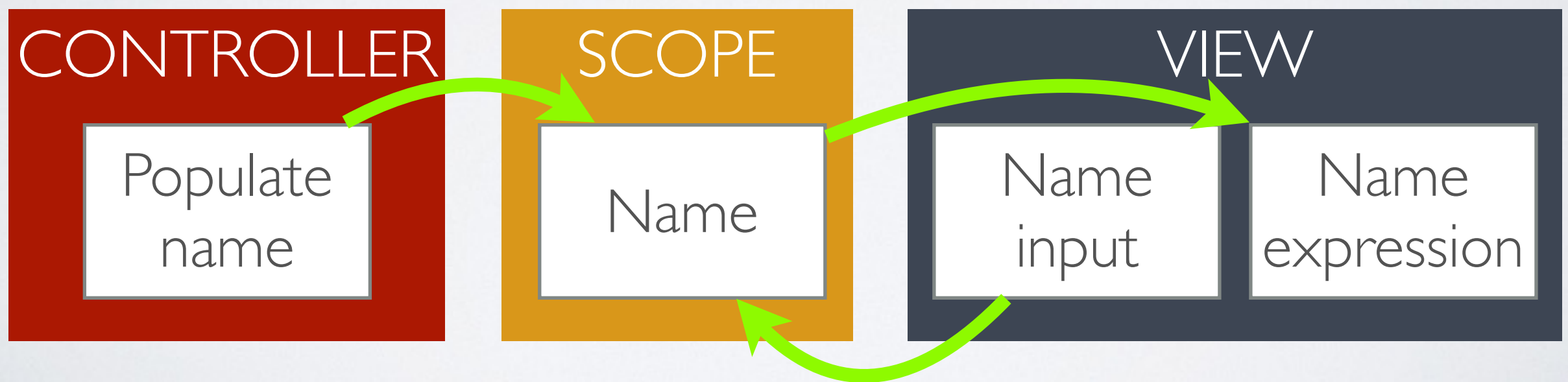
# SCOPE

Scope is an in-memory object that stores the values of all of the bindings used in the view. They can contain both values and methods.



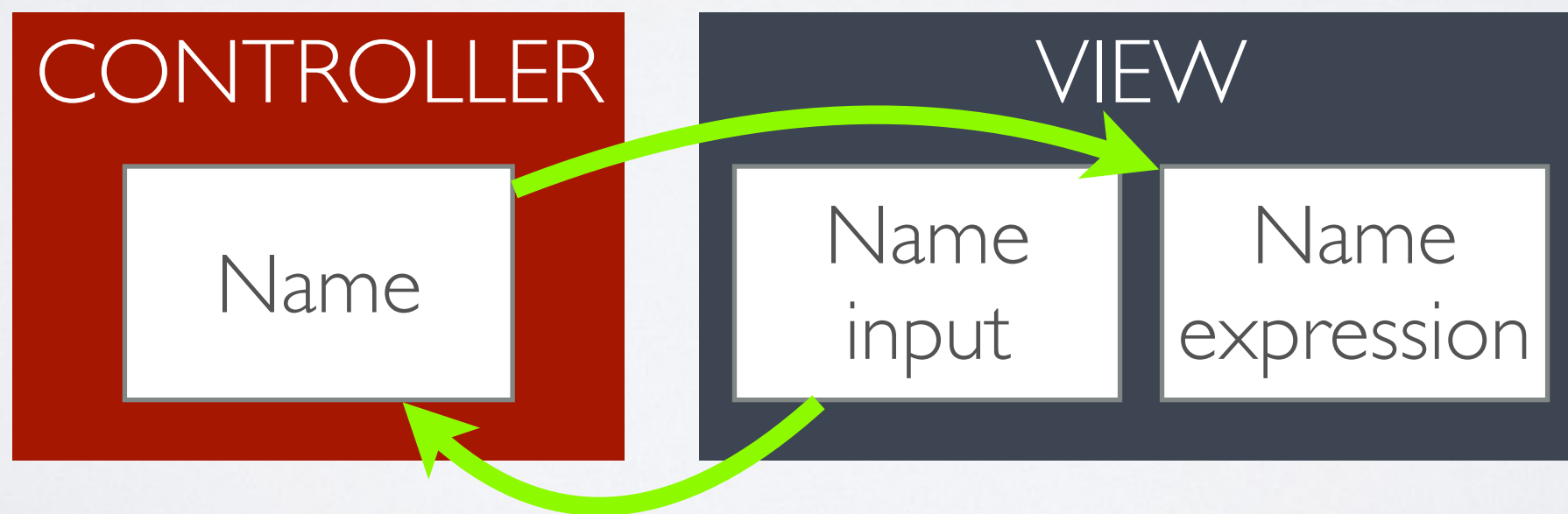
# CONTROLLERS

Controllers can change the view by manipulating values in the scope



# BINDING TO CONTROLLERS

Angular 1.3 introduced the ability to bind directly to the controller, without having to explicitly deal with the scope





# BEHIND THE CURTAINS





# ANGULAR API

Angular JS exposes one global variable - angular.

angular

# MODULES

Modules are the top-level blocks of Angular architecture. They contain controllers as well as other objects like services and directives.

```
angular.module('myApp', []);
```

# CONTROLLERS

Controllers are defined using constructor functions

```
angular.module('myApp')  
  .controller('MainController',  
    function($scope) {  
      $scope.name = 'Mr. Black';  
      $scope.onNameChange = function() {  
        ...  
      }  
    }  
  )
```

# DIRECTIVES

Directives are factory functions that return its definition

```
angular.module('myApp')  
  .directive('greeting', function() {  
    return {  
      scope: {  
        name: '='  
      },  
      template: '<div>Hello, {{name}}!</div>'  
    }  
  })
```

# SEPARATION OF CONCERNS

Controllers are responsible for business and application logic.

Directives are used for DOM manipulation.

**Never work with the DOM inside of controllers.**



# FILTERS

Filters are functions that can be used to transform values before outputting them to the user

```
angular.module('myApp')  
  .filter('uppercase', function() {  
    return function(input) {  
      return input.toUpperCase();  
    }  
  })  
  
{{"John" | uppercase}} // will display "JOHN"
```

# SERVICES

Services are reusable objects that contain business and application logic

```
angular.module('myApp')  
  .service('greeter', function() {  
    return {  
      greet: function(name) {  
        alert('Hello, ' + name + '!');  
      }  
    }  
  })
```

# OTHER STUFF IN A MODULE

Module can also be used to define **constants**, **values**, **animations**, **decorators** and other types of components



# WHAT ELSE



# DEPENDENCY INJECTION

Angular DI can automatically injects the required components based on its definition

```
angular.module('myApp')  
  .controller('MainController',  
    function($scope, $greeter) {  
      $greeter.greet();  
    }  
  )
```



# AJAX

Angular provides two services for implementing AJAX:

**\$http** - for generic XML HTTP requests;

**\$resource** - for RESTful APIs

# GLOBAL OBJECT WRAPPERS

Angular provides wrappers for global JS objects:

**\$document**, **\$location**, **\$window**.

Its **\$timeout** and **\$interval** services can be used instead of `setTimeout()` and `setInterval()`.

# ROUTING

Routing can be implemented using the **ng-route** plugin.

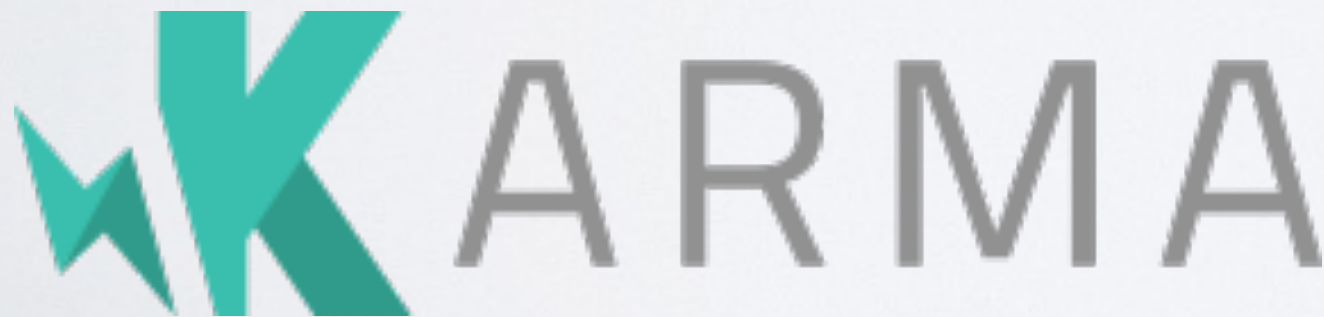
A more advanced **ui-router** plugin adds convenient nested route support.

A backport of the router from Angular 2.0 is available in the **angular-new-router** plugin.

# UNIT TESTS

Angular is built with testability in mind and all of its components can be covered with unit tests.

Angular advises to use **jasmine** for writing unit test and **karma** for launching them.





# FUNCTIONAL TESTING

The **protractor** framework is especially created for testing Angular applications and can serve as an alternative to Selenium.



# Protractor

end to end testing for AngularJS



# BUILD TOOLS

There are Angular-specific plugins available for both Grunt and Gulp



WHAT'S NEXT?

**WHAT'S**  
**NEXT STEP** **your** **?**



# DON'T STOP LEARNING

## JavaScript Learning Curves



**jQuery**



**Node.js**



**AngularJS**

# ALTHOUGH IT WILL BE A BUMPY RIDE



# COMMON SOURCE OF PROBLEMS

Over using data bindings can easily lead to poor performance.

The data flow needs to be carefully thought out, otherwise it will be hard to follow the logic of the code

# ANGULAR 2.0 IS COMING





# ANGULAR 2.0

Some of the features of Angular 2.0 include:

- Type script and ES6 support;
- Improved performance
- Server side rendering
- Building for mobile platforms

# LETS DIVE IN

