Building Strongly-Typed AngularJS Apps with ASP.NET MVC 5

Introduction

Matt Honeycutt

@matthoneycutt | http://trycatchfail.com/strongly-typed-angularjs

Wait… What??

Strongly-typed +  = **No way!**

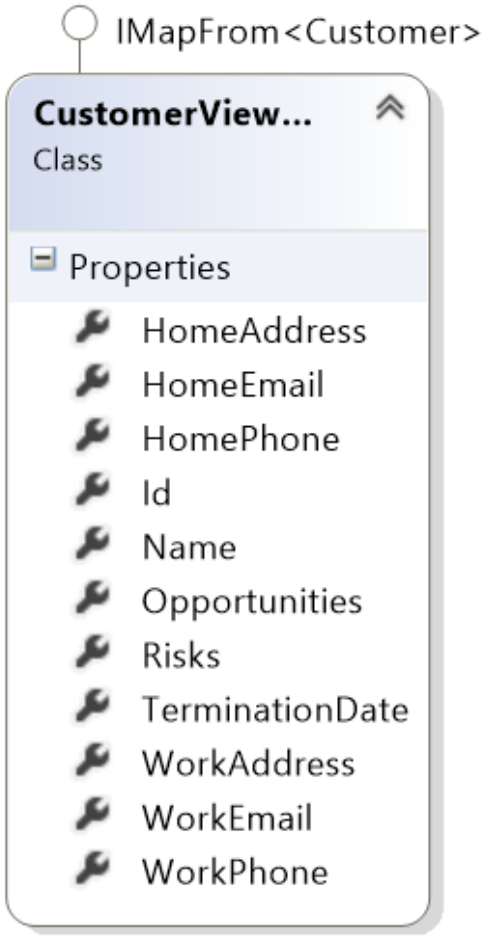
(this is probably you!)

Typical Markup

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.Name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="vm.customer.WorkEmail"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label>
    <input ng-model="vm.customer.HomeEmail"
           class="form-control" name="HomeEmail" type="email" placeholder="user@domain.com...">
</div>

...
```

Typical Markup

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.Name"
           class="form-control" name="Name" type="text" placeholde

</div>

<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label
    <input required ng-model="vm.customer.WorkEmail"
           class="form-control" name="WorkEmail" type="email" plac

</div>

<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label
    <input ng-model="vm.customer.HomeEmail"
           class="form-control" name="HomeEmail" type="email" plac

</div>

...
```



Typical Markup

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="vm.customer.Name"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="vm.customer.WorkEmail"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label>
    <input ng-model="vm.customer.HomeEmail"
           class="form-control" name="HomeEmail" type="email" placeholder="user@domain.com...">
</div>

...
```

Typical Markup
(with just a bit of work)

```
<div class="form-group has-feedback">
    <label class="control-label" for="Name">Name</label>
    <input required ng-model="@customer.ExpressionFor(x => x.Name)"
           class="form-control" name="Name" type="text" placeholder="Full name (ex: John Smith)...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="WorkEmail">Work Email</label>
    <input required ng-model="@customer.ExpressionFor(x => x.WorkEmail)"
           class="form-control" name="WorkEmail" type="email" placeholder="user@domain.com...">
</div>

<div class="form-group has-feedback">
    <label class="control-label" for="HomeEmail">Home Email</label>
    <input ng-model="@customer.ExpressionFor(x => x.HomeEmail)"
           class="form-control" name="HomeEmail" type="email" placeholder="user@domain.com...">
</div>

...
```

Typical Markup (with a bit *more* work)

```
@customer.FormGroupFor(x => x.Name)
@customer.FormGroupFor(x => x.WorkEmail)
@customer.FormGroupFor(x => x.HomeEmail)
@customer.FormGroupFor(x => x.WorkPhone)
@customer.FormGroupFor(x => x.HomePhone)
@customer.FormGroupFor(x => x.WorkAddress)
@customer.FormGroupFor(x => x.HomeAddress)
```

Typical Markup
(with a bit *more* work)

```
@Html.Angular().FormForModel("vm.customer")
```

Is This Course for You?

Not for you if…

- × … you are completely new to Angular
- × …you **really** <3 JavaScript
- × …hate strong-typing
- × …hate C# (or Razor)
- × …you like copy-pasting code and markup.

But it IS for you if…

- ✓ …you want the benefits of AJS, but not the pain
- ✓ …you **really** <3 strong-typing
- ✓ …like to leverage C#'s type system
- ✓ …you aren't afraid of a little magic!

Prerequisites

Pluralsight's AngularJS Library

Courses

A-Z **Newest** Popular Rating Level

Showing 48 of 49 **View:** 48

Angular with TypeScript	by **Deborah Kurata**	Intermediate	3h 33m	10 Aug 2015
Building a Site with AngularJS and PHP	by **Christian Wenz**	Intermediate	2h 50m	16 Jul 2015
Introduction to MEAN.JS	by **Mark Scott**	Intermediate	2h 6m	04 Jun 2015
Angular Application Development	by **Lukas Ruebbelke**	Intermediate	4h 13m	04 Jun 2015
Angular: The Big Picture	by **Joe Eames**	Beginner	1h 9m	12 May 2015
Building a SPA Framework Using AngularJS	by **Mark Zamoyta**	Advanced	4h 48m	28 Apr 2015

(and new ones just about every week!)

Build Your Own Application Framework with ASP.NET MVC 5

Table of contents Expand all 9%

What Is an "Application Framework?"	14:45
The Power of an Inversion of Control Container	1:01:51
Optimize Your Controller Layer	42:03
Optimize Your View Layer	43:13
Optimize Your JavaScript	44:03

Watch it here:
http://goo.gl/9wfNKq

pluralsight

Our Sample Application

HEROIC CRM

Matt Honeycutt

- Dashboard
- Customers
- Opportunities
- Risks
- Reports

Customers

Zack Beasley

Info | $ Opportunities | Risks

E-mail	Work: zack@work.com Home: zack@home.com	**Phone**	Office: (555) 321-5444 Mobile: (555) 123-4555
Work Address	123 Zack Street Suite B New York, NY 55555	**Home Address**	321 Beasley Street Apt 1205 New York, NY 55555

Vere Rowland

Info | $ Opportunities | Risks

E-mail	Work: vere@work.com Home: vere@home.com	**Phone**	Office: (555) 321-5444 Mobile: (555) 123-4555
Work Address	123 Vere Street Suite B New York, NY 55555	**Home Address**	321 Roland Street Apt 1205 New York, NY 55555

Add New Opportunity

Enter details for the new opportunity.

Title

Title...

Add New Risk

Enter details for the new risk.

Title

Title...

Description

Heroic CRM – Based on SB Admin

SB Admin

John Smith

Dashboard
Charts
Tables
Forms
Bootstrap Elements
Bootstrap Grid
Dropdown
Blank Page
RTL Dashboard

Dashboard Statistics Overview

Dashboard

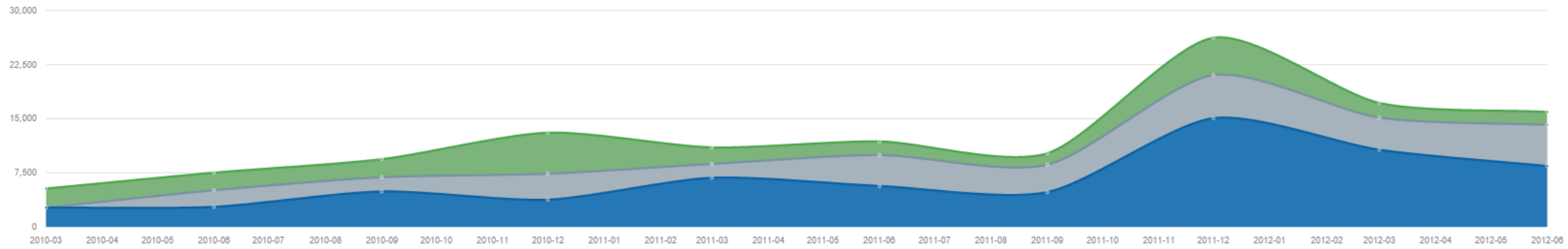
26 New Comments! — View Details

12 New Tasks! — View Details

124 New Orders! — View Details

13 Support Tickets! — View Details

Area Chart



Check it out!
http://goo.gl/I7ZZuT

Heroic CRM Is Built On…

Microsoft

ASP.NET

Entity Framework (v6)

ASP.NET MVC Futures

Dependency Injection

Heroic.Web.IoC

- Dependency injection
- Lifetime management

```
public class CustomerController : HeroicCRMControllerBase
{
    private readonly AppDbContext _context;

    public CustomerController(AppDbContext context)
    {
        _context = context;
    }
```

```
[AllowAnonymous]
public class AuthenticationController : HeroicCRMControllerBase
{
    private readonly ApplicationUserManager _userManager;
    private readonly IAuthenticationManager _authManager;

    public AuthenticationController(ApplicationUserManager userManager,
        IAuthenticationManager authManager)
    {
        _userManager = userManager;
        _authManager = authManager;
```

Mapping

autoxmapper

- Powerful object mapping
- LINQ projection

+

Heroic.AutoMapper

- Conventional configuration

```
public JsonResult All()
{
    var customerModels = _context.Customers
        .OrderByDescending(x => x.CreateDate)
        .Project().To<CustomerViewModel>();

    return Json(customerModels.ToArray());
}

public class RiskViewModel : IMapFrom<Risk>
{
    public int CustomerId { get; set; }

    public string Title { get; set; }

    public string Description { get; set; }

    public DateTime CreateDate { get; set; }

    public string CustomerName { get; set; }
}
```

Client-Side

+

```
CustomerListController.$inject = ['$modal', 'customerSvc'];
function CustomerListController($modal, customerSvc) {
    var vm = this;
    vm.add = add;
    vm.customers = customerSvc.customers;

    function add() {
        $modal.open({
            template: '<add-customer />'
        });
    }
}
```

This is UI Bootstrap!

Fun with Strings

Humanizer

```
"Long text to truncate".Truncate(10) => "Long text…"
```

```
EnumUnderTest.MemberName.Humanize() => "Member name"
```

```
DateTimeOffset.AddHours(1).Humanize() => "an hour from now"
```

Working with HTML

HtmlTags

The HtmlTags Library

```
@using (var opportunity = customer.Repeat(x => x.Opportunities, "opportunity"))
{
    <hr ng-hide="$index == 0" />
    <h3>
        @opportunity.BindingFor(x => x.Title)
    </h3>
    <p>@opportunity.BindingFor(x => x.Description)</p>
}
```

```
<div ng-repeat="opportunity in vm.customer.opportunities">
    <hr ng-hide="$index == 0" />
    <h3>
        {{opportunity.title}}
    </h3>
    <p>{{opportunity.description}}</p>
</div>
```

The HtmlTags Library

```
@Html.TextBoxFor(x => x.Name)
```

Returns a string!

```
"<input id="Name" name="Name" type="text" value="" />"
```

```
@customer.FormGroupFor(x => x.Name)
```

Returns an object!

HtmlTag { Name="div" .. , Attrs={...} }

```
<div form-group-validation="Name" class="form-group has-feedback">
    <label for="Name" class="control-label">Name</label>
    <input ng-model="vm.customer.name" name="Name"
           type="text" placeholder="Name..." class="form-control" />
</div>
```

The HtmlTags Library

```
var tag = new HtmlTag("button");
```

```
<button>
</button>
```

The HtmlTags Library

```
var tag = new HtmlTag("button");
tag.AddClasses("btn", "btn-success");
```

```
<button class="btn btn-success">
</button>
```

The HtmlTags Library

```
var tag = new HtmlTag("button");
tag.AddClasses("btn", "btn-success");
tag.Attr("type", "submit");
```

```
<button class="btn btn-success"
        type="submit">
</button>
```

The HtmlTags Library

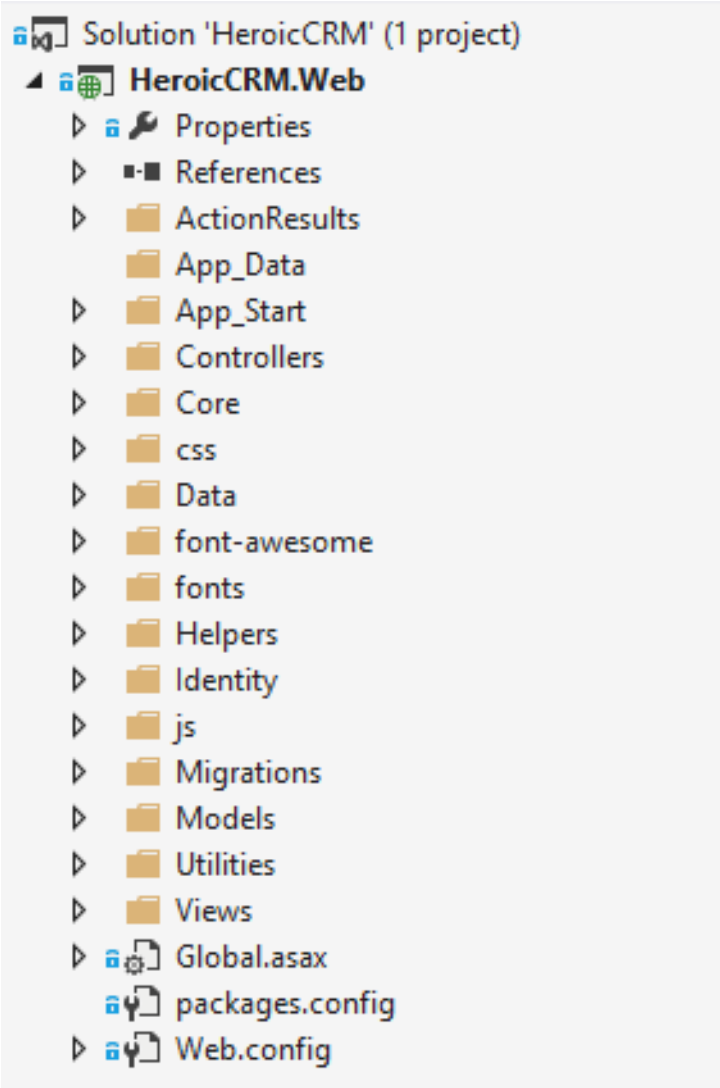
```
var tag = new HtmlTag("button");
tag.AddClasses("btn", "btn-success");
tag.Attr("type", "submit");
tag.Attr("ng-click", "vm.confirm()");
```

```
<button class="btn btn-success"
        type="submit"
        ng-click="vm.confirm()">
</button>
```

HtmlTags

More examples:
https://goo.gl/SUiZhl

The Heroic CRM Solution



Dependency Injection Everywhere

Dependencies

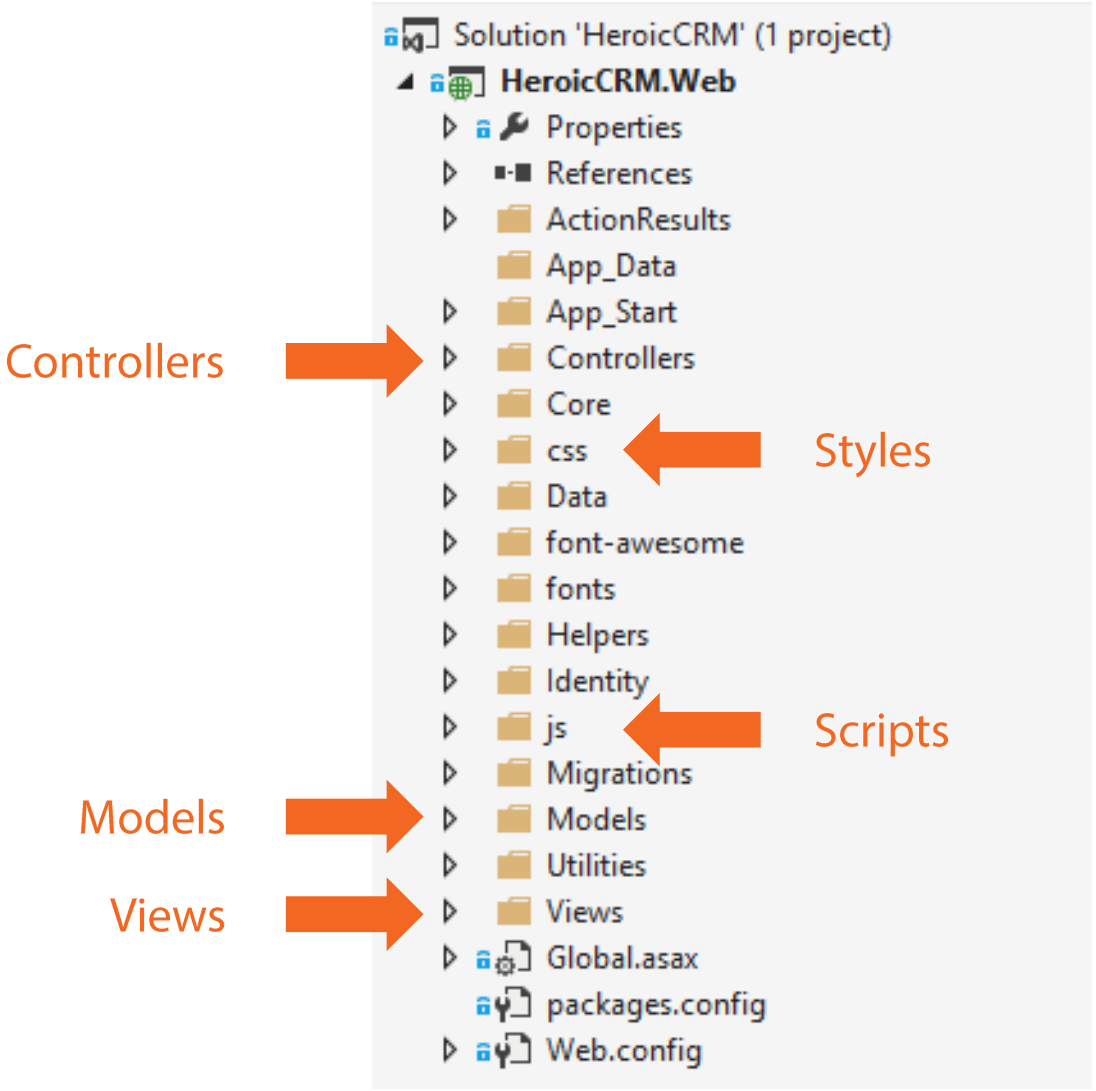
```
private readonly ApplicationUserManager _userManager;
private readonly IAuthenticationManager _authManager;

public AuthenticationController(
    ApplicationUserManager userManager,
    IAuthenticationManager authManager)
{
    _userManager = userManager;
    _authManager = authManager;
}
```

Dependency Injection Everywhere

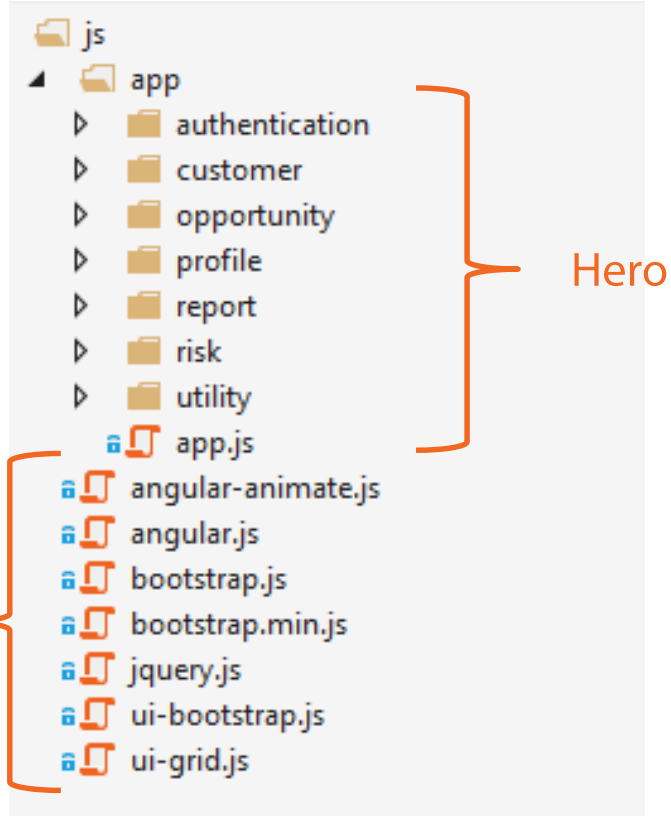
```
public abstract class HeroicCRMControllerBase : Controller
{
    //TODO: Add helpers!
}
```

```
public class CustomerController : HeroicCRMControllerBase
{
    code
}
```

Solution Structure



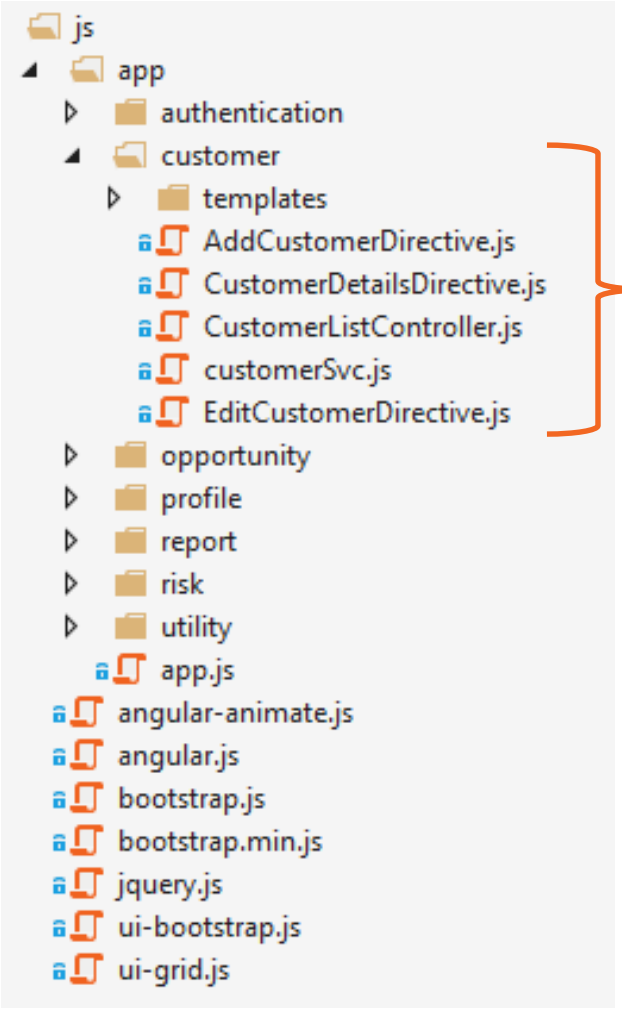
Script Organization



Heroic CRM code

3rd party libraries

Script Organization



Everything related to the "Customers" feature

Controller-As

Controller

Alias

```
<form name="vm.form" novalidate
      ng-controller="LoginController as vm"
      ng-submit="vm.login()">
    <fieldset>...</fieldset>
</form>
```

```
LoginController.$inject = ['$window', '$http'];
function LoginController($window, $http) {
    var vm = this;

    Code
}
```

Minification-friendly

Alias

Course Roadmap

Module 1: Introduction

Module 2: Assembling the Building Blocks

Module 3: Strongly-Typed Binding

Module 4: Building Strongly-Typed Forms with AngularJS

Module 5: Supporting Angular Validation with Data Annotations

Module 6: Building Angular-Powered Templates

Module 7: Building Reusable Directives and Helpers

Module 8: Closing Thoughts

Up Next…

Building the foundation!