

# Web Application Development

---

Produced  
by

David Drohan ([ddrohan@wit.ie](mailto:ddrohan@wit.ie))

Department of Computing & Mathematics  
Waterford Institute of Technology

<http://www.wit.ie>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE





# ANGULARJS

PART 3

---

FILTERS, SERVICES & DIRECTIVES



# Section Outline

---

1. **Introduction** – Why you should be using AngularJS
2. **Terminology** – The critical foundation for understanding
3. **Modules** – Reusable functionality
4. **Views** – UI (User Interaction)
5. **Controllers** – Facilitating communication between the model and the view
6. **Routes** – Navigating the view
7. **Filters** – Changing the way you see things
8. **Services** – Five recipe flavors
9. **Directives** – Extending HTML
10. **Case Study** – Labs in action
11. **Conclusions** – The end is nigh



# Section Outline

---

1. **Introduction** – Why you should be using AngularJS
2. **Terminology** – The critical foundation for understanding
3. **Modules** – Reusable functionality
4. **Views** – UI (User Interaction)
5. **Controllers** – Facilitating communication between the model and the view
6. **Routes** – Navigating the view
7. **Filters** – Changing the way you see things
8. **Services** – Five recipe flavors
9. **Directives** – Extending HTML
10. **Case Study** – Labs in action
11. **Conclusions** – The end is nigh

# Basic Building Blocks

---

WHAT YOU NEED TO BUILD A BASIC ANGULAR WEB APP

# Filters

---

CHANGING THE WAY YOU SEE THINGS

# Filters

---

What is a filter:

- A filter formats the value of an expression for display to the user.
  - (<https://docs.angularjs.org/guide/filter>)

Filters can be used in HTML using the bar notation or they can be used in JavaScript by injecting the **\$filter** service.

# Filters

---

## HTML EXAMPLE

```
<div ng-app='myApp' ng-controller="myController">
  <p>{{name | uppercase}}</p>
  <p>{{uppercaseName()}}</p>
</div>
```

## JAVASCRIPT EXAMPLE

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

module.controller('myController', [
  '$scope', '$filter', function($scope, $filter) {

    $scope.name = 'John Smith';

    $scope.uppercaseName = function() {
      return $filter('uppercase')($scope.name);
    };

  }]);
```



# Filters with Parameters

---

## HTML EXAMPLE

```
{{ expression | filterName : param1 : param2 }}
```

## JAVASCRIPT EXAMPLE

```
$filter('filterName')(expression, param1, param2);
```

# Core Filters

---

AngularJS has several filters built in:

- currency
- date
- filter
- json
- limitTo
- lowercase
- number
- orderBy
- uppercase

```
{{ '2015-03-19T19:00:00.000Z' | date : 'MMMM yyyy' }}
```

```
$filter('date')(new Date(), 'MMMM yyyy');
```



# Core Filters

AngularJS has several filters built in:

- currency
- date
- filter
- json
- limitTo
- lowercase
- number
- orderBy
- uppercase

Select a subset of an array for display

```
{{ ['pear', 'apple', 'orange'] | filter : 'r' }}
```

```
$filter('filter')(['pear', 'apple', 'orange'], 'r');
```

# Core Filters

Format in json for  
easy searching

AngularJS has several filters built in:

- currency
- date
- filter
- json
- limitTo
- lowercase
- number
- orderBy
- uppercase

```
{{ { first: 'John', last: 'Smith' } | json }}
```

```
$filter('json')({ first: 'John', last: 'Smith'});
```

# Core Filters – More Examples

---

<code>{{ 'hello'   uppercase }}</code>	→	HELLO
<code>{{ 2000   currency }}</code>	→	\$2,000.00
<code>{{ 2500   currency:'EUR€' }}</code>	→	EUR€2,500.00
<code>{{ 1288323623006   date }}</code>	→	Oct 29, 2010
<code>{{ 1288323623006   date:'dd/MM/yyyy' }}</code>	→	29/10/2010
<code>{{ 1288323623006   date:'EEE dd/MM/yy' }}</code>	→	Fri 29/10/10

# Core Filters

---

Filters can be chained/stacked (where sensible)

1. *customers | orderBy:'-balance' | limitTo:2* --- only show the two richest customers
2. *customers | filter:{name:'ki'} | orderBy:'-balance' | limitTo:5*  
the five richest customers whose name contains the substring 'ki'.

# Core Filters – Our Example

```
<div ng-controller="donationsController as list">
<table align="center" valign="middle">
  <tbody ng-repeat="donation in donations | orderBy:'-upvotes'">
    <tr style="height:55px; font-size:20px; margin-left:20px; margin-right:20px; valign="middle"; >
      <td>
        <span class="glyphicon glyphicon-thumbs-up" ng-click="incrementUpvotes(donation)">
          {{donation.upvotes}}
        </td>
      <td>
        <span style="height:55px; font-size:20px; margin-left:20px; margin-right:20px;">
          {{donation.paymenttype}}
        </span>
      </td>
      <td>
        <span class="glyphicon glyphicon-euro"></span>
          {{donation.amount}}
        </td>
      <td>
```

## List All Donations

Donations Page!

👍 4	Direct	€ 1002	✖ Remove	✎ Edit
👍 3	Direct	€ 998	✖ Remove	✎ Edit
👍 1	PayPal	€ 997	✖ Remove	✎ Edit
👍 1	Direct	€ 99981	✖ Remove	✎ Edit
👍 0	PayPal	€ 1000	✖ Remove	✎ Edit

# Filter Definition

You can define a new filter within a module.

```
angular.module('reverseFilter', [])  
  .filter('reverse', [function() {  
    return function(input, uppercase) {  
      var i, out = "";  
      input = input || '';  
      for (i = 0; i < input.length; i++) {  
        out = input.charAt(i) + out;  
      }  
      // conditional based on optional argument  
      if (uppercase) {  
        out = out.toUpperCase();  
      }  
      return out;  
    };  
  }]);
```

```
{{ 'Hello, World!' | reverse : true }}
```

```
$filter('reverse')('Hello, World!', true);
```



# Services

---

FIVE RECIPE FLAVORS

# AngularJS Services

---

**Services** provide a method for us to keep data around for the lifetime of the app and communicate across controllers in a consistent manner.

Services are singletons, which are objects that are instantiated only once per app (by the \$injector). They provide an interface to keep together methods that relate to a specific function.

Angular comes with several built-in services, of different types (*or recipes*) which we'll cover in the next few slides.

You can also make your *own* services for any decently complex application.

# Value

---

The value recipe stores a value within an injectable service.

A value can store any service type: a string, a number, a function, and object, etc.

This value of this service can now be injected into any controller, filter, or service.

```
//define a module
var myModule = angular.module('myModule', []);

//define a value
myModule.value('clientId', 'a12345654321x');

//define a controller that injects the value
myModule.controller('myController', ['$scope', 'clientId', function ($scope, clientId) {
    $scope.clientId = clientId;
}]);
```

# Factory

---

The factory recipe is similar to a value recipe except that it adds these abilities:

- Ability to use dependency injection
- Lazy initialization

A factory (like a value) can also return any data type.

```
//define a factory
myModule.factory('toUpperCase', ['$filter', function($filter) {

    return function(value) {
        return $filter('uppercase')(value);
    }

}]);

//inject the toUpperCase factory
myModule.controller('myController', ['toUpperCase', function(toUpperCase) {
    this.uppercase = toUpperCase('john');
}]);
```



# Service

---

Just to make things a little confusing, we have a service recipe called service.

- Yes, we have called one of our service recipes 'Service'. We regret this and know that we'll be somehow punished for our misdeed. It's like we named one of our offspring 'Child'. Boy, that would mess with the teachers.  
(<https://docs.angularjs.org/guide/providers>)

# Service

---

The service recipe will generate a singleton of an instantiated object.

```
//define a service
myModule.service('person', [function() {
  this.first = 'John';

  this.last = 'Jones';

  this.name = function() {
    return this.first + ' ' + this.last;
  };
}]);

//inject the person service
myModule.controller('myController', ['$scope', 'person', function($scope, person) {
  $scope.name = person.name();
}]);
```

# Provider

---

“...the Provider recipe is the core recipe type and all the other recipe types are just syntactic sugar on top of it. It is the most verbose recipe with the most abilities, but for most services it's overkill”

(<https://docs.angularjs.org/guide/providers>).

The provider recipe can be injected during a module's configuration phase.



# Provider

```
//define a provider
myModule.provider('prefix', [function() {
  var prefix = '';

  //setPrefix can be called during the module's config phase
  this.setPrefix = function(value) {
    prefix = value;
  };

  //this special property is required and returns a factory
  this.$get = [function() {
    return function(value) {
      return prefix + value;
    }
  }];
}]);

//inject the provider in the config phase
myModule.config(['prefixProvider', function(prefixProvider) {
  prefixProvider.setPrefix('John ');
}]);

//inject the provider's factory
myModule.controller('myController', ['prefix', function(prefix) {
  this.value = prefix('Smith'); //value is set to "John Smith"
}]);
```



# Constant

---

The constant recipe is similar to the value recipe except that its service value is also available during the module's configuration phase.

```
myModule.constant('author', 'John Smith');

myModule.config(['author', function(author) {
  console.log('This app was made by: ' + author); // "John Smith"
}]);

myModule.controller('myController', ["author", function (author) {
  this.author = author; // "John Smith"
}]);
```

# Directives

---

EXTENDING HTML

# Directives

---

## What is it?

- At a high level, directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler (\$compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children.
- (<https://docs.angularjs.org/guide/directive>)

```
<form>
  <p>
    Enter your name:
    <input type="text" ng-model="name" required>
    <button type="button" ng-click="submit()" ng-disabled="!name">Submit</button>
  </p>
  <p ng-show="name">
    Hello, {{name}}
  </p>
</form>
```



# Directives

---

Directives **extend** HTML to suit the needs of dynamic web app (SPA) development

- ng-app
- ng-model
- ng-repeat
- ng-controller
- ng-init
- etc

Custom directives also supported

**Recall our 2-way data binding**



# Directive Documentation (for writing your own!)

---

For full documentation on how to write directives, see these pages:

<https://docs.angularjs.org/guide/directive>

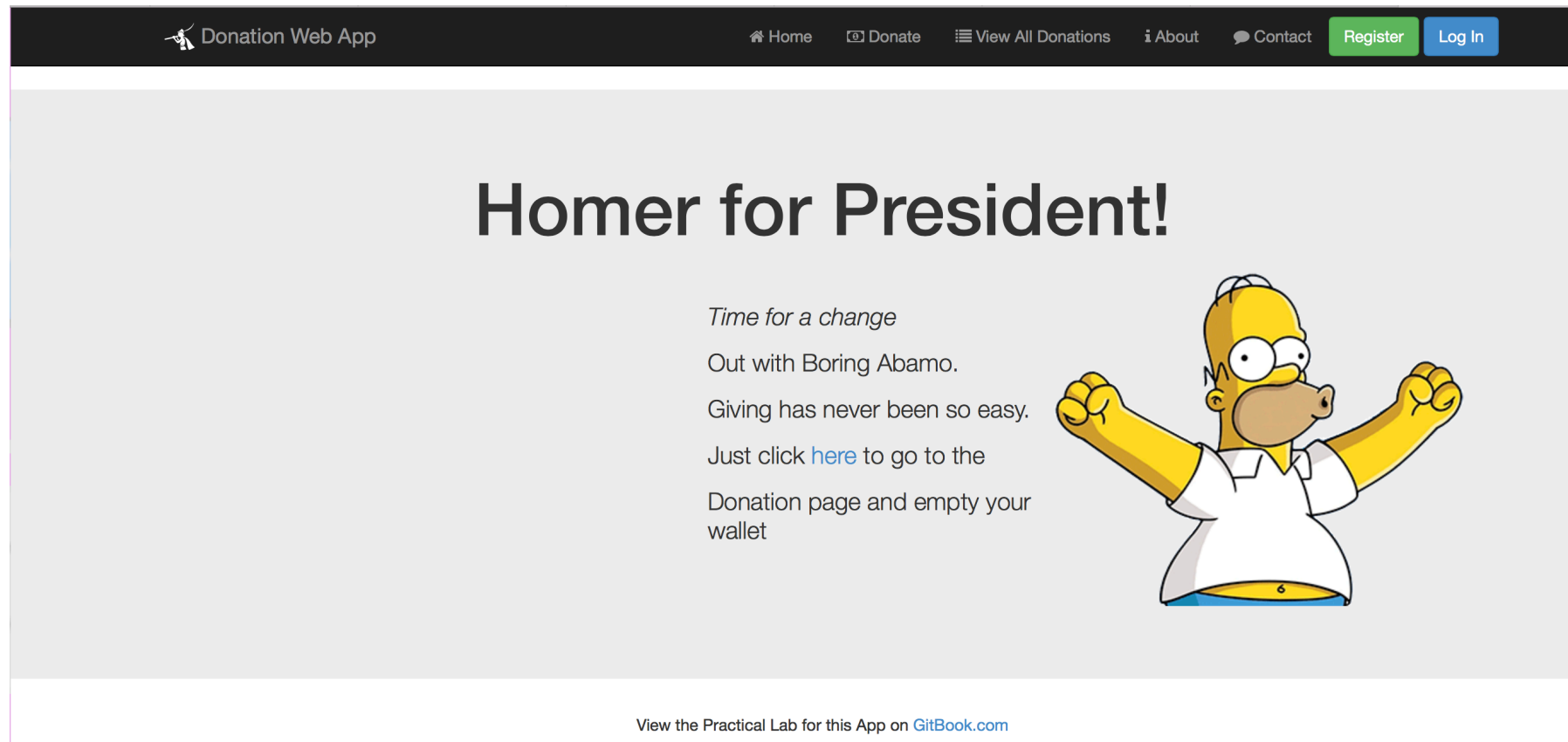
[https://docs.angularjs.org/api/ng/service/\\$compile](https://docs.angularjs.org/api/ng/service/$compile)

# Case Study

---

LABS IN ACTION

# Demo Application



The screenshot shows a web application interface. At the top is a dark navigation bar with the title 'Donation Web App' on the left and several menu items on the right: 'Home', 'Donate', 'View All Donations', 'About', 'Contact', 'Register', and 'Log In'. The 'Register' and 'Log In' buttons are highlighted in green and blue respectively. Below the navigation bar is a large light gray area containing the main content. The main heading is 'Homer for President!' in a large, bold, black font. Below the heading is a block of text: 'Time for a change', 'Out with Boring Abamo.', 'Giving has never been so easy.', 'Just click [here](#) to go to the', 'Donation page and empty your wallet'. To the right of this text is a cartoon illustration of Homer Simpson with his arms raised in a celebratory gesture. At the bottom of the screenshot, there is a small link: 'View the Practical Lab for this App on [GitBook.com](#)'.

# Conclusions

---

THE END IS NIGH



# Essentials for Building Your First App

---

## HTML DOCUMENT

```
<!doctype html>
<html lang="en" ng-app="app">
  <head>
    <title>App Title</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <script src="js/angular.js"></script>
    <script src="js/app.js"></script>
    <script src="js/moduleA.js"></script>
  </body>
</html>
```

## APP.JS

```
var module = angular.module('app', ['moduleA']);
```

# Great Resources

---

Official Tutorial – <https://docs.angularjs.org/tutorial>

Official API – <https://docs.angularjs.org/api>

Developer Guide – <https://docs.angularjs.org/guide>

Video Tutorials – <https://egghead.io/>

Video Introduction – <https://www.youtube.com/watch?v=i9MHigUZKEM>

YouTube Channel – <https://www.youtube.com/channel/UCm9iilfgmVODUJxINecHQkA>

Articles, explanations, tutorials – <http://www.ng-newsletter.com/>



# About the Original Author

---

James Speirs

Application Foundations

OIT Core Services

Brigham Young University



---

# Questions?