Dynamic Web Development

Produced by

David Drohan (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 1

INTRODUCTION & TERMINOLOGY

Objectives

To give you a foundation on which you can begin to understand all of the other tutorials and documentation out there.

We will cover…

- Why
- Terminology
- and How

We will also cover how to build your first AngularJS application (mostly in the Labs!)

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

Introduction

WHY YOU SHOULD BE USING ANGULARJS

What is AngularJS and why use it?

- First off, why use JavaScript?
  - To dynamically update your HTML document and to get data to and from it.
- AngularJS is an open-source framework (a JavaScript library) that makes it easier to communicate between your HTML document and JavaScript.
  Used for developing Dynamic Web Apps (SPAs in modern lingo).
  - Developed in 2009 by Miško Hevery and Adam Abrons
  - Maintained by Google
  - Actively developed and supported (GitHub)

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop

Anatomy of an Angular Application

- Controllers
- Models
- Services
- Directives
- Modules
- Filters
- Factories
- Scopes (Models)
- Templates
- Routes

We will cover most of these in some form during the sections on Angular

ANGULARJS

What Makes an AngularJS App?

- Load AngularJS

```
<script src="https://ajax.googleapis.com/ajax/libs/
                angularjs/1.0.6/angular.min.js"></script>
```

- Bootstrap

```
<html ng-app>
```

- Declare relationships
  - `ng-model="yourName"`
  - `{{yourName}} //html`

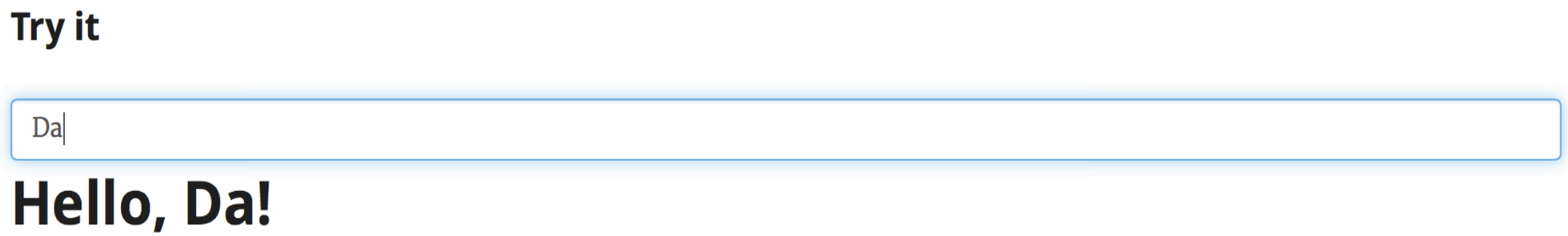
Bootstrapping an Angular App

```
1   <!doctype html>
2   <html ng-app>
3     <head>
4       <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
5     </head>
6     <body>
7       <div>
8         <input type="text" ng-model="yourName" placeholder="Enter a name here">
9         <h1>Hello, {{ yourName }}!</h1>
10      </div>
11    </body>
12  </html>
```

1. Browsers generates ‘document ready’ event when DOM is built.
2. **ng-app** directive causes AngularJS to register a handler for this ‘document ready’ event.
3. Handler amends the DOM based on other directives contained in the HTML.

Bootstrapping an Angular App

Bootstrapping an Angular App



Bootstrapping an Angular App

Try it

Dave

Hello, Dave!

Terminology

THE CRITICAL FOUNDATION FOR UNDERSTANDING

Model View Controller

What is the MVC?

- Model – the data
- View – the user interface, what the user sees and interact with
- Controller – the interface between the model and the view

The model is not necessarily the data from the database

The browser, server, and database can have their own MVC systems and often do

When we talk about the MVC in this presentation we'll be talking about the MVC on client side, in the browser

Data Binding

What is data binding?

- Data-binding in Angular apps is the automatic synchronization of data between the model and view components. (https://docs.angularjs.org/guide/databinding)

From the AngularJS example below we have several data binding instances (marked in red).

```
<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>
```

ANGULARJS

Expressions

What is a JavaScript expression?

- An expression produces a value and can be written wherever a value is expected. (http://www.2ality.com/2012/09/expressions-vs-statements.html)

```
Myvar;
3 + x;
myfunc("a", "b");
```

AngularJS also has expressions:

- Angular expressions are JavaScript-like code snippets that are usually placed in bindings such as
  {{ expression }}. (https://docs.angularjs.org/guide/expression)

```
<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>
```

Directive

What is a directive?

- It is a marker on an HTML element (such as an attribute, element name, comment, or CSS class) that AngularJS recognizes and uses to create a customized set of functionality and/or user interface.

AngularJS comes with many pre-built directives, but you can also write your own custom directives.

```
<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>
```

Controller

What is a controller?

- Controllers link the model and the view using the AngularJS service: **$scope**
- Nesting controllers is both possible and frequently done.

HTML Fragment

```
<div ng-app='myApp' ng-controller="myController">
   <p>Hello, {{name}}!</p>
   <p>{{greet()}}</p>
</div>
```

JavaScript fragment

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

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

      $scope.name = 'John Smith';

      $scope.greet = function() {
         return 'Hello, ' + $scope.name + '!';
      };

   }]);
```

Scope

What is scope?

- It is the context in which an expression is evaluated.

This example has three scopes, one of which inherits a variable from its parent scope.

```
function foo() {
   var name = "John";

   function hello() {
      var name = "Jack";
      return "Hello, " + name;
   }

   function goodbye() {
      return "Good bye, " + name;
   }

   hello();   //returns "Hello, Jack"
   goodbye(); //returns "Good bye, John";
}
```

Modules

What is a module?

- A container for code for the different parts of your applications.

A module is used to define **services** that are reusable by both the HTML document and other modules:

- Controller
- Directive
- Constant, Value
- Factory, Provider, Service
- Filter

Best Practice: Divide your code into modules with distinct functionality. Don't put everything in one module.

Service

What is a service?

- The supplying or supplier of utilities or commodities, as water, electricity, or gas, required or demanded by the public. (http://dictionary.reference.com/browse/service)

In programming we can define a service as…

- The supplying or supplier of utilities or commodities, as **functions**, **values**, or **objects**, required or demanded by **an expression**.

In AngularJS you can define services that provide functionality that can be used repeatedly throughout your code.

Dependency Injection

When one function is dependent on the data or functionality of something else, that something else must be accessible.

OPTION 1: GLOBALLY ACCESSIBILITY

```
var a = 4, b = 2;

function divide() {
   return a / b;
}
```

OPTION 2: USE DEPENDENCY INJECTION

```
function divide(a, b) {
   return a / b;
}
```

Dependency Injection

Dependency injection requires specific data from each parameter. You can't use the divide function like this:

```
function divide(a, b) {
   return a / b;
}

divide('hello', { x: 2 });
```

Routing

What is Routing?

Allows SPAs behave like traditional Web Applications/sites:

1. forward/back button support
2. deep-linking to specific content.
   - Old-style AJAX Web apps did not support routing – the addressability problem.

Advantages:

Bookmarking; Link sharing; Direct navigation.

Two solution approaches:

1. Hash-based, e.g. http://domain_name/#/some_app_url
2. PushState, e.g. http://domain_name/some_app_url

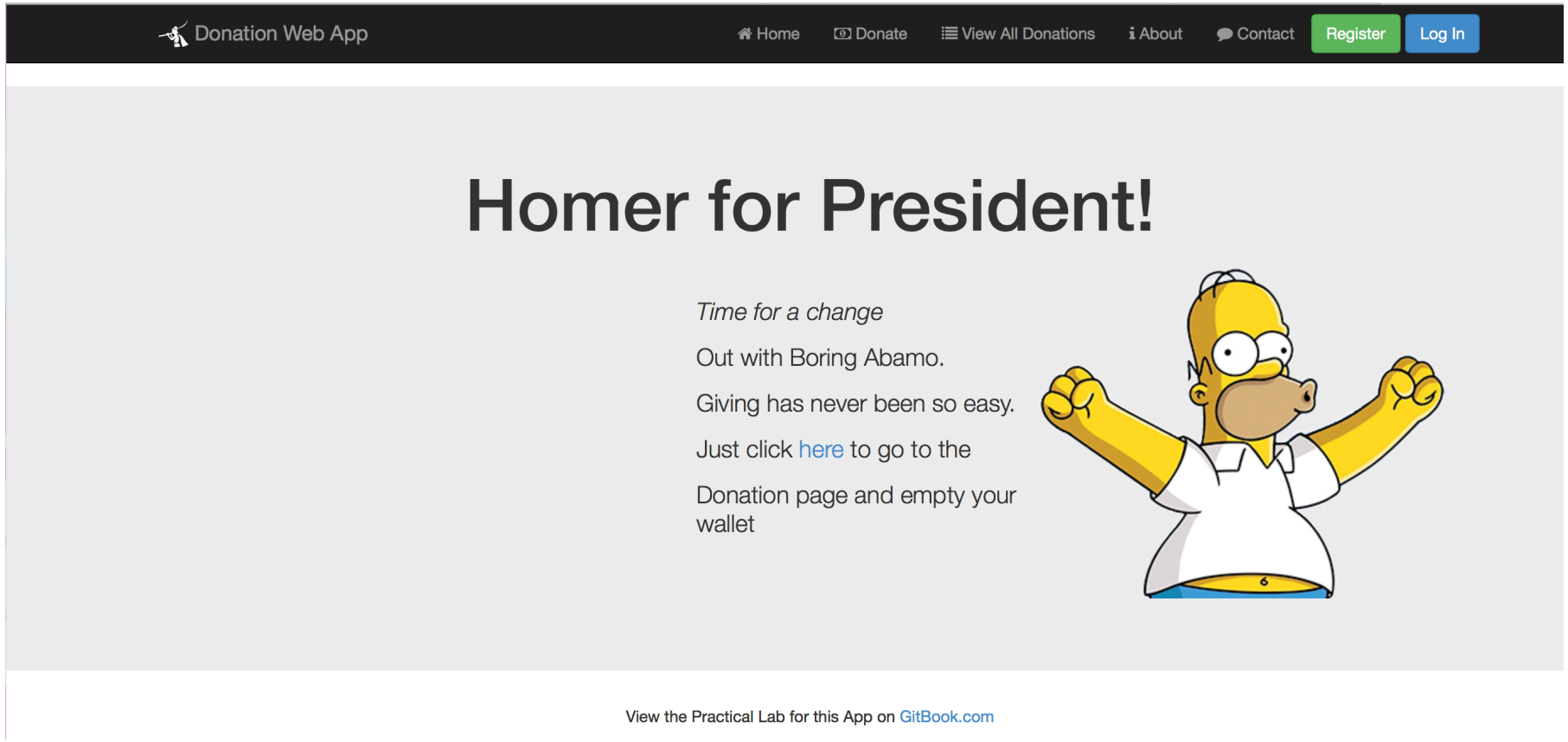
Angular supports both – (1) is default.

Case Study

LABS IN ACTION

Demo Application

http://donationweb-4-0.herokuapp.com



Questions?