Web Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

http://www.wit.ie

ANGULARJS

PART 2

MODULES, VIEWS, CONTROLLERS & ROUTES

Section Outline

Section Outline

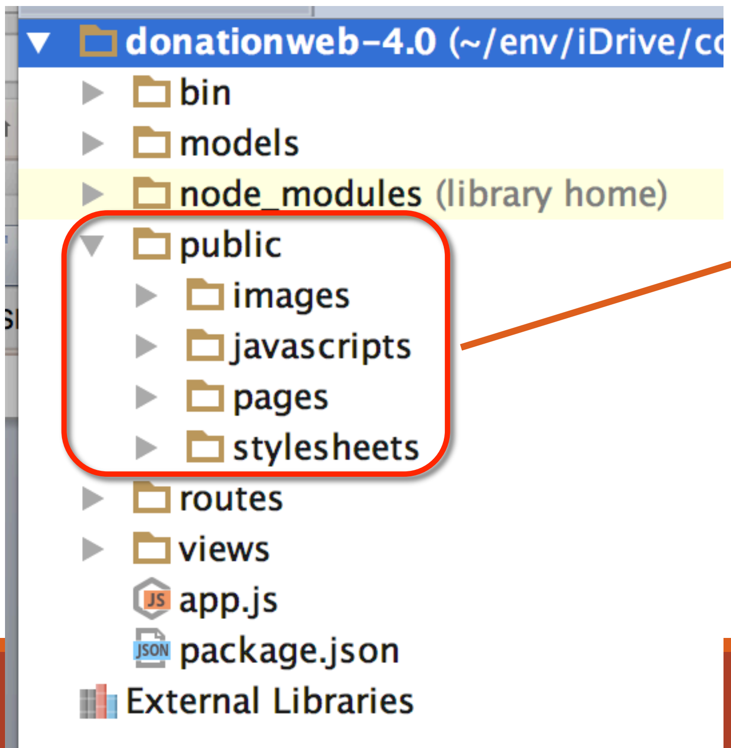
1. **Introduction** – Why you should be using AngularJS
2. **Terminology** – The critical foundation for understanding
3. **Modules** – Reusable functionality
4. **Views** – UI (User Interaction)
5. **Controllers** – Facilitating communication between the model and the view
6. **Routes** – Navigating 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

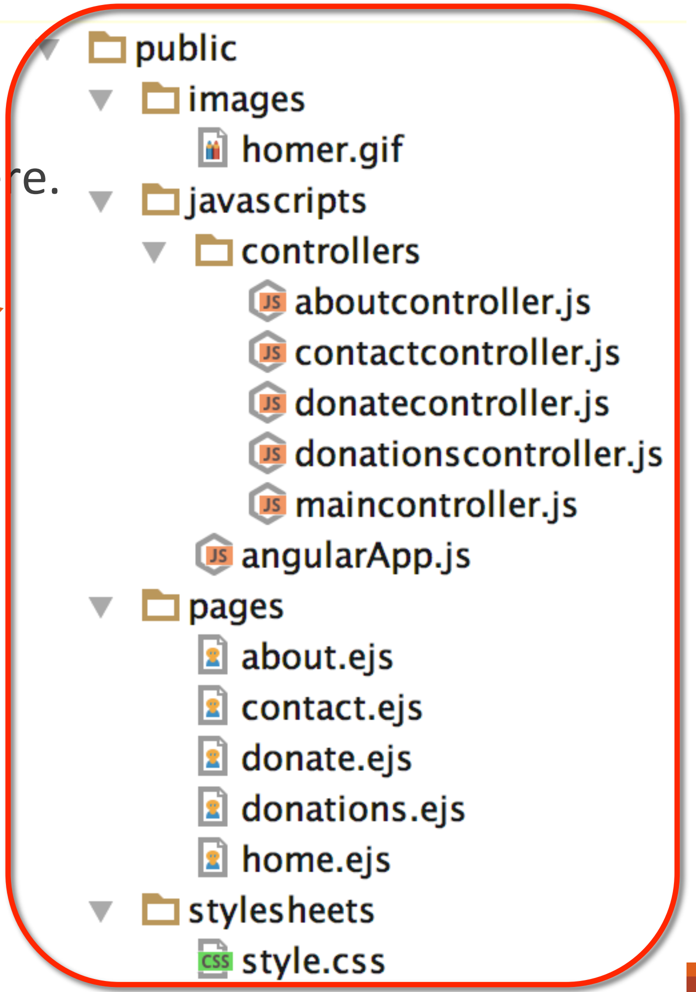
WHAT YOU NEED TO BUILD A BASIC ANGULAR WEB APP

Basic Building Blocks *

- Installing AngularJS is pretty simple. It is just like adding any other library.
- Go to the AngularJS.org website and download the stable version from there.
- You can manage the file directory as:



```
▼ donationweb-4.0 (~/env/iDrive/c
  ▶ bin
  ▶ models
  ▶ node_modules (library home)
  ▼ public
    ▶ images
    ▶ javascripts
    ▶ pages
    ▶ stylesheets
  ▶ routes
  ▶ views
    app.js
    package.json
External Libraries
```



```
▼ public
  ▼ images
      homer.gif
  ▼ javascripts
    ▼ controllers
        aboutcontroller.js
        contactcontroller.js
        donatecontroller.js
        donationscontroller.js
        maincontroller.js
      angularApp.js
  ▼ pages
      about.ejs
      contact.ejs
      donate.ejs
      donations.ejs
      home.ejs
  ▼ stylesheets
      style.css
```

Views

WHAT THE USER INTERACTS WITH

Basic Building Blocks – Views *

- Recall that the **View** is the User Interface, it's what the user interacts with
- When writing an AngularJS app, we write the behavior and interaction together alongside the presentation (the View)
- Views are often referred to as ***templates*** in Angular
- In SPA apps, a rendered template (called a ***partial***) is dynamically inserted into a 'shell page' - The shell's templates changes dynamically over time, hence SPA
- Angular uses ***directives*** to achieve this template insertion
- A ***directive*** is a fancy name for a function that's attached to a DOM element.
  - Directives have the ability to execute methods, define behavior, attach controllers and **$scope** objects, manipulate the DOM, and more.

Basic Building Blocks – Views *

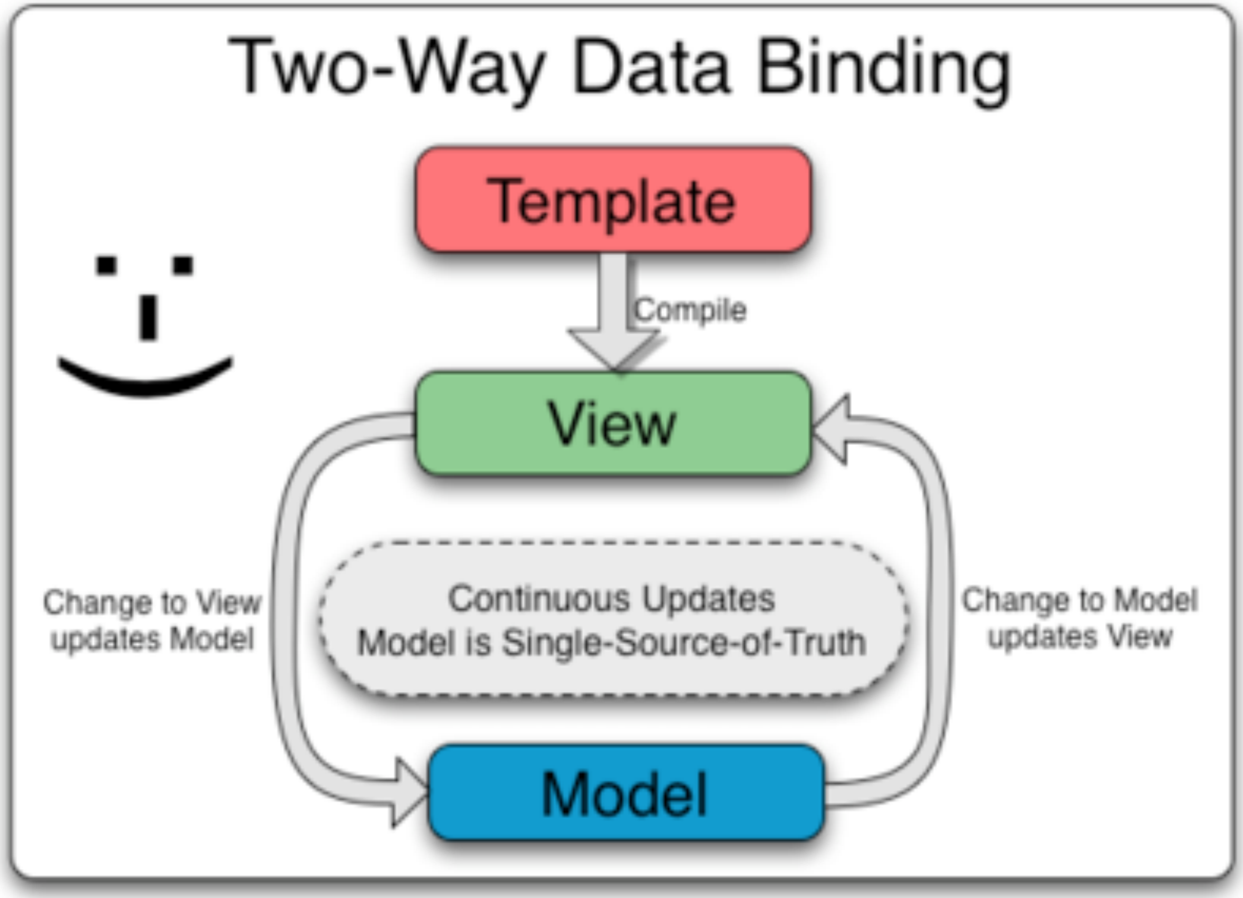
Directives: **ng-**

- A few of the most frequently used are:
  - ✓**ng-app**
    - Determines which part of the page will use AngularJS
    - If given a value it will load that application module
  - ✓**ng-controller**
    - Determines which JavaScript Controller should be used for that part of the page
  - ✓**ng-model**
    - Determines what model the value of an input field will be bound to
    - Used for ***two-way data binding*** (next slide)

2-Way Data Binding

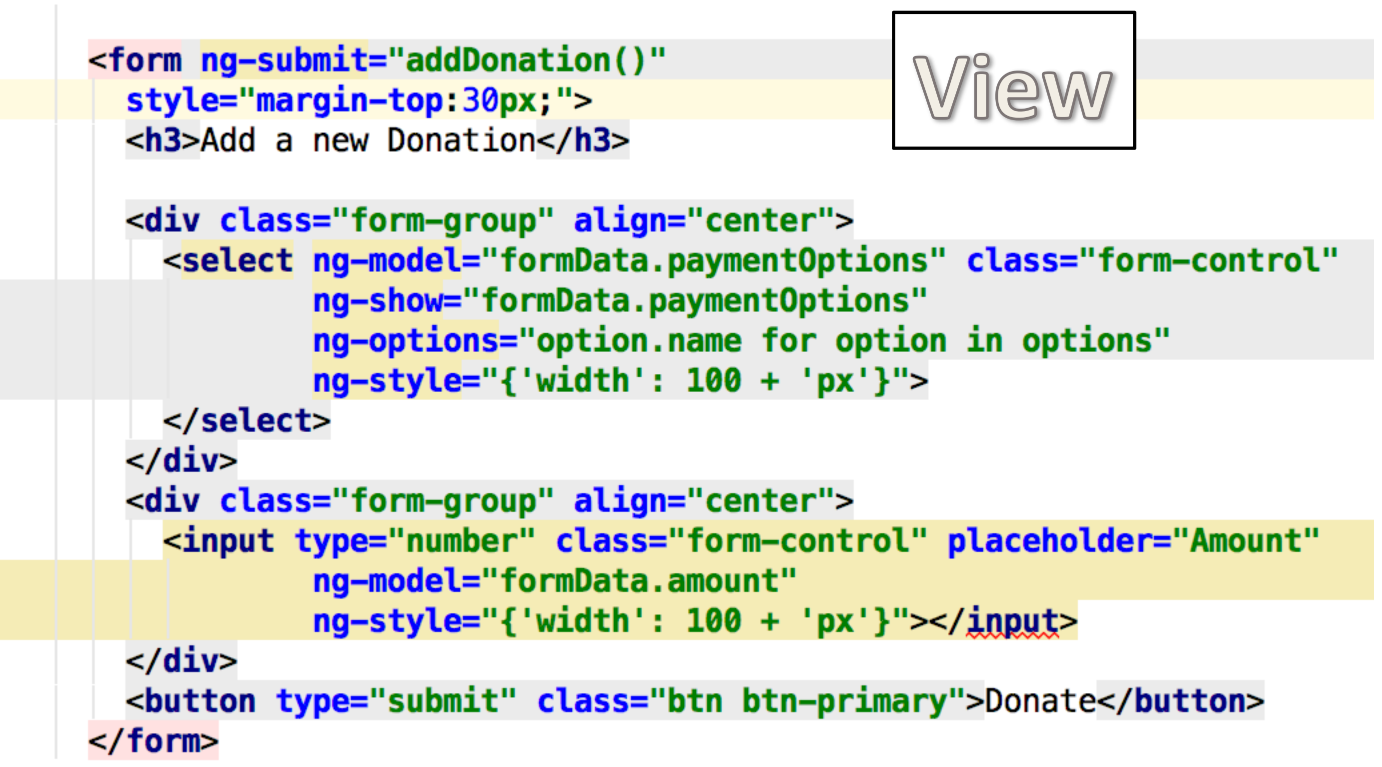
Automatic propagation of data changes

Model is single source of truth


Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates
Model is Single-Source-of-Truth
Change to Model updates View
Model

Two Way Data Binding Example *

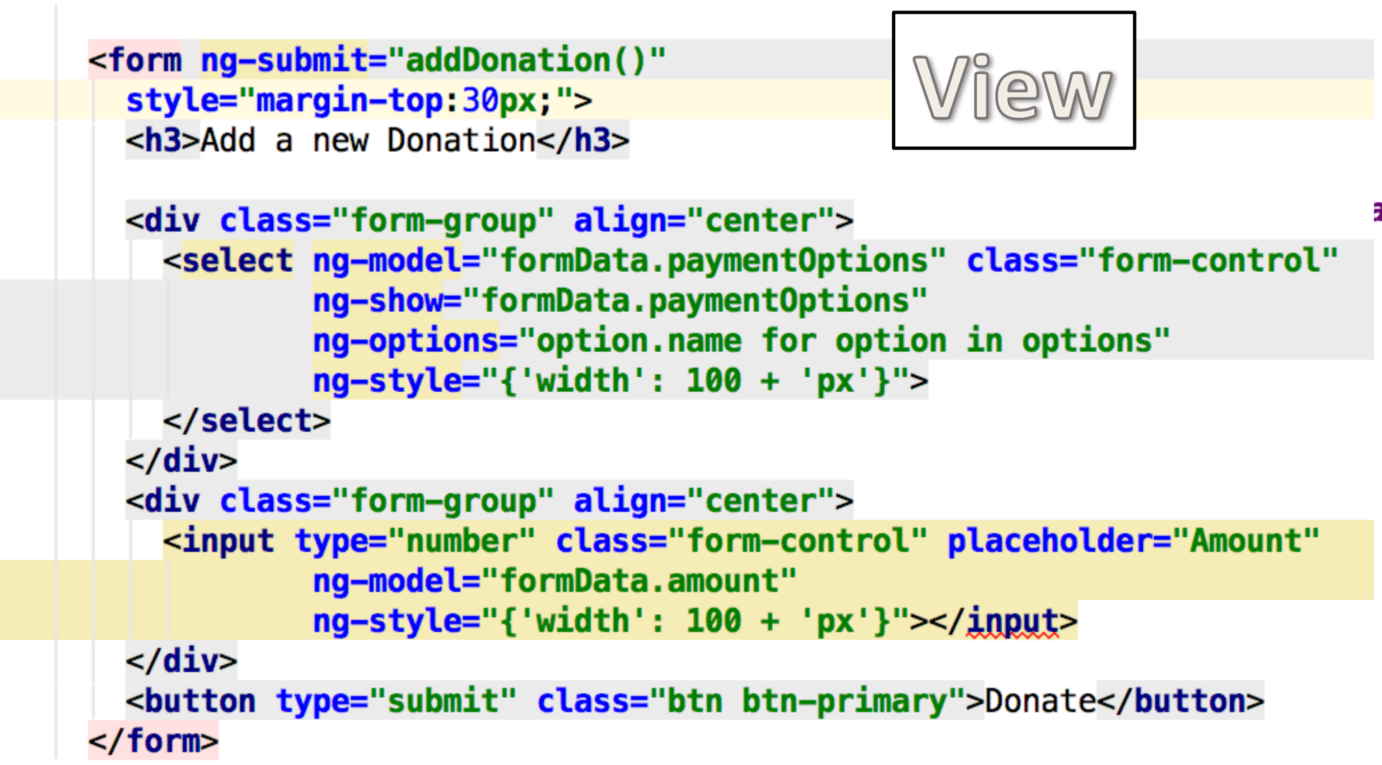
View



```
<form ng-submit="addDonation()"
  style="margin-top:30px;">
  <h3>Add a new Donation</h3>

  <div class="form-group" align="center">
    <select ng-model="formData.paymentOptions" class="form-control"
            ng-show="formData.paymentOptions"
            ng-options="option.name for option in options"
            ng-style="{'width': 100 + 'px'}">
    </select>
  </div>
  <div class="form-group" align="center">
    <input type="number" class="form-control" placeholder="Amount"
            ng-model="formData.amount"
            ng-style="{'width': 100 + 'px'}"></input>
  </div>
  <button type="submit" class="btn btn-primary">Donate</button>
</form>
```

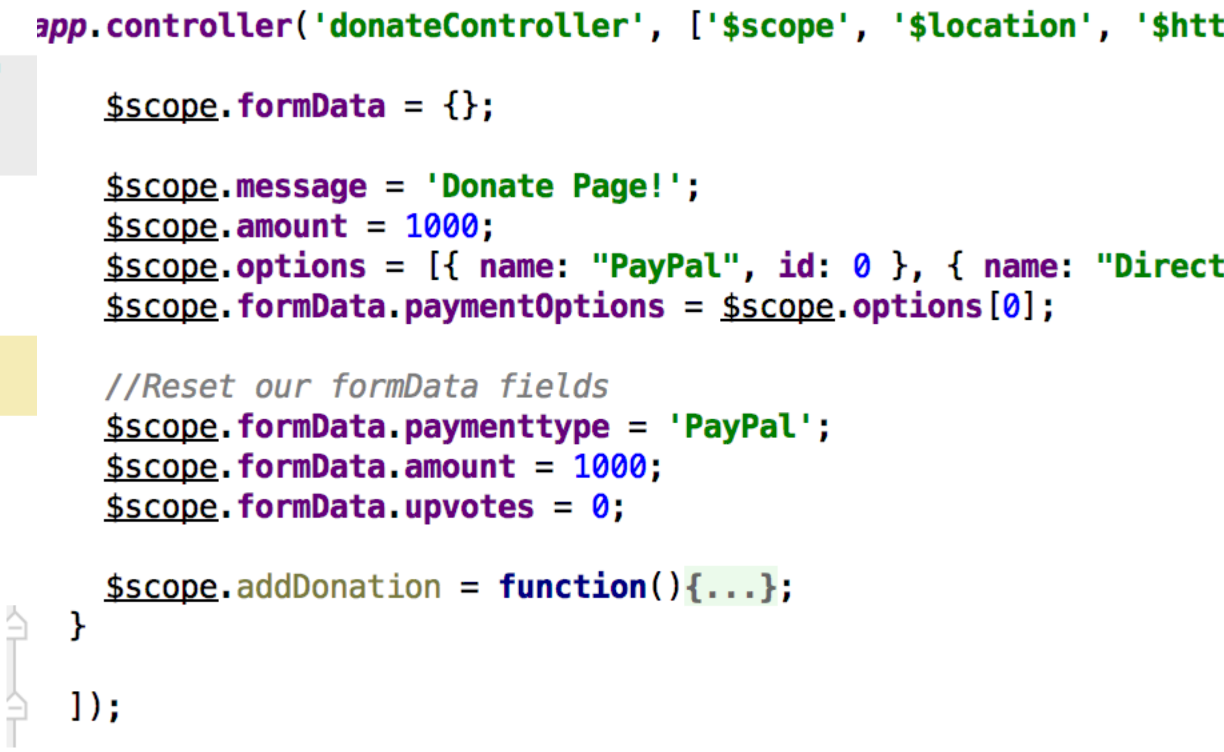
Two Way Data Binding Example

View



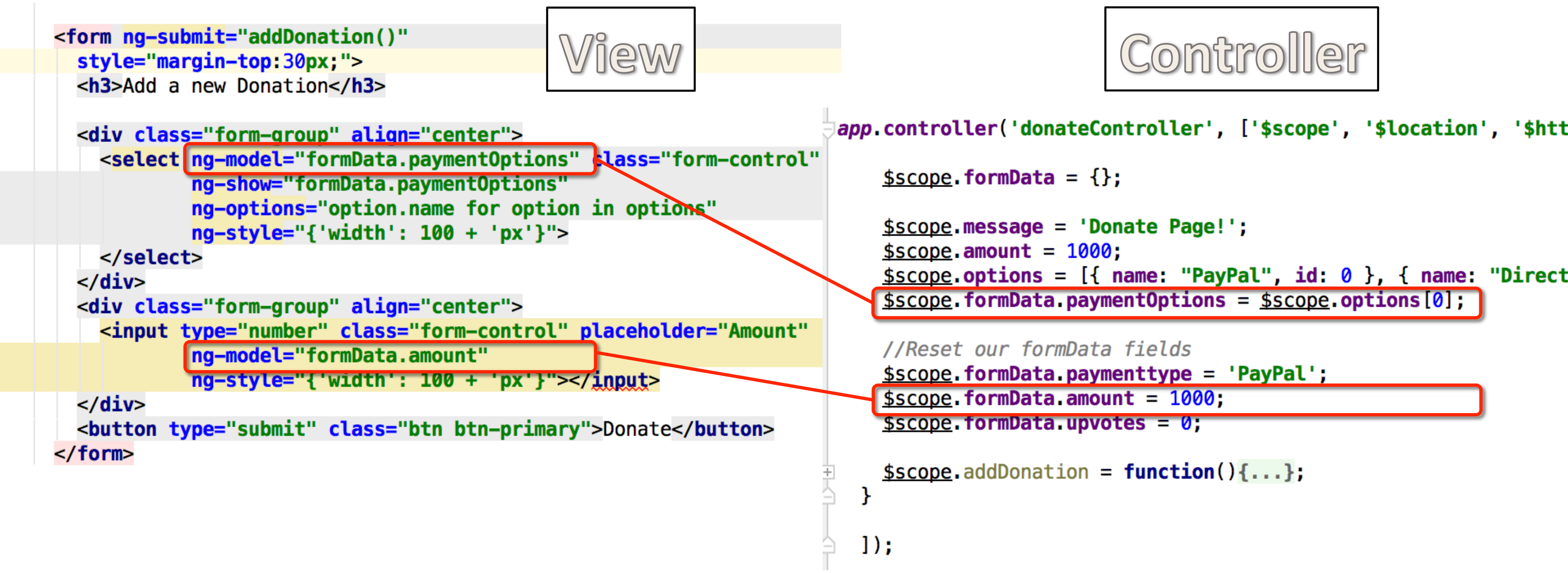
```
<form ng-submit="addDonation()"
  style="margin-top:30px;">
  <h3>Add a new Donation</h3>

  <div class="form-group" align="center">
    <select ng-model="formData.paymentOptions" class="form-control"
            ng-show="formData.paymentOptions"
            ng-options="option.name for option in options"
            ng-style="{'width': 100 + 'px'}">
    </select>
  </div>
  <div class="form-group" align="center">
    <input type="number" class="form-control" placeholder="Amount"
            ng-model="formData.amount"
            ng-style="{'width': 100 + 'px'}"></input>
  </div>
  <button type="submit" class="btn btn-primary">Donate</button>
</form>
```

Controller



```
app.controller('donateController', ['$scope', '$location', '$htt

    $scope.formData = {};

    $scope.message = 'Donate Page!';
    $scope.amount = 1000;
    $scope.options = [{ name: "PayPal", id: 0 }, { name: "Direct
    $scope.formData.paymentOptions = $scope.options[0];

    //Reset our formData fields
    $scope.formData.paymenttype = 'PayPal';
    $scope.formData.amount = 1000;
    $scope.formData.upvotes = 0;

    $scope.addDonation = function(){...};
  }

  ]);
```

Two Way Data Binding Example

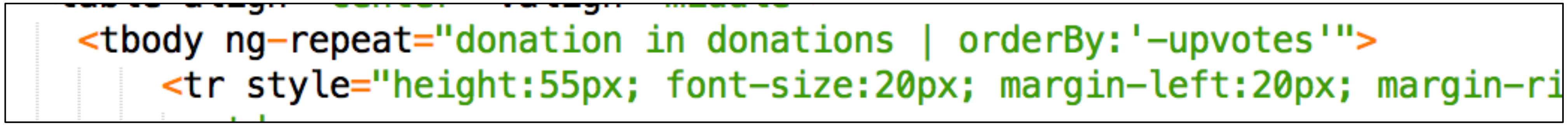


```
<form ng-submit="addDonation()"
  style="margin-top:30px;">
  <h3>Add a new Donation</h3>

  <div class="form-group" align="center">
    <select ng-model="formData.paymentOptions" class="form-control"
            ng-show="formData.paymentOptions"
            ng-options="option.name for option in options"
            ng-style="{'width': 100 + 'px'}">
    </select>
  </div>
  <div class="form-group" align="center">
    <input type="number" class="form-control" placeholder="Amount"
            ng-model="formData.amount"
            ng-style="{'width': 100 + 'px'}"></input>
  </div>
  <button type="submit" class="btn btn-primary">Donate</button>
</form>
```

```
app.controller('donateController', ['$scope', '$location', '$htt

    $scope.formData = {};

    $scope.message = 'Donate Page!';
    $scope.amount = 1000;
    $scope.options = [{ name: "PayPal", id: 0 }, { name: "Direct
    $scope.formData.paymentOptions = $scope.options[0];

    //Reset our formData fields
    $scope.formData.paymenttype = 'PayPal';
    $scope.formData.amount = 1000;
    $scope.formData.upvotes = 0;

    $scope.addDonation = function(){...};
  }

]);
```

Basic Building Blocks – Views *

- More ng directives
  - ✓ ng-if="<model expression>"
    - Inserts HTML element if expression is true
    - Does not insert element in the DOM if it is false
  - ✓ ng-repeat="<variable> in <array>"
    - Repeats the HTML element for each value in the array

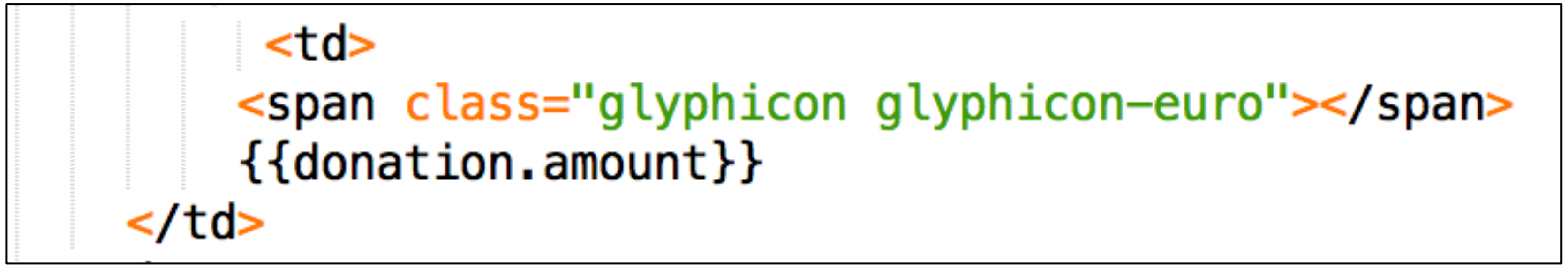


```
<tbody ng-repeat="donation in donations | orderBy:'-upvotes'">
    <tr style="height:55px; font-size:20px; margin-left:20px; margin-ri
```

Basic Building Blocks - **Views**

Angular Expression: {{ }}

- Used to insert model values directly into the view
- (an extract from ***donations.ejs***)



```
        <td>
       <span class="glyphicon glyphicon-euro"></span>
       {{donation.amount}}
   </td>
```

Modules

REUSABLE FUNCTIONALITY

Basic Building Blocks – Modules

Modules are a way of organizing your code in which you split up the work between different sections of your code rather than writing a single huge application.

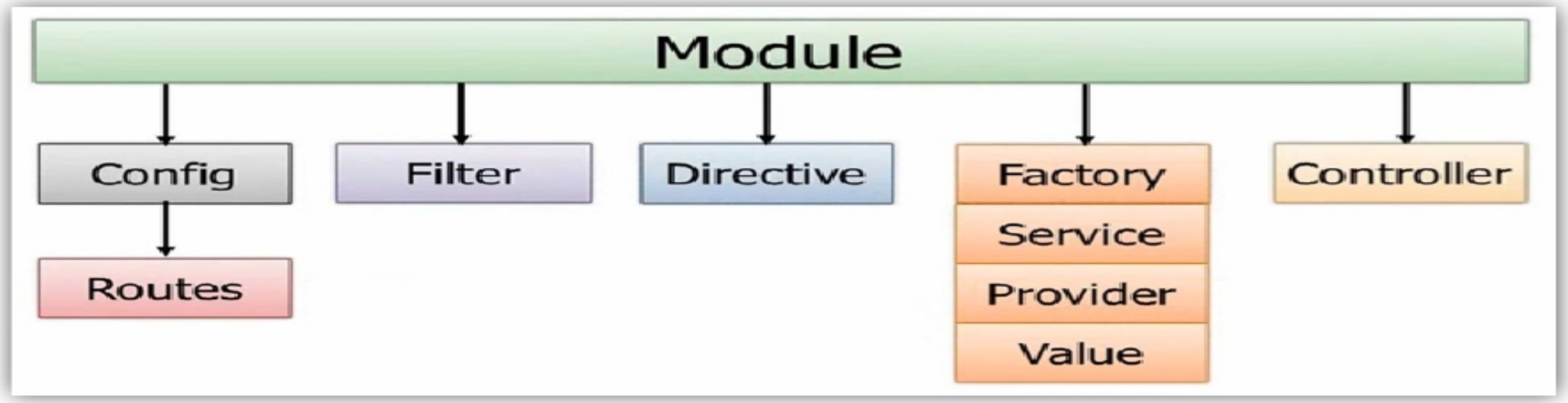
An application module can include the other modules (sections) by listing them as dependencies.

```
angular.module(<name>, [<dependencies>]);
```

Basic Building Blocks – Modules

Modules are a way of organizing your code in which you split up the work between different sections of your code rather than writing a single huge application.

An application module can include the other modules (sections) by listing them as dependencies.



Module Definition *

To define an AngularJS app, we first need to define an ***angular.module***. An Angular module is simply a collection of functions that are run when the application is “booted.” All apps have ***at least*** one module.

Define a module:

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

Define a module with dependencies on other modules:

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

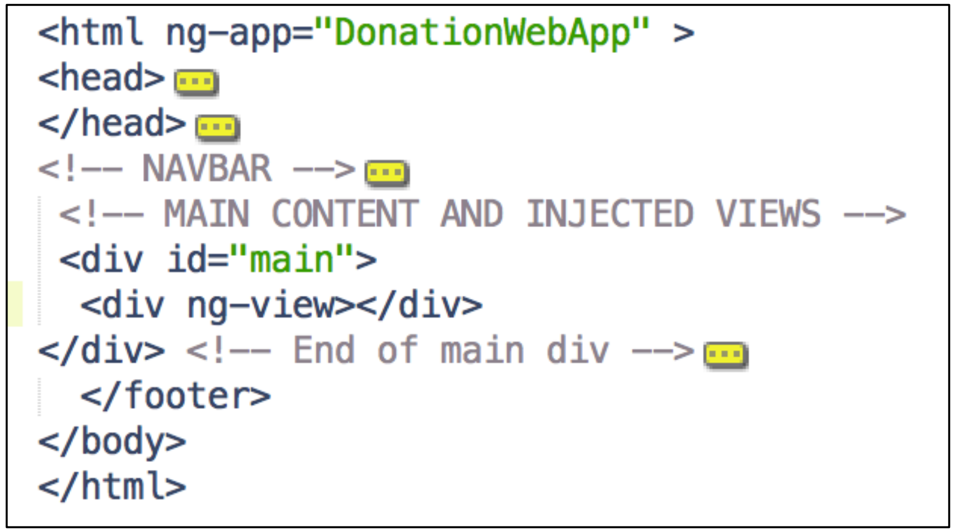
Get an existing module:

```
var app = angular.module('DonationWebApp');
```

The Application Module

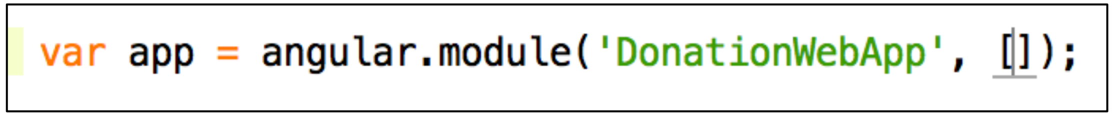
AngularJS provides a way for you to bind your main module to the HTML document using the ***ng-app*** directive.

HTML FRAGMENT



```
<html ng-app="DonationWebApp" >
<head>
</head>
<!-- NAVBAR -->
 <!-- MAIN CONTENT AND INJECTED VIEWS -->
 <div id="main">
  <div ng-view></div>
</div> <!-- End of main div -->
  </footer>
</body>
</html>
```

JAVASCRIPT FRAGMENT

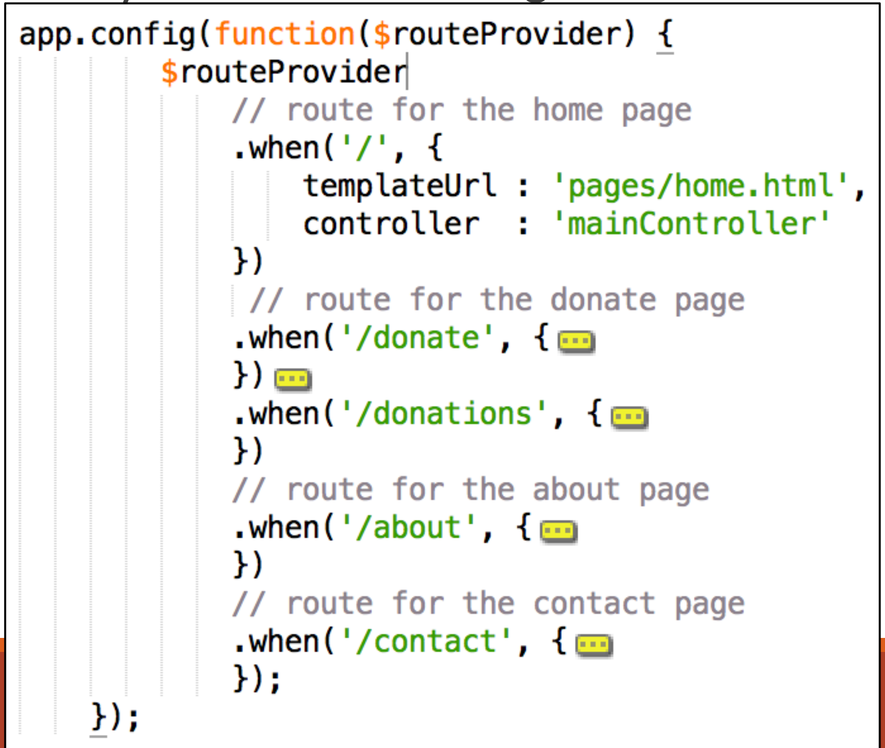


```
var app = angular.module('DonationWebApp', []);
```

Module Phases

CONFIG

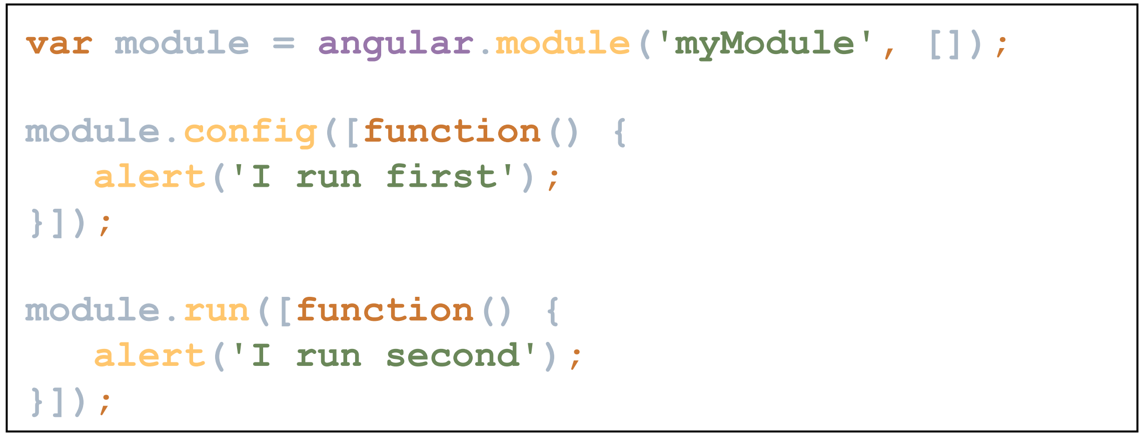
The config phase happens early while the application is still being built. Only the provider services and constant services are ready for D.I. at this stage.



```
app.config(function($routeProvider) {
        $routeProvider
            // route for the home page
            .when('/', {
                templateUrl : 'pages/home.html',
                controller  : 'mainController'
            })
            // route for the donate page
            .when('/donate', {
            })
            .when('/donations', {
            })
            // route for the about page
            .when('/about', {
            })
            // route for the contact page
            .when('/contact', {
            });
    });
```

RUN

The run phase happens once the module has loaded all of its services and dependencies.



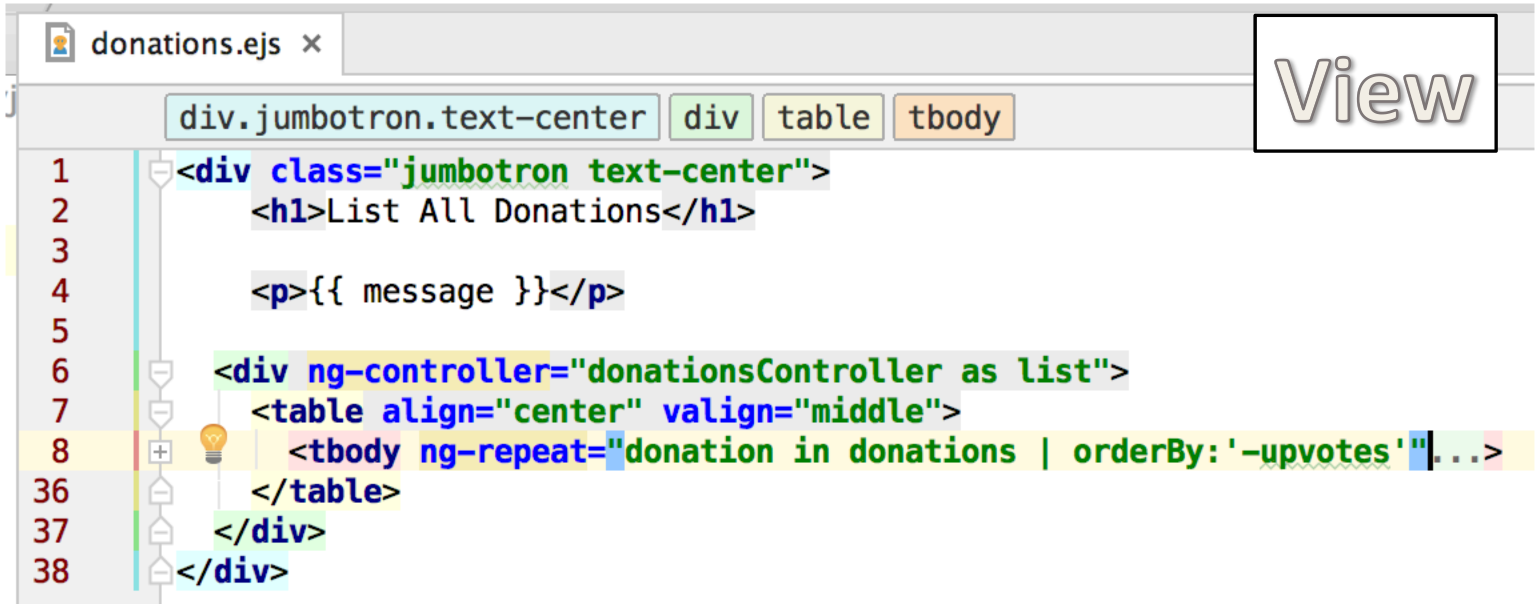
```
var module = angular.module('myModule', []);

module.config([function() {
    alert('I run first');
}]);

module.run([function() {
    alert('I run second');
}]);
```

Module Components & D.I.

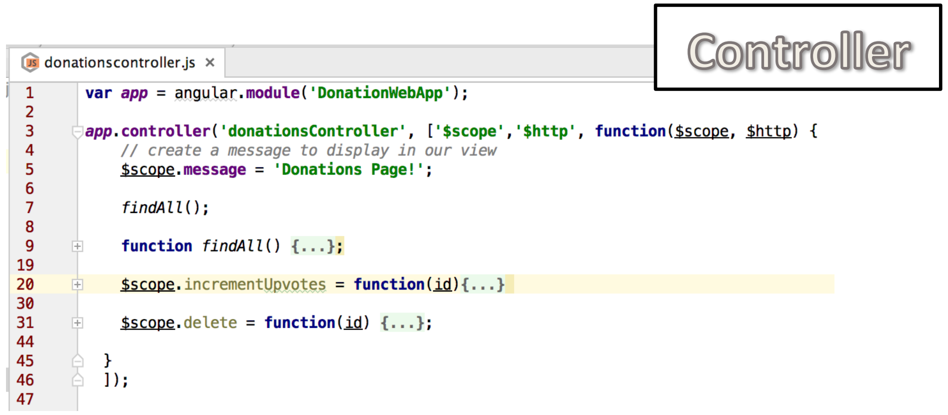
AngularJS lets you inject **services** (either from its own module or from other modules) with the following pattern:

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

module.service('serviceA', function() { ... });

module.service('serviceB', function(serviceA) { ... });
```

Donation MVC App using Modules *



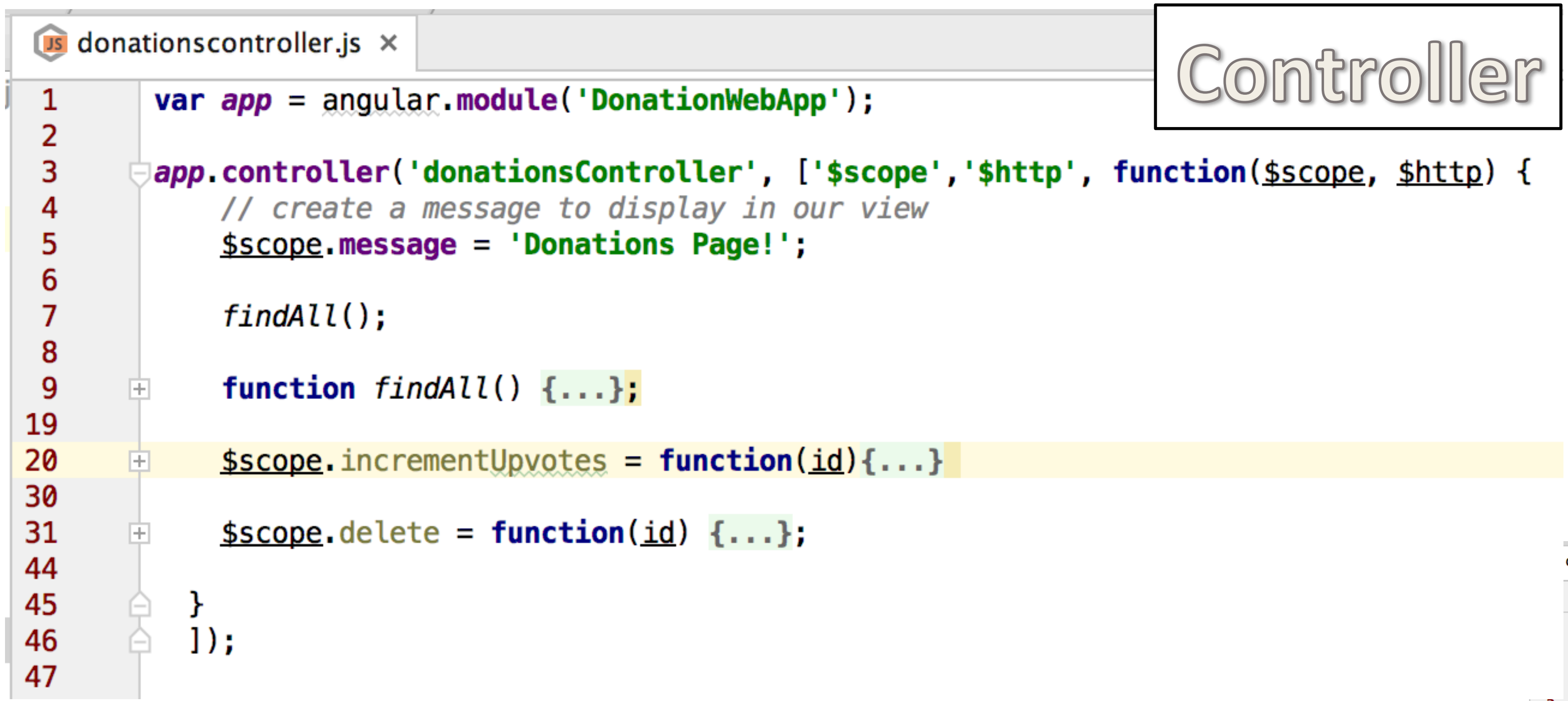
```
<div class="jumbotron text-center">
    <h1>List All Donations</h1>

    <p>{{ message }}</p>

  <div ng-controller="donationsController as list">
    <table align="center" valign="middle">
      <tbody ng-repeat="donation in donations | orderBy:'-upvotes'"...>
    </table>
  </div>
</div>
```



```
var app = angular.module('DonationWebApp');

app.controller('donationsController', ['$scope','$http', function($scope, $http) {
    // create a message to display in our view
    $scope.message = 'Donations Page!';

    findAll();

    function findAll() {...};

    $scope.incrementUpvotes = function(id){...}

    $scope.delete = function(id) {...};

  }
  ]);
```

Donation MVC App using Modules

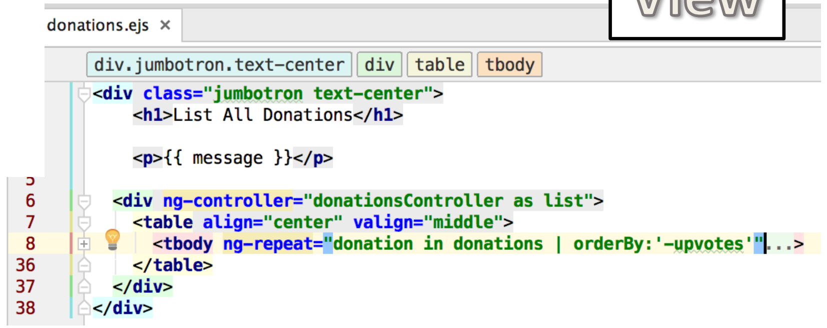
Controller



donationscontroller.js

```
var app = angular.module('DonationWebApp');

app.controller('donationsController', ['$scope','$http', function($scope, $http) {
    // create a message to display in our view
    $scope.message = 'Donations Page!';

    findAll();

    function findAll() {...};

    $scope.incrementUpvotes = function(id){...}

    $scope.delete = function(id) {...};

  }
  ]);
```

View



donations.ejs

div.jumbotron.text-center | div | table | tbody

```
<div class="jumbotron text-center">
    <h1>List All Donations</h1>

    <p>{{ message }}</p>

  <div ng-controller="donationsController as list">
    <table align="center" valign="middle">
      <tbody ng-repeat="donation in donations | orderBy:'-upvotes'"...>
    </table>
  </div>
</div>
```

Modules – App Design

Recommendations:

- A module for each feature.
- A module for each reusable component (especially custom directives and filters)
- And an application level module which depends on the above modules and contains any initialization code.

Controllers

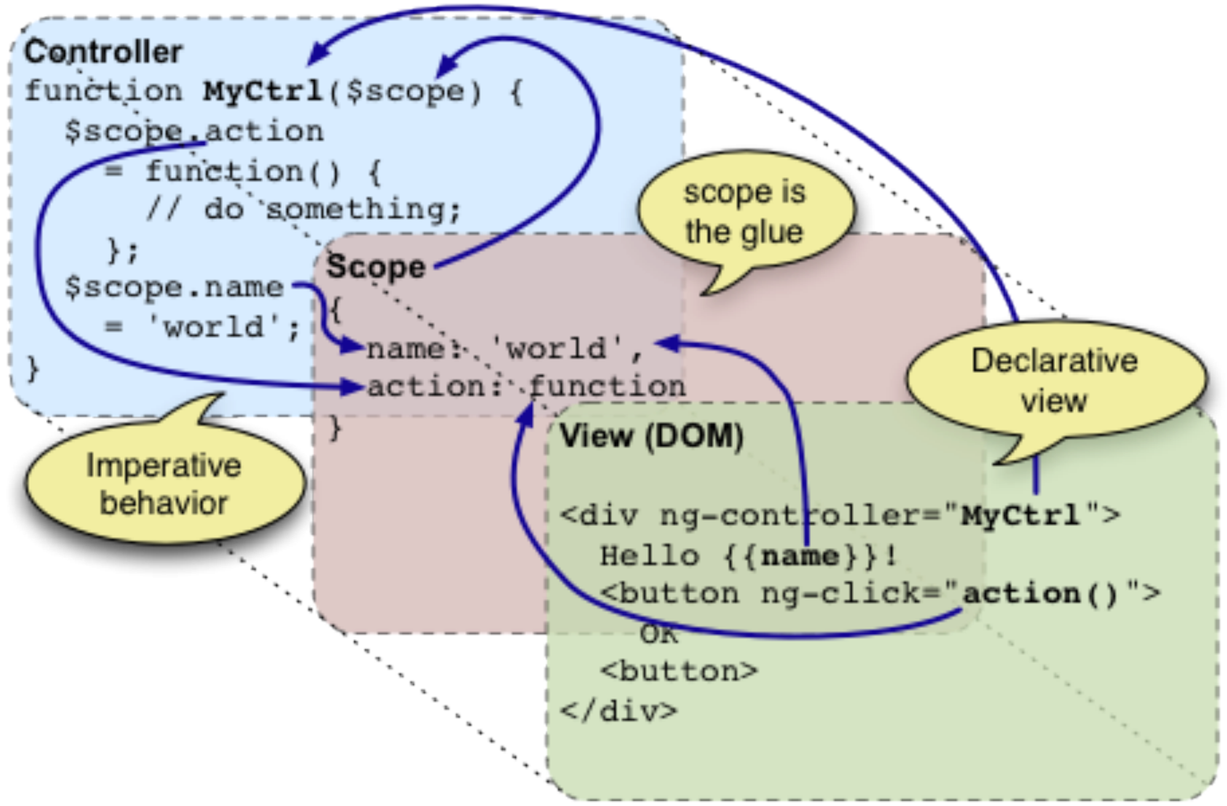
FACILITATING COMMUNICATION BETWEEN THE MODEL AND THE VIEW

Basic Building Blocks - Controller

The interface between the model and the view

Contains the code behind the view

Try to keep lightweight



```
function MyCtrl($scope) {
  $scope.action
    = function() {
      // do something;
    };
  $scope.name
    = 'world';
}
```

```
{
name: 'world',
action: function
}
```

```
<div ng-controller="MyCtrl">
  Hello {{name}}!
  <button ng-click="action()">
    OK
  <button>
</div>
```

ANGULARJS

Basic Building Blocks - Scope

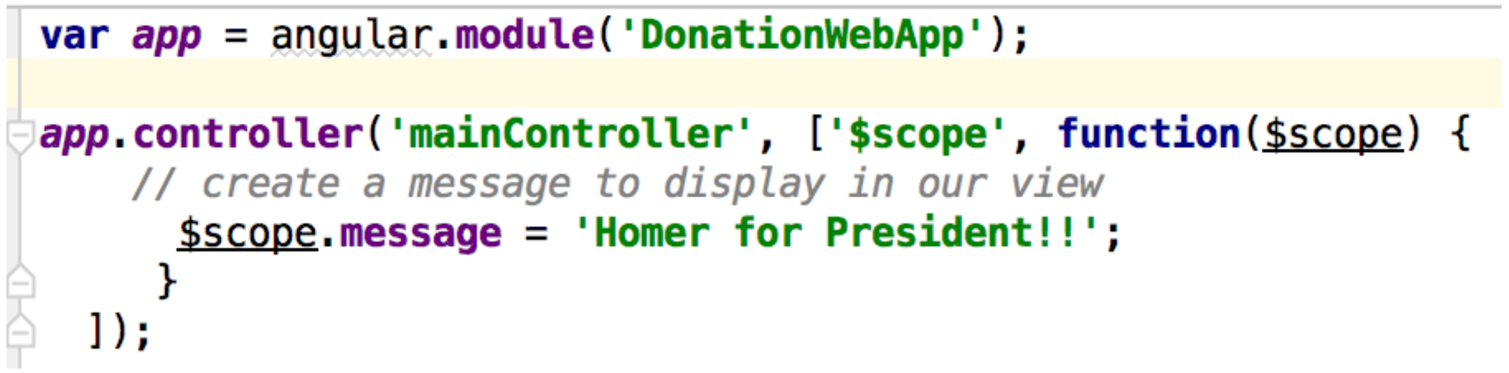
- A **$scope** is an object that ties a view (a DOM element) to the **controller**
  - In the Model-View-Controller structure, this **$scope** object becomes the model.
  - It provides an *execution context* that is *bound* to the DOM element (and its children).
- Although it sounds complex, the **$scope** is *just a JavaScript object*.
  - Both the **controller** and the **view** have access to the **$scope** so it can be used for communication between the two.
  - This **$scope** object will house both the data and the functions that we'll want to run in the view, as we'll see.

Basic Building Blocks - Scope

$scope

- Contains data (i.e. models) and methods (i.e. functions)
- Is the engine for 2-way data binding
- Can add your own properties
  - $scope.<my new property> = <value>;

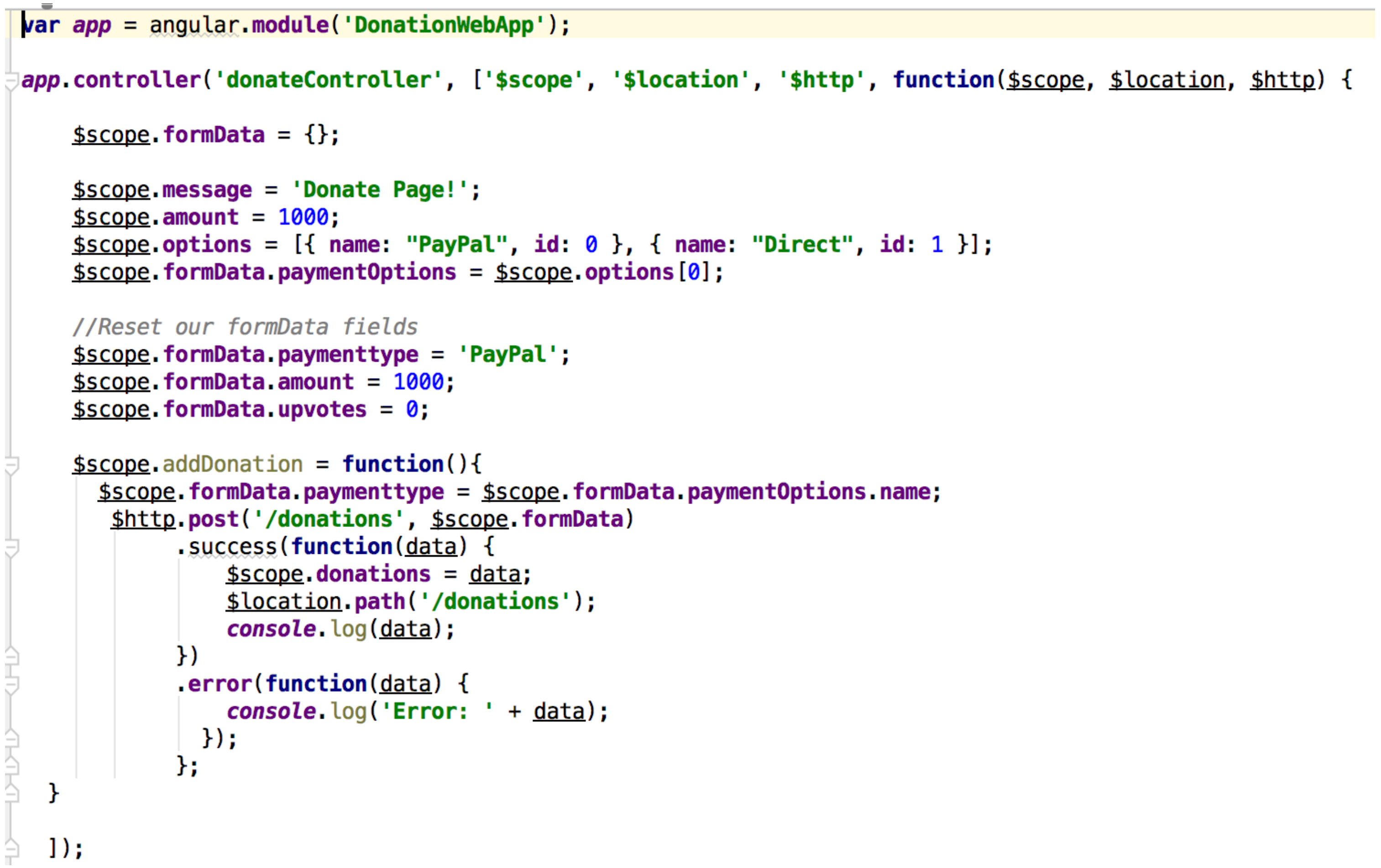
Controller function takes at least one parameter: **$scope**



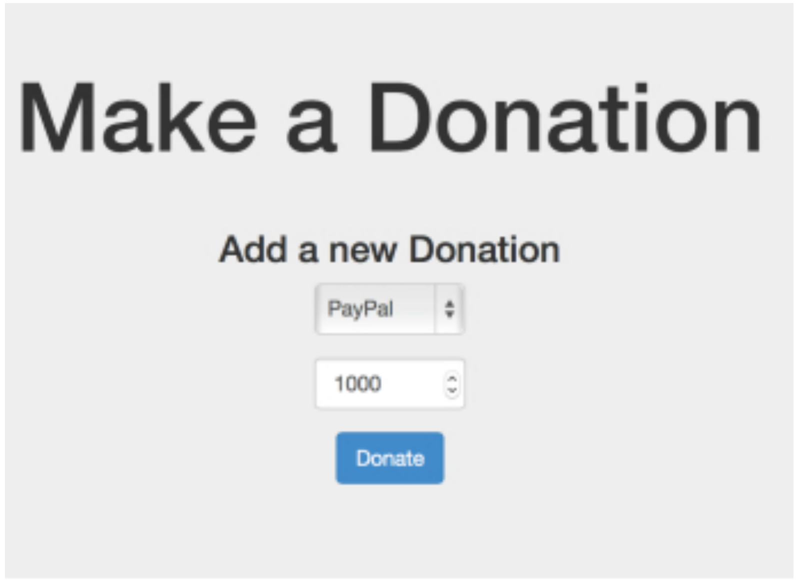
```
var app = angular.module('DonationWebApp');

app.controller('mainController', ['$scope', function($scope) {
    // create a message to display in our view
      $scope.message = 'Homer for President!!';
     }
  ]);
```

‘Donate’ Example

Controllers and Scope *

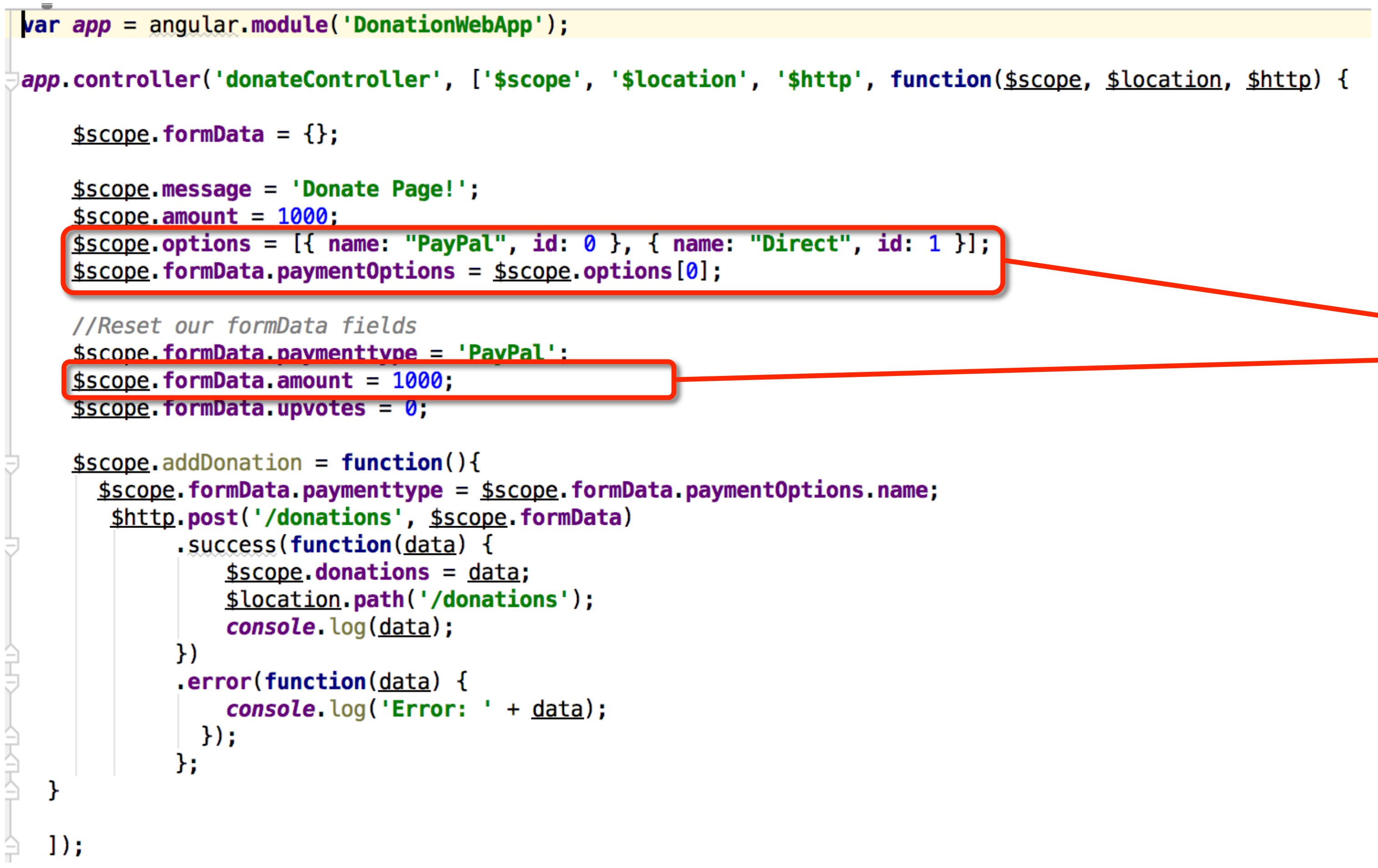


```
var app = angular.module('DonationWebApp');

app.controller('donateController', ['$scope', '$location', '$http', function($scope, $location, $http) {

    $scope.formData = {};

    $scope.message = 'Donate Page!';
    $scope.amount = 1000;
    $scope.options = [{ name: "PayPal", id: 0 }, { name: "Direct", id: 1 }];
    $scope.formData.paymentOptions = $scope.options[0];

    //Reset our formData fields
    $scope.formData.paymenttype = 'PayPal';
    $scope.formData.amount = 1000;
    $scope.formData.upvotes = 0;

    $scope.addDonation = function(){
      $scope.formData.paymenttype = $scope.formData.paymentOptions.name;
       $http.post('/donations', $scope.formData)
            .success(function(data) {
                $scope.donations = data;
                $location.path('/donations');
                console.log(data);
            })
            .error(function(data) {
                console.log('Error: ' + data);
              });
            };
  }

  ]);
```

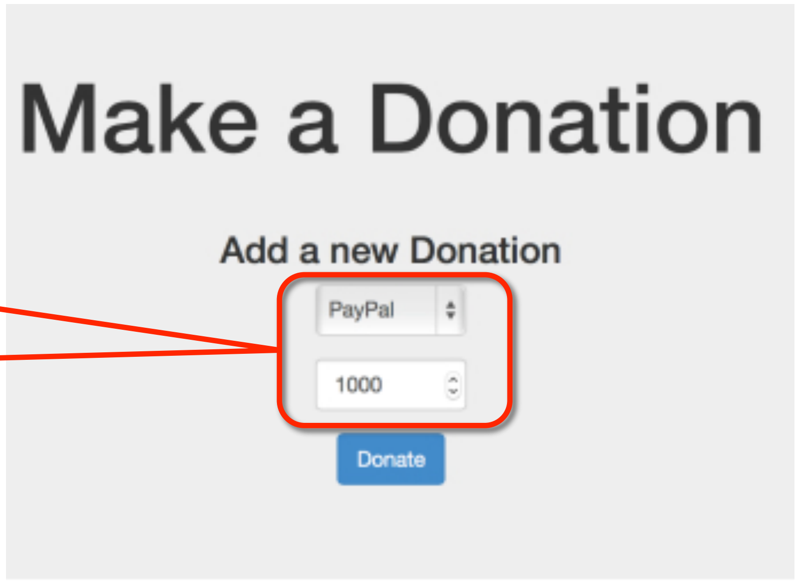


Controllers and Scope *

‘Donate’ Example

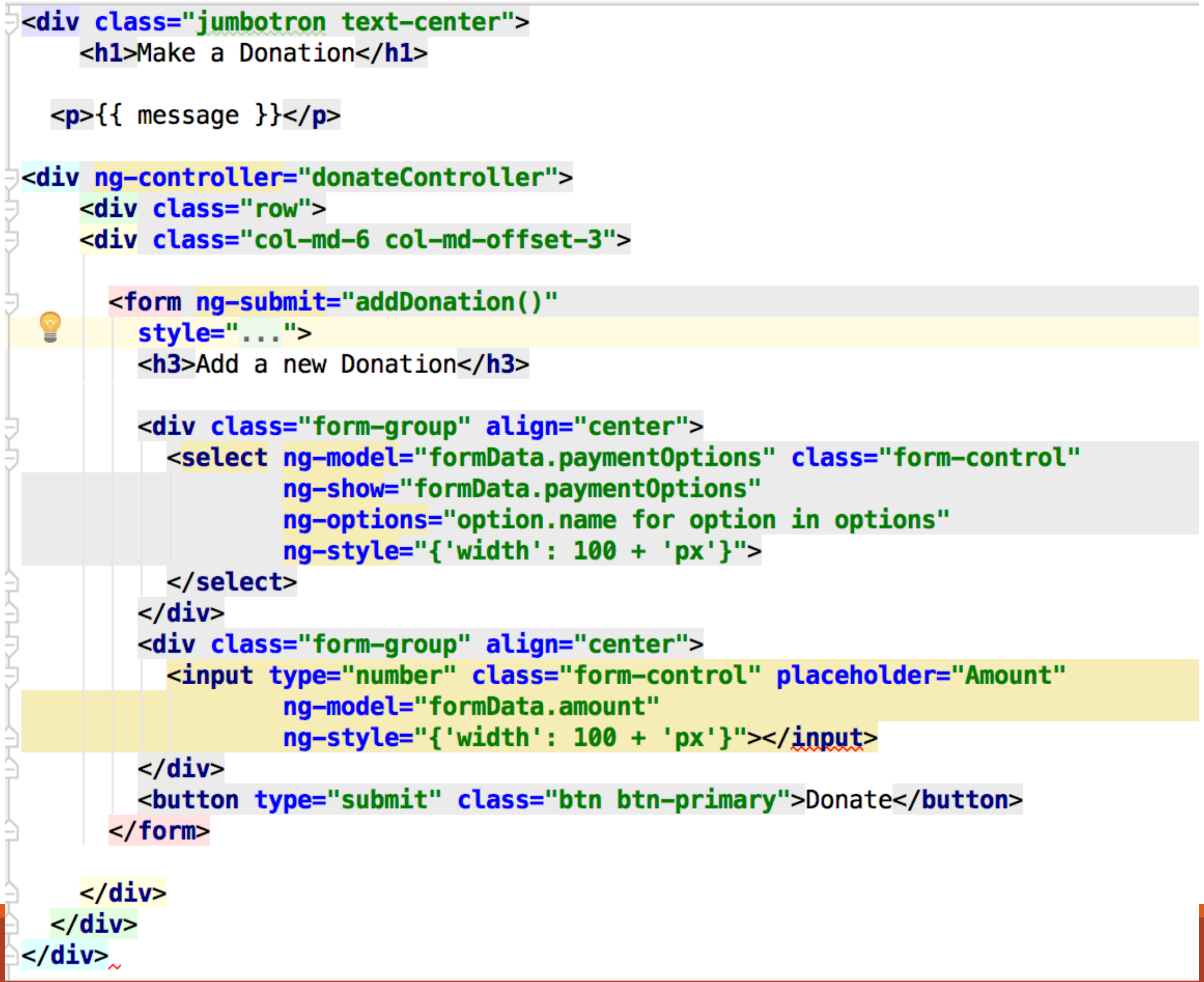


```
var app = angular.module('DonationWebApp');

app.controller('donateController', ['$scope', '$location', '$http', function($scope, $location, $http) {

    $scope.formData = {};

    $scope.message = 'Donate Page!';
    $scope.amount = 1000;
    $scope.options = [{ name: "PayPal", id: 0 }, { name: "Direct", id: 1 }];
    $scope.formData.paymentOptions = $scope.options[0];

    //Reset our formData fields
    $scope.formData.paymenttype = 'PayPal';
    $scope.formData.amount = 1000;
    $scope.formData.upvotes = 0;

    $scope.addDonation = function(){
      $scope.formData.paymenttype = $scope.formData.paymentOptions.name;
       $http.post('/donations', $scope.formData)
            .success(function(data) {
                $scope.donations = data;
                $location.path('/donations');
                console.log(data);
            })
            .error(function(data) {
                console.log('Error: ' + data);
              });
            };
  }

  ]);
```

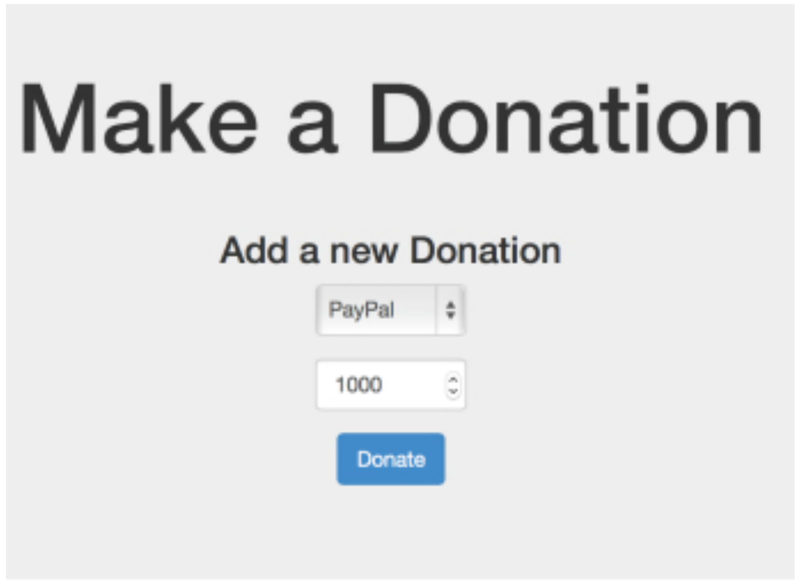


'Donate' Example

Controllers and Scope *

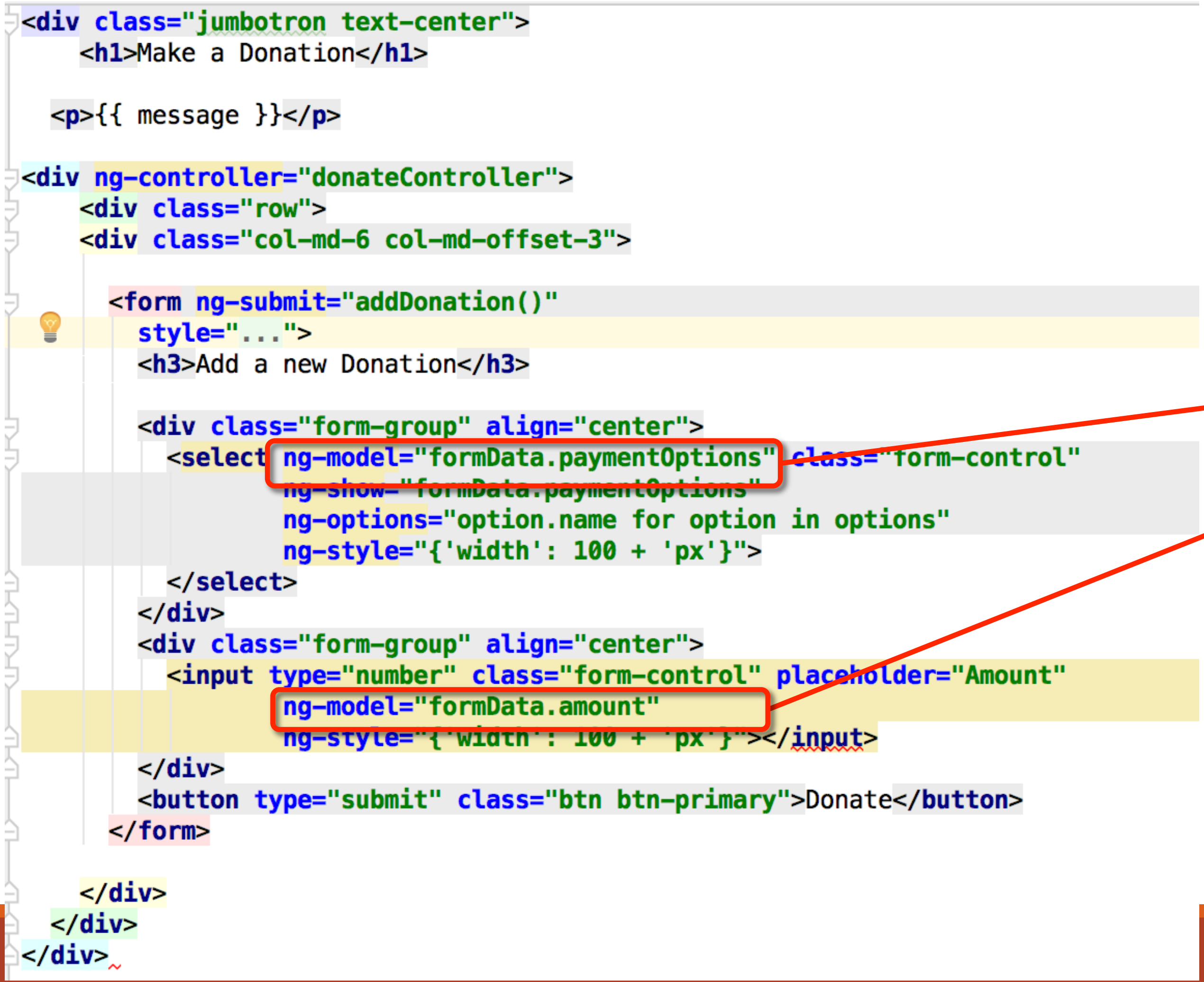


```
<div class="jumbotron text-center">
    <h1>Make a Donation</h1>

  <p>{{ message }}</p>

<div ng-controller="donateController">
    <div class="row">
    <div class="col-md-6 col-md-offset-3">

      <form ng-submit="addDonation()"
        style="...">
        <h3>Add a new Donation</h3>

        <div class="form-group" align="center">
          <select ng-model="formData.paymentOptions" class="form-control"
                  ng-show="formData.paymentOptions"
                  ng-options="option.name for option in options"
                  ng-style="{'width': 100 + 'px'}">
          </select>
        </div>
        <div class="form-group" align="center">
          <input type="number" class="form-control" placeholder="Amount"
                  ng-model="formData.amount"
                  ng-style="{'width': 100 + 'px'}"></input>
        </div>
        <button type="submit" class="btn btn-primary">Donate</button>
      </form>

    </div>
  </div>
</div>
```

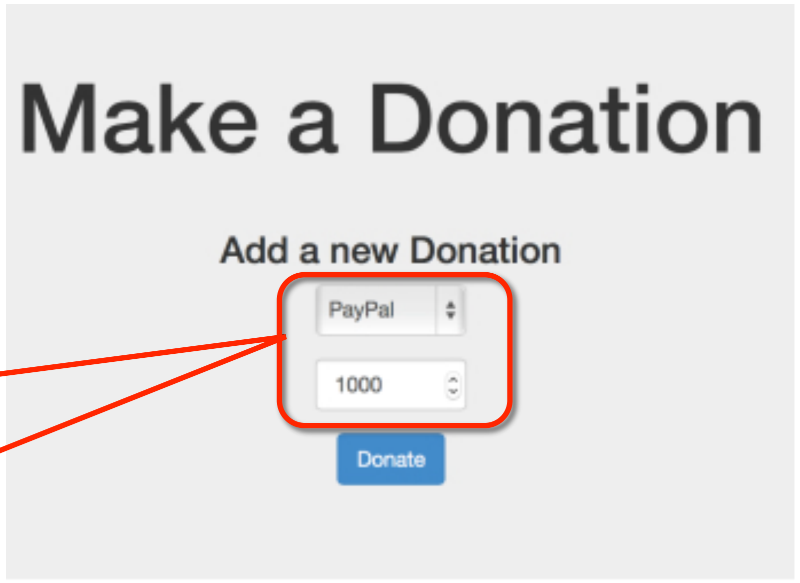


'Donate' Example

Controllers and Scope



```
<div class="jumbotron text-center">
  <h1>Make a Donation</h1>

 <p>{{ message }}</p>

<div ng-controller="donateController">
  <div class="row">
  <div class="col-md-6 col-md-offset-3">

   <form ng-submit="addDonation()"
    style="...">
    <h3>Add a new Donation</h3>

    <div class="form-group" align="center">
     <select ng-model="formData.paymentOptions" class="form-control"
         ng-show="formData.paymentOptions"
         ng-options="option.name for option in options"
         ng-style="{'width': 100 + 'px'}">
     </select>
    </div>
    <div class="form-group" align="center">
     <input type="number" class="form-control" placeholder="Amount"
         ng-model="formData.amount"
         ng-style="{'width': 100 + 'px'}"></input>
    </div>
    <button type="submit" class="btn btn-primary">Donate</button>
   </form>

  </div>
 </div>
</div>
```



Routes

NAVIGATING THE VIEW

Basic Building Blocks - Routing

Allows SPAs behave like traditional Web Apps/sites

- forward/back button support
- deep-linking to specific content
- Old-style AJAX Web Apps didn't support routing (addressability problem)

- Advantages
  - Bookmarking, link sharing, direct navigation
- Two Solution Approaches:
  - Hash-based,   e.g.   http://domain_name/#/some_app_url
  - PushState,   e.g.   http://domain_name/some_app_url

Angular supports both (next slide)

Basic Building Blocks - Routing

Paths default to Hash-based mode

- Example URL.
  - http://www.mysite.com/#/users

Can use HTML 5 mode by configuring the **$locationProvider** (next few slides)

- Ex.
  - // Inject $locationProvider into the module using config

    $locationProvider.html5Mode(true);
- Example URL:
  - http://www.mysite.com/users

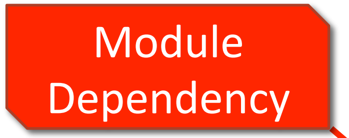
Basic Building Blocks - Routing

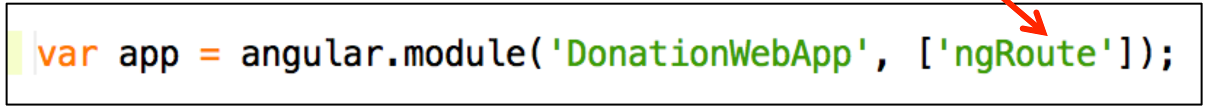
Use different views for different URL fragments

Makes use of template partials

- Templates that are not a whole web page (i.e. part of a page)
- Used in conjunction with the **ng-view** directive
  - ng-view determines where the partial will be placed
  - Can only have one ng-view per page

Basic Building Blocks - Routing





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

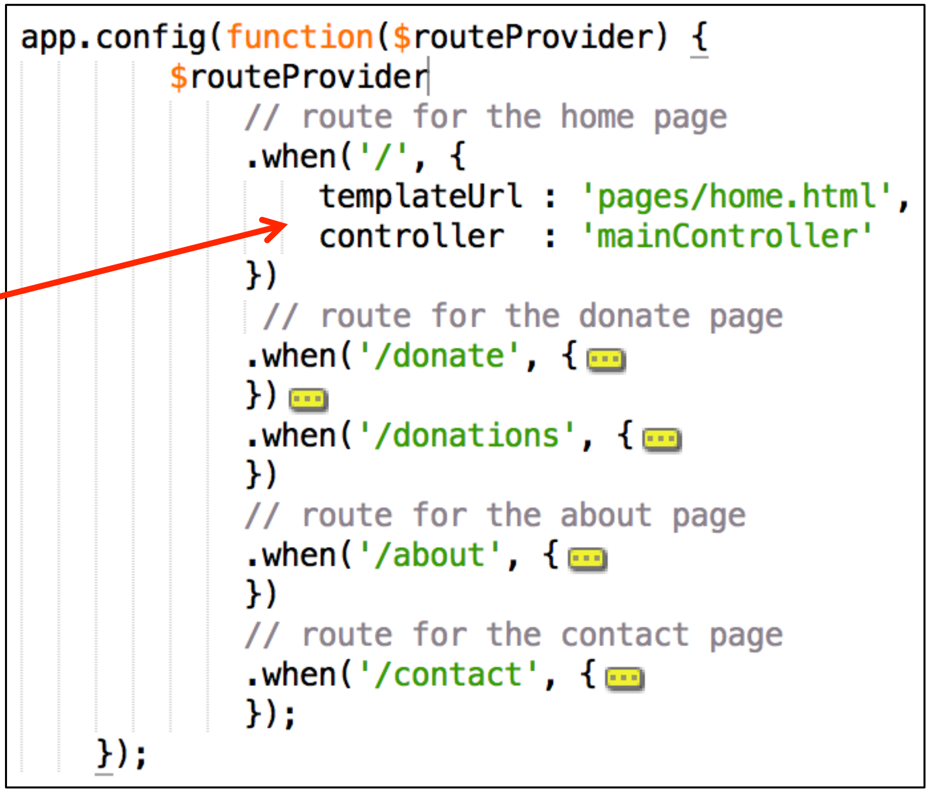
Enable by injecting the $routeProvider

- myApp = angular.module('myApp', ['ngRoute']);

  myApp.config(['$routeProvider', function($routeProvider) { … }]);

$routeProvider.when(<path>, {<route>});

- Defines a new route that uses the given path
- The path may have parameters
  - Parameters start with a colon (':')
  - Ex - '/user/:userId'   ($routeParams.userid)
- Typical route fields:
  - controller = The name of the controller that should be used
  - templateUrl = A path to the template partial that should be used



```
app.config(function($routeProvider) {
        $routeProvider
            // route for the home page
            .when('/', {
                templateUrl : 'pages/home.html',
                controller  : 'mainController'
            })
             // route for the donate page
            .when('/donate', {
            })
            .when('/donations', {
            })
            // route for the about page
            .when('/about', {
            })
            // route for the contact page
            .when('/contact', {
            });
    });
```

$routeProvider.otherwise({<route>});

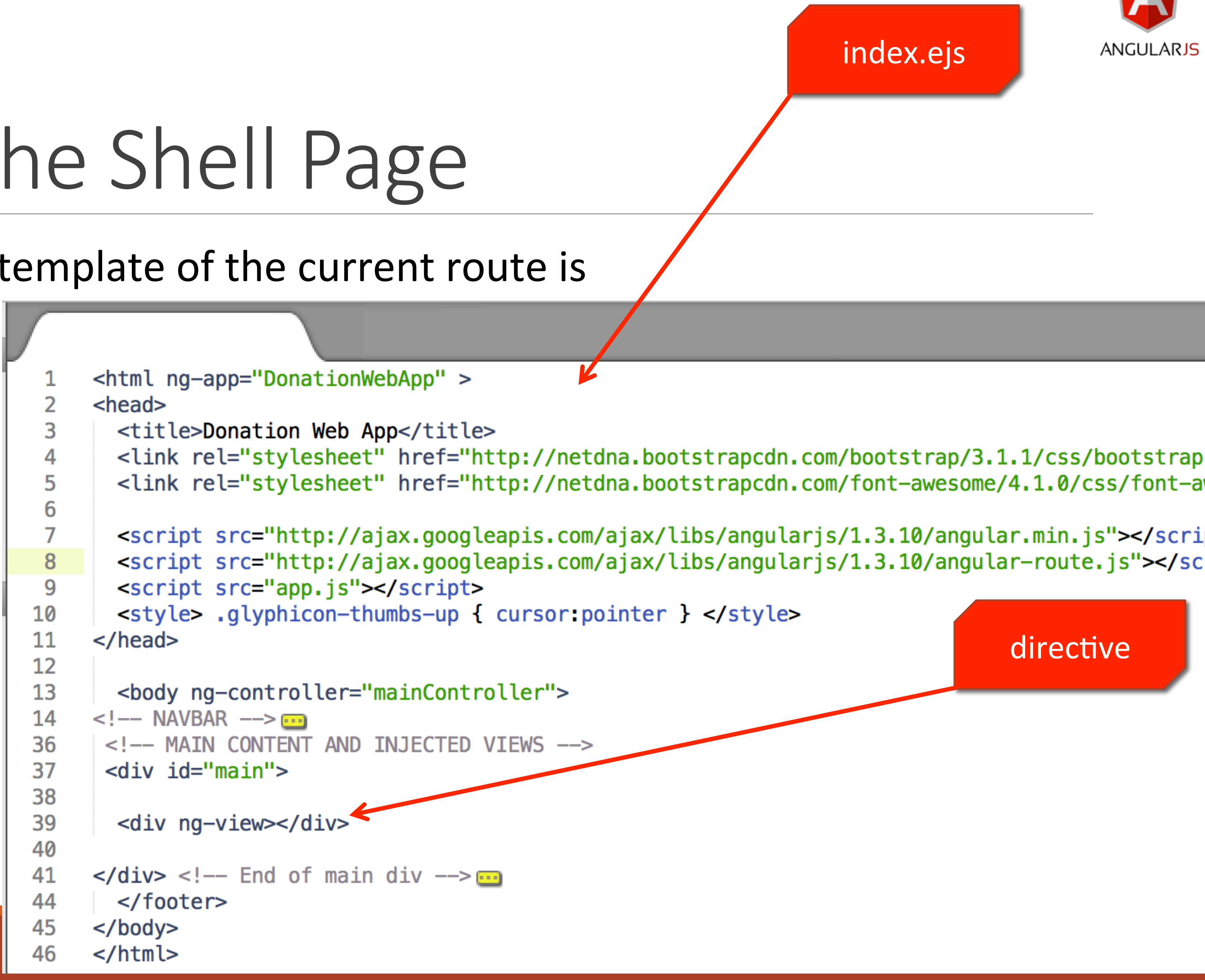
- Typical route fields:
  - redirectTo: '<path>'

API: http://docs.angularjs.org/api/ngRoute.$routeProvider

Routing – The Shell Page

ng-view directive – rendered template of the current route is

- dynamically inserted into the shell page (index.ejs)
- URL change → Template change + Controller instantiation
- Shell page = Layout page.

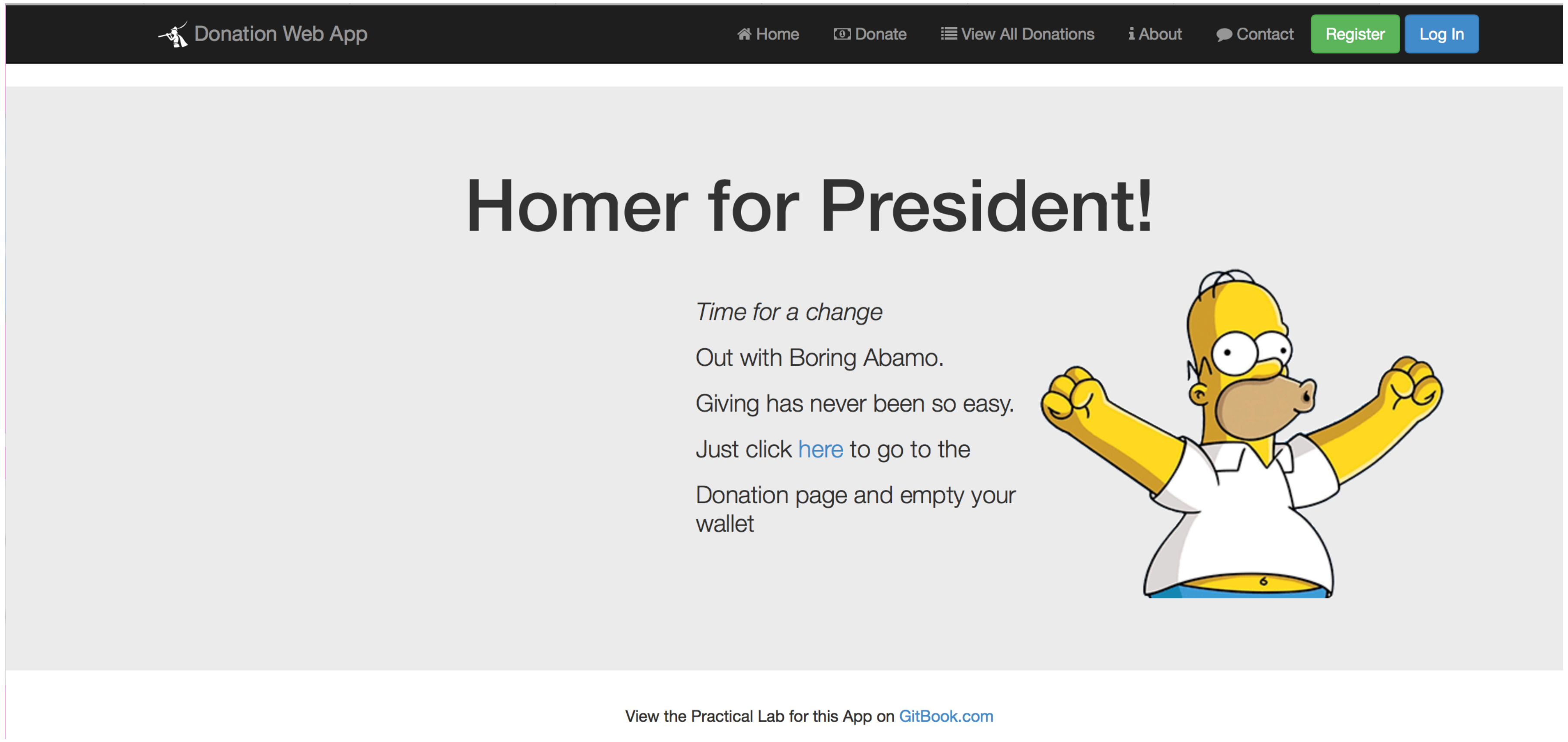


```
<html ng-app="DonationWebApp" >
<head>
  <title>Donation Web App</title>
  <link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.mir
  <link rel="stylesheet" href="http://netdna.bootstrapcdn.com/font-awesome/4.1.0/css/font-aweso

  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.10/angular.min.js"></script>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.10/angular-route.js"></script
  <script src="app.js"></script>
  <style> .glyphicon-thumbs-up { cursor:pointer } </style>
</head>

  <body ng-controller="mainController">
<!-- NAVBAR -->
 <!-- MAIN CONTENT AND INJECTED VIEWS -->
 <div id="main">

  <div ng-view></div>

</div> <!-- End of main div -->
  </footer>
</body>
</html>
```

Case Study

LABS IN ACTION

Demo Application



Homer for President!

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

View the Practical Lab for this App on GitBook.com

About the Original Author

James Speirs

Application Foundations

OIT Core Services

Brigham Young University

Questions?