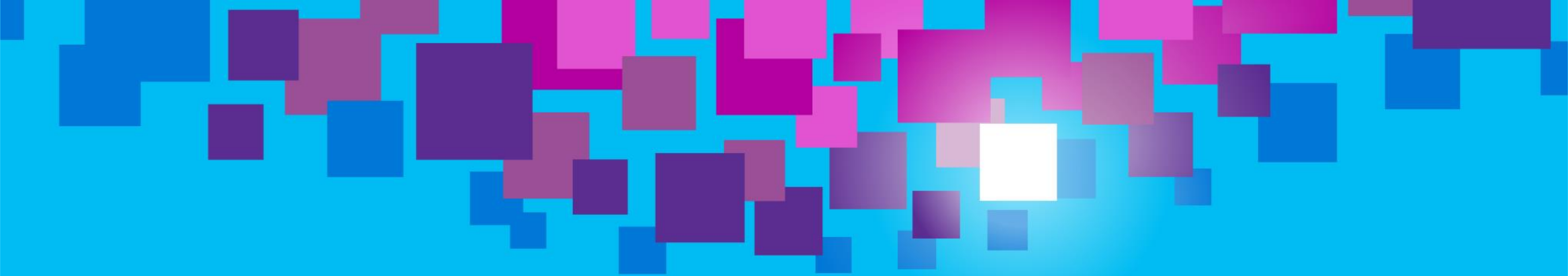


Spark the future.

Microsoft **Ignite**

May 4 – 8, 2015
Chicago, IL

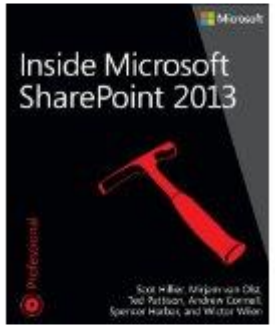
Microsoft

Office 365 Developer On Ramp (Part 1)

Scot Hillier
Scot Hillier Technical Solutions, LLC
scot@scothillier.net
@ScotHillier

Microsoft

Scot Hillier



LOOK INSIDE!
Professional Business Connectivity Services in SharePoint 2010

Scot Hillier Technical Solutions, LLC

CriticalPath TRAINING

pluralsight hardcore developer training

MVP Microsoft Most Valuable Professional

scot@scothillier.net
@ScotHillier

Agenda

Enterprise JavaScript

JavaScript
TypeScript

Enterprise Services Architecture

REST and OData
Promises
Web API

Enterprise Frameworks

Angular
Bootstrap

Enterprise JavaScript

JavaScript

Key Concept: JavaScript is Object Based

An object is a set of key-value pairs…

```
var emptyObject = {};
var person = { firstName: "Scot", lastName: "Hillier" };
```

…which can be accessed using dot notation

```
alert(person.lastname);
```

…and can have values assigned dynamically

```
person.middleName = "Patrick";
```

Functions are also objects…

```
var myfunc = function(){};
```

…and can return objects.

```
var person = function(fn, ln) { return { firstName: fn, lastName: ln}; }();
```

The Browser DOM is a collection of objects

```
window.document
```

Always

Use “strict” mode
Encapsulate your code
Minify and bundle libraries

Use Strict Mode

Declared with "use strict";

Cannot use a variable without declaring it
Cannot write to a read-only property
Cannot add properties to non-extensible objects
Cannot illegally delete functions and variables
Cannot define a property more than once in an object literal
Cannot use a parameter name more than once in a function
Cannot use reserved words, eval, or arguments, as names for functions and variables
The value of this in a function is no longer the window object
Cannot declare functions inside of statements
Cannot change the members of the arguments array

Encapsulate Your Code

Anonymous functions
Singleton Pattern
Revealing Module Pattern
Prototype Pattern

Anonymous Functions

```
(function () {

}());
```

Using the Singleton Pattern

```
var Wingtip = window.Wingtip || {};

Wingtip.Customer = {
     name: "Brian Cox",
     speak: function () {
         alert("My name is " + this.name);
     }
 };

 Wingtip.Customer.speak();
```

Using the Revealing Module Pattern

```
var Wingtip = window.Wingtip || {};
Wingtip.Module = Wingtip.Module || {};

Wingtip.Module.Customer = function () {
     //private members
     var name,
         setname = function (n) { name = n; },
         getname = function () { return name; },
         talk = function () { alert("My name is " + name); };
     //public interface
     return {
         set_name: setname,
         get_name: getname,
         speak: talk
    }
 }();
```

Using the Prototype Pattern

```
window.Wingtip = window.Wingtip || {};

Wingtip.Customer = function (n) {
     this.name = n
};

Wingtip.Customer.prototype = {
     get_name: function () { return this.name; },
     set_name: function (n) { this.name = n; },
     speak: function () { alert("My name is " + this.name);
 }
};

 var customer = new Wingtip.Customer("Brian Cox");
Customer.speak();
```

Minify and Bundle Libraries

Minification

Removes unnecessary characters and white space to minimize the size of the library

Bundling

Groups multiple libraries into a single library because it is more efficient to make one larger request than multiple small requests.

Make use of the “Web Essentials” add-in for Visual Studio

Traps

Global variables

Semicolon insertion

Coercive equality operators

==, != are coercive, ===, !== are not coercive

parseInt

Leading zero means octal, parseInt(“08”), which causes problems for dates

Decimal fractions

0.1 + 0.2 !== 0.3

Testing for numbers

typeof(myvar) === “number” && isFinite(myvar)

DEMO

JavaScript Best Practices

Microsoft

Agenda

Enterprise JavaScript

~~JavaScript~~
TypeScript

Enterprise Services Architecture

REST and OData
Promises
Web API

Enterprise Frameworks

Angular
Bootstrap

Enterprise JavaScript

TypeScript

Introduction to TypeScript

Typed superset of JavaScript that compiles to plain JavaScript

You write .ts files and it compiles to .js files
Cross-browser compatible

Integrated into Visual Studio 2013

Compilation
Intellisense

Key Features

Static typing
Classes, constructors, properties, methods
Modules
Interfaces

Static Typing

```
private getQueryStringParameter(p: string): string { ... };
```

scope

Input type

Return type

```
private displayName: string = "Scot";
```

scope

type

Modules

```
module Wingtip { … }
```

Classes

```
class Welcome {
     //private members
     private displayName: string = "Scot Hillier";
     private pictureUrl: string = "/images/sh.jpg";
     //public methods     public get_viewModel() {
        return {
             "pictureUrl": Welcome.pictureUrl,
             "displayName": Welcome.displayName
         };
     }
 }
```

Interfaces

```
module Wingtip { //Namespace
     interface WelcomeData {
         pictureUrl: string;
         displayName: string;
     }
     export class Welcome {
         public get_viewModel(): WelcomeData {
             return {
                 "pictureUrl": Welcome.pictureUrl,
                 "displayName": Welcome.displayName
             };
         }
     }
```

Define Interface

Implement Interface

DEMO

TypeScript 101

Agenda

Enterprise JavaScript

~~JavaScript~~
~~TypeScript~~

Enterprise Services Architecture

REST and OData
Promises
Web API

Enterprise Frameworks

Angular
Bootstrap

Enterprise Services Architecture

REST and OData

Representational State Transfer

In the beginning there was SOAP

XML-based protocol for executing web service operations
SOAP = Simple Object Access Protocol
SOAP makes simple things more complicated than they could be
Acronym status of SOAP revoked in 2003

REST is simpler and much easier to use

REST = REpresentational State Transfer
Simple approach based on HTTP request/response pairs
HTTP requests target specific resources using unique URIs
Resources move back and forth using representations
Representations of resources defined using Internet Media Types

REST Constraints

Client-Server

Client pulls representations from the server
Separation of concerns

Stateless

Client provides all necessary context
Server returns all necessary state

Cache

Responses indicate whether or not they can be cached
eTag, Date, Expires headers

Interface

Resources are accessible through URIs
Resources operations are through HTTP verbs
The same representations can be used for all operations
Resources are interconnected to allow linking

Layered

Resources are unaffected by proxy servers, gateways, etc.

RESTful Web Services

RESTful Web Service

implemented using the principles of REST
REST URI = [base URI] + [resource path] + [query options]
Calls based on standard HTTP verbs (GET, POST, PUT, DELETE)
Passes data to and from client using representations
Can be designed to implement custom APIs and/or standard APIs

Data passed across network using representations

Representations model resources – but they're different
Based on common formats: HTML, XML, ATOM and JSON
Based on specific Internet media types

Internet Media Types

Internet media type defines format of representation

text/html
text/xml
application/xml
application/atom+xml
application/json

HTTP headers used to indicate Internet Media Type

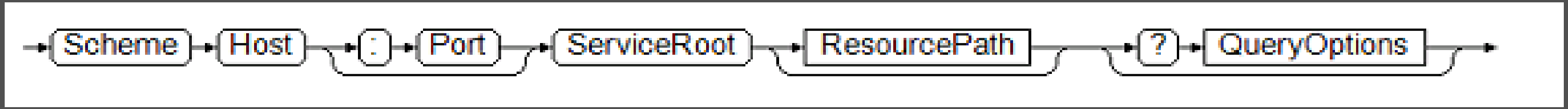
Accept request header indicates what client wants in response
Content-Type header indicates type of request/response body

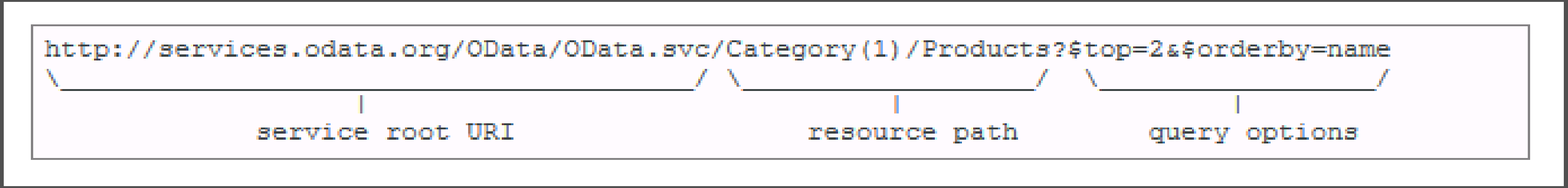
Open Data Protocol (OData)

Standardized REST API for CRUD operations
Standardized Data Types

```
<Property Name="Id" Type="Edm.Guid" Nullable="false"/>
<Property Name="Title" Type="Edm.String"/>
<Property Name="TreeViewEnabled" Type="Edm.Boolean" Nullable="false"/>
<Property Name="UIVersion" Type="Edm.Int32" Nullable="false"/>
```

Standardized URI format





```
http://services.odata.org/OData/OData.svc/Category(1)/Products?$top=2&$orderby=name
\______________________________________/ \________________/  \________________/
                   |                              |                    |
            service root URI                resource path         query options
```

OData Entity Model

Service Document

$metadata

Entity Types define entities

```
<EntityType Name="Site">
<EntityType Name="Web" BaseType="SP.SecurableObject">
<EntityType Name="List" BaseType="SP.SecurableObject">
<EntityType Name="ListItem" BaseType="SP.SecurableObject" OpenType="true">
```

Entity Key defines unique property

```
 <Key><PropertyRef Name="Id"/></Key>
```

Associations link entities together

```
<NavigationProperty Name="RootWeb" …
```

OData Query Options

$select
$filter
$orderby
$top
$skip
$expand

DEMO

REST

Agenda

Enterprise JavaScript

~~JavaScript~~
~~TypeScript~~

Enterprise Services Architecture

~~REST and OData~~
Promises
Web API

Enterprise Frameworks

Angular
Bootstrap

Enterprise Services Architecture

Promises

AJAX Request

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function () {
     if (xmlhttp.readyState === 4 && xmlhttp.status === 200) {
         var results = JSON.parse(xmlhttp.responseText).d.results;
              for (var i = 0; i < results.length; i++) {
             var title = results[i].Title;
         }
     };
var url = "../_api/web/lists/getByTitle('Contacts')/items";
xmlhttp.open("GET", url, false);
xmlhttp.setRequestHeader("accept", "application/json;odata=verbose");
xmlhttp.send();
```

jQuery AJAX Request

```
jQuery.ajax({
       url: "../_api/web/lists/getByTitle('Contacts')/items",
       type: "GET",
       headers: {
           "accept": "application/json",
       },
       success: function (data, status, jqXHR) { },
       error: function (jqXHR, status, message) { }
});
```

- jqXHR is a superset of XMLHttpRequest
- status is a string
- message is a string
- data is a JSON object

Adding Items to a SharePoint List

```
jQuery.ajax({
     url: "../_api/web/lists/getByTitle('Contacts')/items",
     type: "POST",
     contentType: "application/json",
     data: JSON.stringify({
           '__metadata': { 'type': 'SP.Data.ContactsListItem' },
           'Title': lname,
           'FirstName': fname,
           'WorkPhone': wphone,
           'Email': email
       }),
     headers: {
        "accept": "application/json;",
         "X-RequestDigest": jQuery("#__REQUESTDIGEST").val()
     },
     success: function (data, status, jqXHR) { },
     error: function (jqXHR, status, message) { }
 });
```

Request Digest

ETags and Optimistic Concurrency

OData v2 requires items to carry ETags

ETag is integer value in that it identities version of item
ETag is automatically incremented with each update

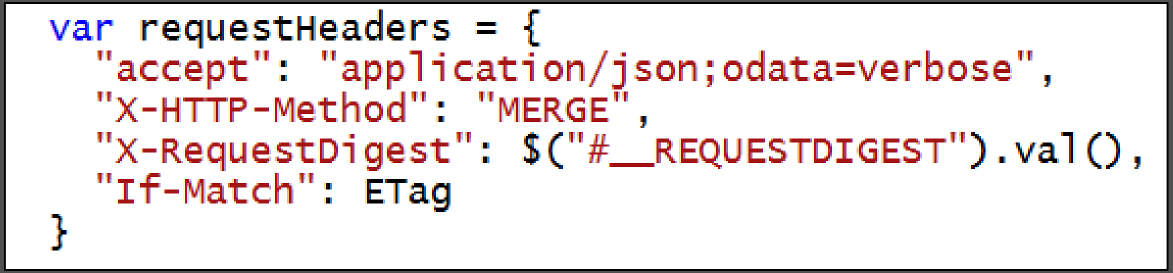
ETag use to support for optimistic concurrency control

ETag works to eliminate the “lost update” scenario
ETag must be tracked in order to post updates in most scenarios

ETags and the If-Match Header

Update and Delete operations require If-Match Header

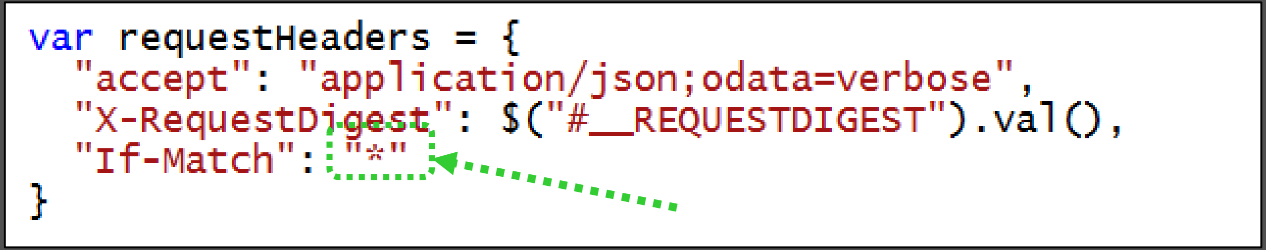
Allows you to pass ETag value during an update
Update fails if ETag value changed due to update by other user



```
var requestHeaders = {
  "accept": "application/json;odata=verbose",
  "X-HTTP-Method": "MERGE",
  "X-RequestDigest": $("#__REQUESTDIGEST").val(),
  "If-Match": ETag
}
```

You can pass wildcard (*) value inside If-Match Header

Done to disable optimistic concurrency control
This is commonly done with delete operations



```
var requestHeaders = {
  "accept": "application/json;odata=verbose",
  "X-RequestDigest": $("#__REQUESTDIGEST").val(),
  "If-Match": "*"
}
```

Updating a SharePoint List Item

```
jQuery.ajax({
     url: "../_api/web/lists/getByTitle('Contacts')/items(" + Id + ")",
     type: "POST",
     contentType: "application/json ",
     data: JSON.stringify({ __metadata': { 'type': 'SP.Data.ContactsListItem' },
           'FirstName': fname
       }),
     headers: {
         "accept": "application/json",
         "X-HTTP-Method": "MERGE",
         "X-RequestDigest": jQuery("#__REQUESTDIGEST").val(),
         "If-Match": eTag     },
     success: function (data, status, jqXHR) { },
     error: function (jqXHR, status, message) { } });
```

eTag

Deleting a SharePoint List Item

```
jQuery.ajax({
     url: "../_api/web/lists/getByTitle('Contacts')/items(" + Id + ")",
     type: "DELETE",
     headers: {
         "accept": "application/json",
         "X-RequestDigest": jQuery("#__REQUESTDIGEST").val(),
         "If-Match": "*“
     },
     success: function (data, status, jqXHR) { },
     error: function (jqXHR, status, message) { } });
```

Promises

jQuery "Deferred"

Automatically returned from jQuery ajax method
Callbacks include jqXHR object, which contains promise

Angular "Q"

Used with the Angular framework

Allow for Orchestration of multiple calls

Sequence calls
Batch calls

Promises in jQuery

```
function myPromise() {
     var deferred = $.Deferred();
     setTimeout(function () {
         deferred.resolve("success!");
     }, 1000);
     return deferred.promise();
 }
myPromise().then(
     function (value) {
         alert(value);
     },
     function () {
         alert("error!");
     }
 );
```

Success

Failure

1 Create a Deferred

3 Resolve or Reject

2 Return the Promise

DEMO

Promises

Microsoft

Agenda

Enterprise JavaScript

~~JavaScript~~
~~TypeScript~~

Enterprise Services Architecture

~~REST and OData~~
~~Promises~~
Web API

Enterprise Frameworks

Angular
Bootstrap

Enterprise Services Architecture

Web API

Introducing WebAPI

- Part of ASP.NET MVC
- Use the same Controller and Routing paradigm
- Simplified creation of REST services
  - GET, POST, PUT, DELETE operations
  - JSON, XML returned
- Can be a stand-alone service or part of an app
- Supports using Entity Framework to wrap database calls

Controllers

```
public class CustomersController : EntitySetController<Customer, int> {
     List<Customer> customers = new List<Customer>()
     {
         new Customer() { Id=1, LastName="Doyle", FirstName="Patricia" },
         new Customer() { Id=2, LastName="Burke", FirstName="Brian" }
     };
     [Queryable]
     public override IQueryable<Customer> Get()
     {
         return customers.AsQueryable();
     }
     protected override Customer GetEntityByKey(int Id)
     {
         return (from c in customers where c.Id == Id select c).FirstOrDefault();
     }
 }
```

Routes

```
ODataModelBuilder builder = new ODataConventionModelBuilder();
builder.EntitySet<Customer>("Customers");
IEdmModel model = builder.GetEdmModel();
config.Routes.MapODataRoute("CustomersRoute", "odata", model);
```

Consuming Web API

```
$.ajax(
        {
            url: "http://webs.wingtip.com/SimpleOdata/odata/Customers(" + id + ")",
            type: "GET",
            headers: {
                "accept": "application/json",
            },
            success: onSuccess,
            error: onError
        }
 );
```

Beware of cross-origin calls!

DEMO

WebAPI

Microsoft

Agenda

Enterprise JavaScript

~~JavaScript~~
~~TypeScript~~

Enterprise Services Architecture

~~REST and OData~~
~~Promises~~
~~Web API~~

Enterprise Frameworks

Angular
Bootstrap

JavaScript Frameworks

Angular

Fundamentals

Introducing AngularJS

Description

Single-Page Application (SPA) Framework
Implements Model-View-Controller (MVC) Pattern

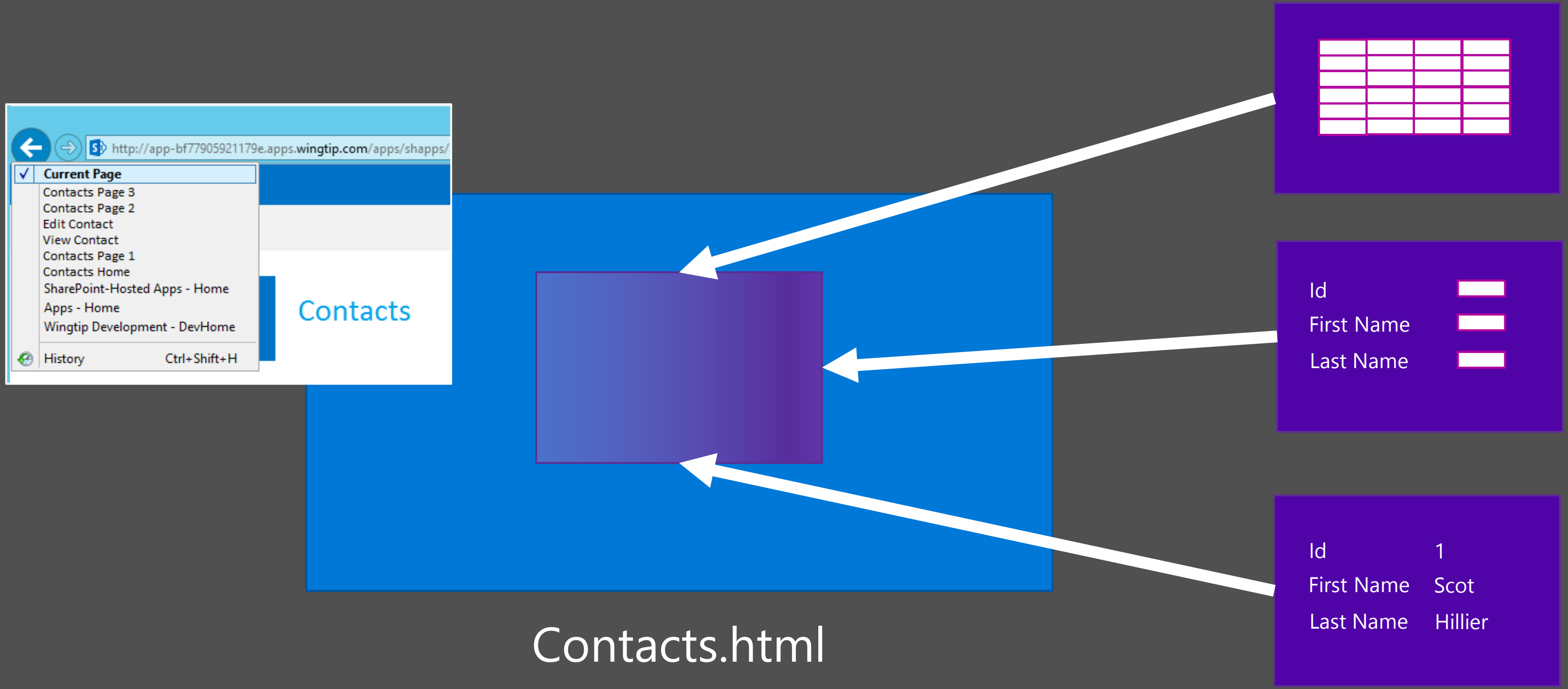
Why Angular

True framework instead of patchwork of libraries
Strong separation of concerns

Single-Page Applications

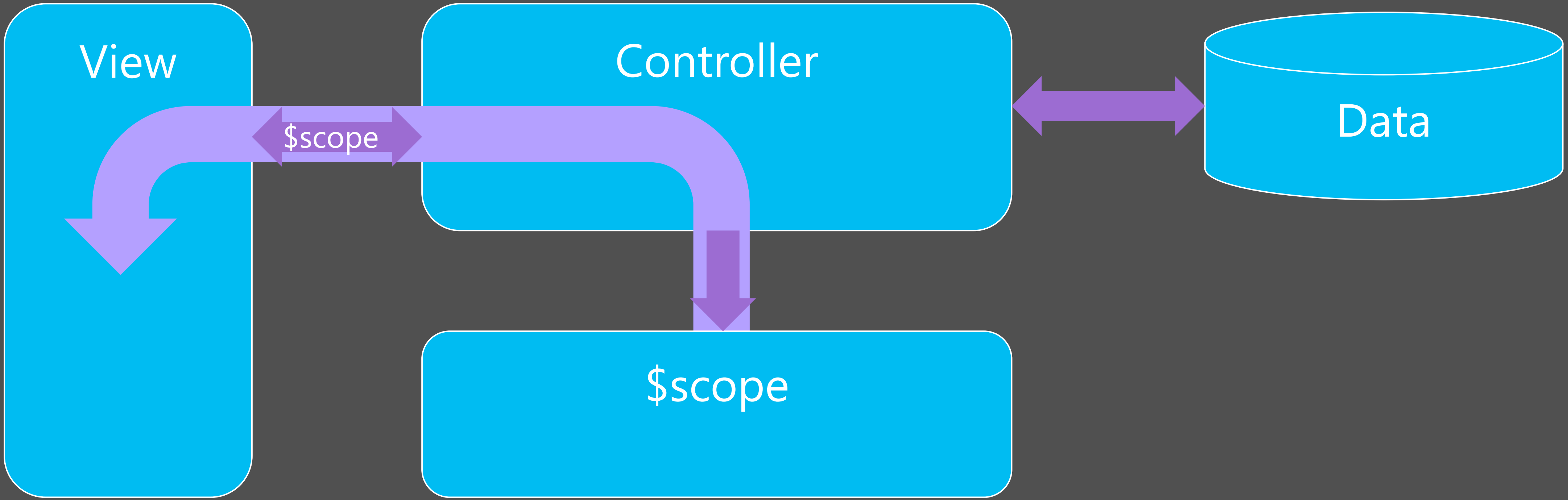
App has one page

Different views are loaded dynamically
Routes are used to simulate pages
History list reflects route navigation



Contacts.html

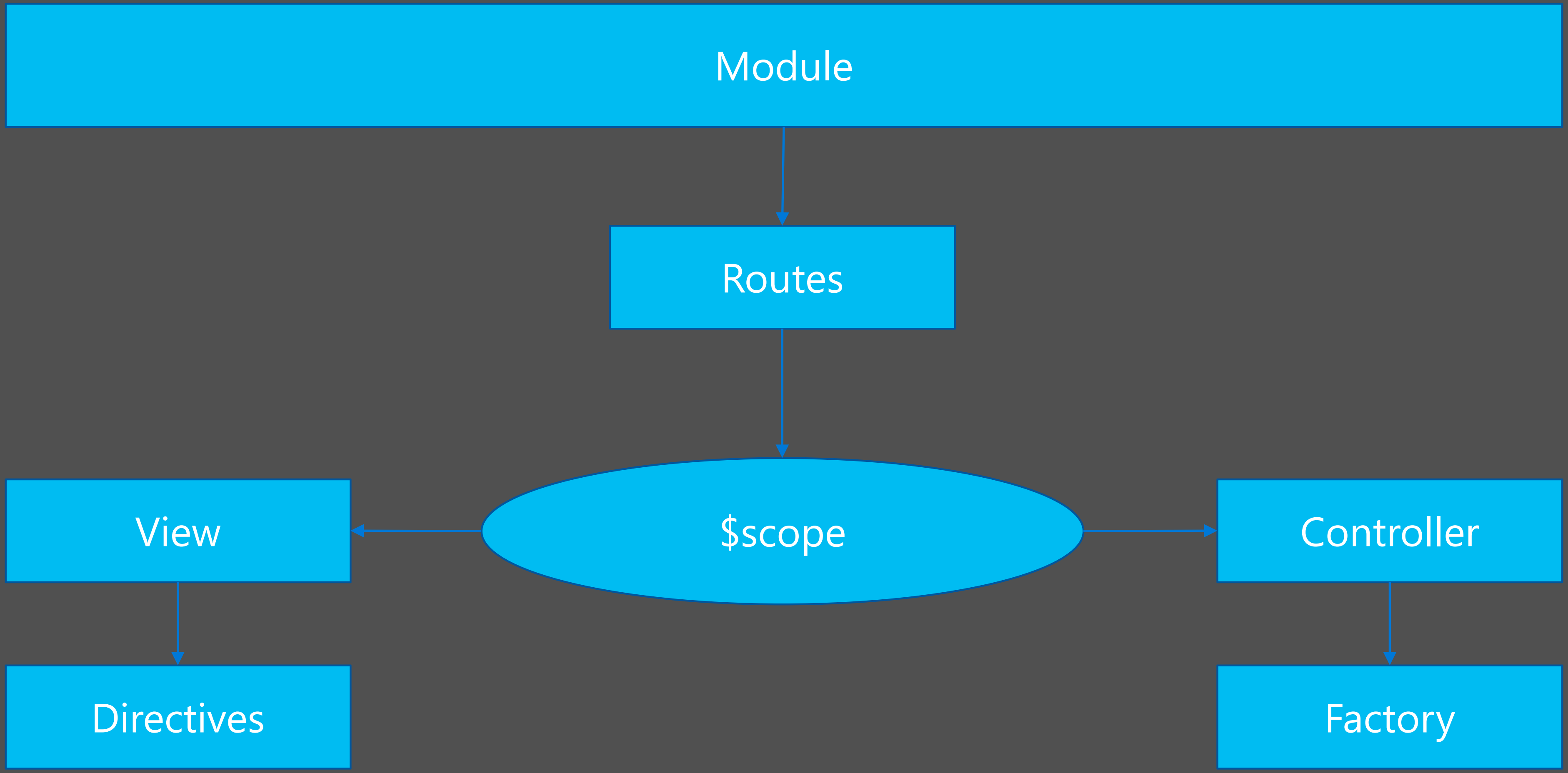
Model-View-Controller with Angular



Angular Features

Utilizes “Modules” to define scope
“Injects” scope into Controllers to maintain context
“Scope” is used to create and maintain the View
“Services” and “Directives” for app-wide functionality
View is bound declaratively to HTML

Angular Framework



Modules and Routes

Modules

A container for the components of the app

```
//module
var myapp = angular.module("MyApp", []);


<!-- html -->
<div data-ng-app="MyApp">
```

Routes

- Used for loading different partial views in a SPA
- Angular manages history automatically
- HTML chunks make up the partial views
  - Views can be embedded as a script template in the SPA
  - Views can also be kept as separate partial HTML pages
- Defining Routes
  - Add Angular JS Route NuGet Package
  - Reference the ngRoute Module
  - Define routes using the $routeProvider

Defining Routes

```
//module
Wingtip.App = angular.module("App", ["ngRoute"]);
 Wingtip.App.config(["$routeProvider",
     function ($routeProvider) {
         $routeProvider.
             when("/welcome", {
                 templateUrl: "partials/welcome.html",
                 controller: "welcomeCtrl“
             }).
             otherwise({
                 redirectTo: "/“
             });
     }] );
```

Route module

Partial page

```
   <!-- HTML -->
   <div data-ng-view> </div>
```

Rendered here

Directives

Directives

Utilizes HTML5 custom data attributes

Allows for the addition of custom attributes starting with data-
Angular uses directives for declarative programming

Angular directives start with “ng-”

data-ng-app, defines scope of the framework
data-ng-controller, invokes a controller
data-ng-click, handles click event

Key Directives

`data-ng-app:` initialize the Angular app

`data-ng-controller:` designate controller scope

`data-ng-repeat:` for-each loop

`data-ng-cloak:` `hides` elements during app initialization

`data-ng-hide:` shows or hides an HTML element

`data-ng-href:` creates Angular-compliant anchor tags

`data-ng-src:` creates Angular-compliant img tags

`data-ng-click:` handles click event

Using Angular Directives

Initializes the app. Can be anonymous or named.

Creates a property on the ViewModel

```
<!DOCTYPE html>
 <html data-ng-app>
 <head> </head>
 <body>
     <input type="text" data-ng-model="displayName" />
     <div data-ng-click="update" ng-controller="myCtrl">
     </div>
 </body>
 </html>
```

References a controller method to call on a click event

References a controller named “myCtrl”, which creates a new ViewModel.

Data Binding

- Binds ViewModels to HTML elements
  - Uses {{…}} syntax
  - References a property of a ViewModel
  - Supports two-way binding

```
<div ng-app="App">
     <div>
         <input type="text" data-ng-model="firstName" />
         <div>{{firstName}}</div>
     </div>
 </div>
```

Display whatever the user types

Filters

Perform common operations on data bound elements

- Takes the form of {{ expression | filter }}

```
<div ng-app="App">
     <div>
         <input type="text" data-ng-model="firstName" />
         <div>{{firstName | uppercase}}</div>
     </div>
 </div>
```

Display data in all caps

Key Filters

Format

currency
date
number

Displaying data sets

orderBy
limitTo

String manipulation

uppercase
lowercase

Controllers

Controllers

Build up the `$scope` (a.k.a, View Model)

```
//controller
 myapp.controller("welcomeCtrl", ["$scope",
     function welcomeCtrl($scope) {
         //model
         $scope.welcomeMessage = "Hi";
     }]
 );
```

```
<!-- html -->
<div data-ng-controller="welcomeCtrl">
```

View Binding

Bind the `$scope` to the HTML elements

```
<div ng-app="App">
     <div ng-controller="welcomeCtrl">
         <h3>{{welcomeMessage}}</h3>
     </div>
 </div>
```

Services

Understanding Services

Allows common functionality to be factored out into a single component and used by many Controllers

Defined by the Module in the same way Controllers are defined

```
Wingtip.App.factory("welcomeService", function () {
     var welcomeService = {};
     welcomeService.greet = function () {
         alert("Hi!");     };
     return welcomeService; }
);
```

The new Factory is injected into Controllers

```
Wingtip.App.controller("myCtrl", ["$scope", "welcomeService",
function contactsCtrl($scope, welcomeService) {
     welcomeService.greet();
 }] );
```

DEMO

Angular

Agenda

Enterprise JavaScript

~~JavaScript~~
~~TypeScript~~

Enterprise Services Architecture

~~REST and OData~~
~~Promises~~
~~Web API~~

Enterprise Frameworks

~~Angular~~
Bootstrap

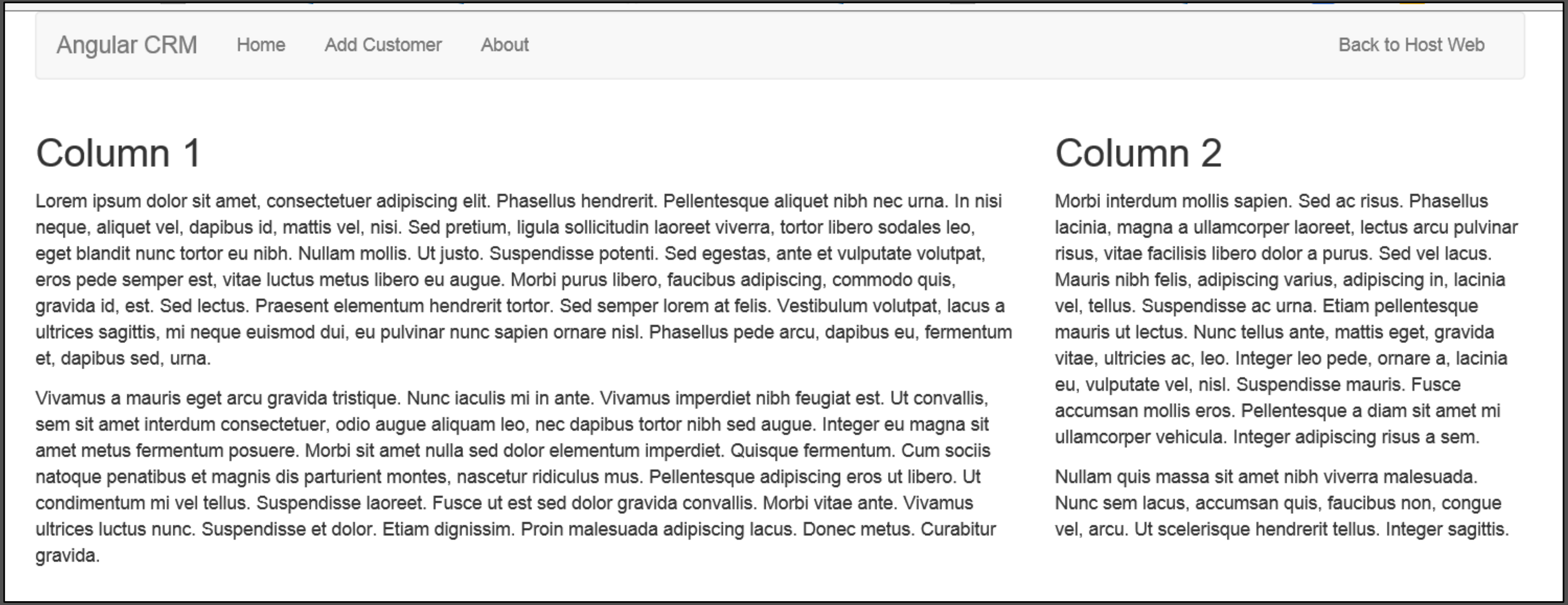
JavaScript Frameworks

Bootstrap

What is Bootstrap?

A CSS-based Framework for faster web development

Predefined classes for page layout with navbars, columns, forms, tables, etc.
Provides much faster way to create HTML-based user interfaces
Very good for creating mobile-friendly layouts



Column 1

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti. Sed egestas, ante et vulputate volutpat, eros pede semper est, vitae luctus metus libero eu augue. Morbi purus libero, faucibus adipiscing, commodo quis, gravida id, est. Sed lectus. Praesent elementum hendrerit tortor. Sed semper lorem at felis. Vestibulum volutpat, lacus a ultrices sagittis, mi neque euismod dui, eu pulvinar nunc sapien ornare nisl. Phasellus pede arcu, dapibus eu, fermentum et, dapibus sed, urna.

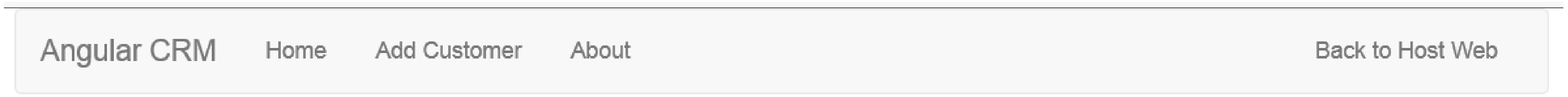
Vivamus a mauris eget arcu gravida tristique. Nunc iaculis mi in ante. Vivamus imperdiet nibh feugiat est. Ut convallis, sem sit amet interdum consectetuer, odio augue aliquam leo, nec dapibus tortor nibh sed augue. Integer eu magna sit amet metus fermentum posuere. Morbi sit amet nulla sed dolor elementum imperdiet. Quisque fermentum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque adipiscing eros ut libero. Ut condimentum mi vel tellus. Suspendisse laoreet. Fusce ut est sed dolor gravida convallis. Morbi vitae ante. Vivamus ultrices luctus nunc. Suspendisse et dolor. Etiam dignissim. Proin malesuada adipiscing lacus. Donec metus. Curabitur gravida.

Column 2

Morbi interdum mollis sapien. Sed ac risus. Phasellus lacinia, magna a ullamcorper laoreet, lectus arcu pulvinar risus, vitae facilisis libero dolor a purus. Sed vel lacus. Mauris nibh felis, adipiscing varius, adipiscing in, lacinia vel, tellus. Suspendisse ac urna. Etiam pellentesque mauris ut lectus. Nunc tellus ante, mattis eget, gravida vitae, ultricies ac, leo. Integer leo pede, ornare a, lacinia eu, vulputate vel, nisl. Suspendisse mauris. Fusce accumsan mollis eros. Pellentesque a diam sit amet mi ullamcorper vehicula. Integer adipiscing risus a sem.

Nullam quis massa sit amet nibh viverra malesuada. Nunc sem lacus, accumsan quis, faucibus non, congue vel, arcu. Ut scelerisque hendrerit tellus. Integer sagittis.

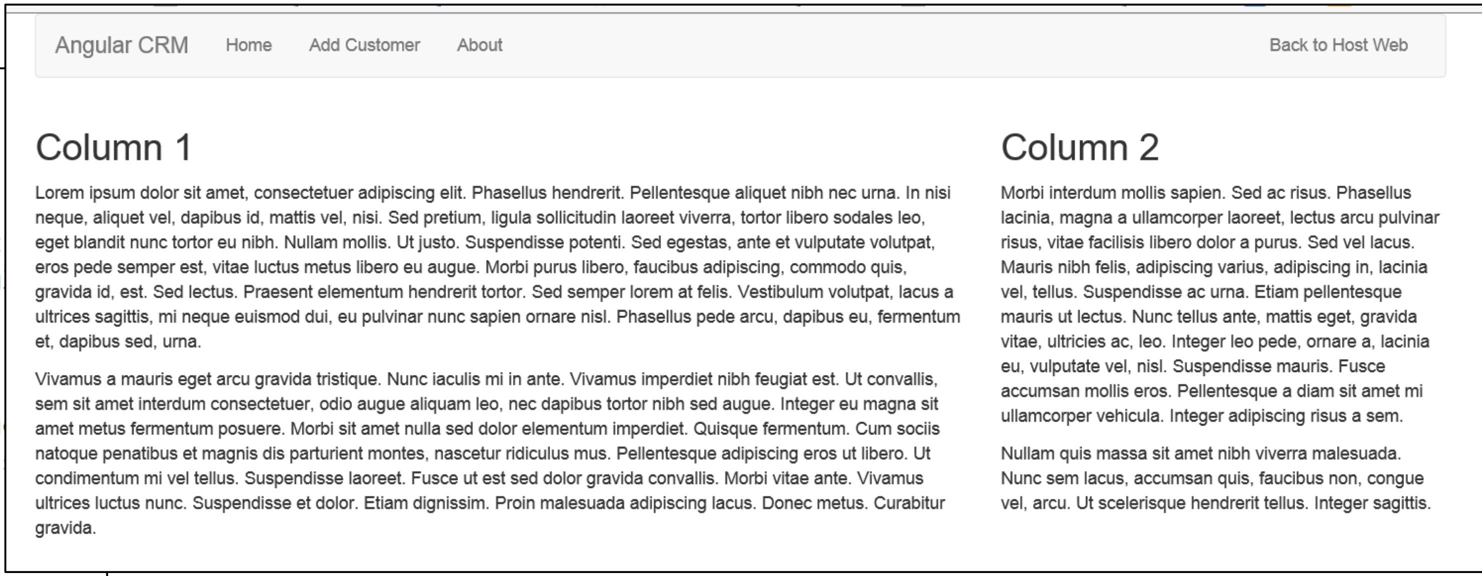
Creating a Navbar using Bootstrap



```
<div class="container">
    <div class="navbar navbar-default" role="navigation">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="#">Angular CRM</a>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="#">Home</a></li>
                    <li><a href="#/new">Add Customer</a></li>
                    <li><a href="#/about">About</a></li>
                </ul>
                <ul class="nav navbar-nav navbar-right">
                    <li><a id="lnkHostWeb">Back to Host Web</a></li>
                </ul>
            </div>
        </div>
    </div>
</div>
```

Grid Layout

```
<div class="container">
    <div class="row">
        <div class="col-md-8">
            <h2>Column 1</h2>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit
            <p> Vivamus a mauris eget arcu gravida tristique. Nunc iacu
        </div>
        <div class="col-md-4">
            <h2>Column 2</h2>
            <p>Morbi interdum mollis sapien. Sed ac risus. Phasellus la
            <p>Nullam quis massa sit amet nibh viverra malesuada. Nunc
        </div>
    </div>
</div>
```



Column 1

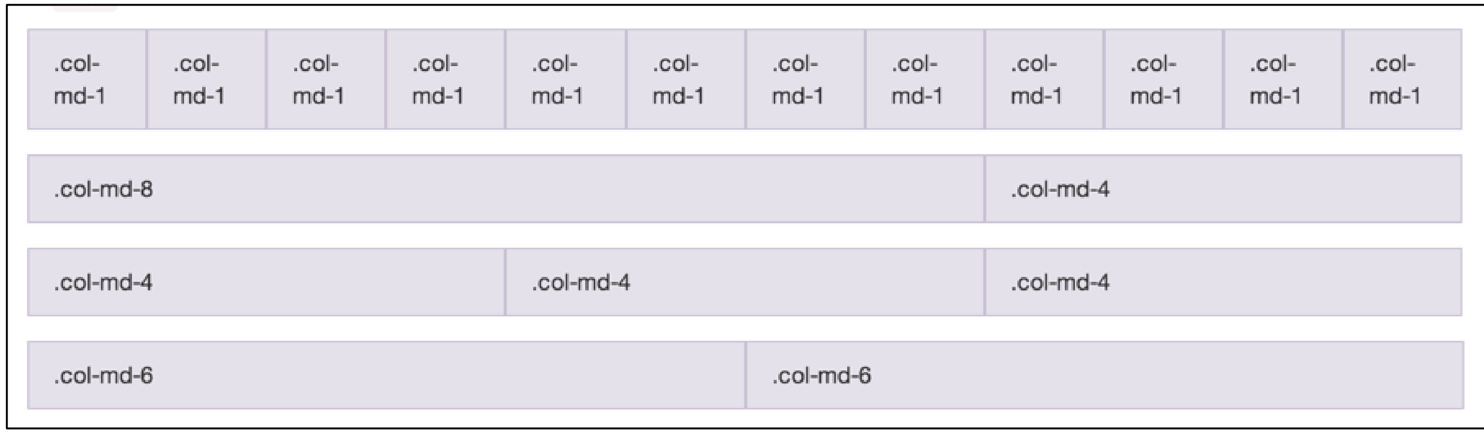
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus hendrerit. Pellentesque aliquet nibh nec urna. In nisi neque, aliquet vel, dapibus id, mattis vel, nisi. Sed pretium, ligula sollicitudin laoreet viverra, tortor libero sodales leo, eget blandit nunc tortor eu nibh. Nullam mollis. Ut justo. Suspendisse potenti. Sed egestas, ante et vulputate volutpat, eros pede semper est, vitae luctus metus libero eu augue. Morbi purus libero, faucibus adipiscing, commodo quis, gravida id, est. Sed lectus. Praesent elementum hendrerit tortor. Sed semper lorem at felis. Vestibulum volutpat, lacus a ultrices sagittis, mi neque euismod dui, eu pulvinar nunc sapien ornare nisl. Phasellus pede arcu, dapibus eu, fermentum et, dapibus sed, urna.

Vivamus a mauris eget arcu gravida tristique. Nunc iaculis mi in ante. Vivamus imperdiet nibh feugiat est. Ut convallis, sem sit amet interdum consectetuer, odio augue aliquam leo, nec dapibus tortor nibh sed augue. Integer eu magna sit amet metus fermentum posuere. Morbi sit amet nulla sed dolor elementum imperdiet. Quisque fermentum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque adipiscing eros ut libero. Ut condimentum mi vel tellus. Suspendisse laoreet. Fusce ut est sed dolor gravida convallis. Morbi vitae ante. Vivamus ultrices luctus nunc. Suspendisse et dolor. Etiam dignissim. Proin malesuada adipiscing lacus. Donec metus. Curabitur gravida.

Column 2

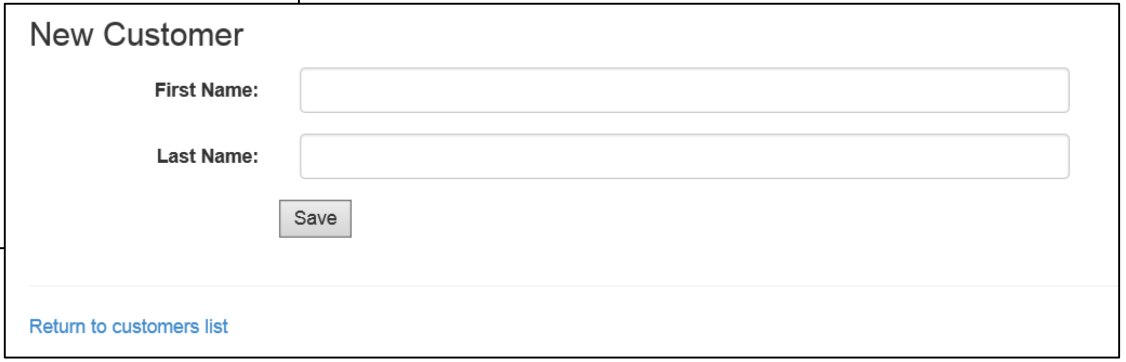
Morbi interdum mollis sapien. Sed ac risus. Phasellus lacinia, magna a ullamcorper laoreet, lectus arcu pulvinar risus, vitae facilisis libero dolor a purus. Sed vel lacus. Mauris nibh felis, adipiscing varius, adipiscing in, lacinia vel, tellus. Suspendisse ac urna. Etiam pellentesque mauris ut lectus. Nunc tellus ante, mattis eget, gravida vitae, ultricies ac, leo. Integer leo pede, ornare a, lacinia eu, vulputate vel, nisl. Suspendisse mauris. Fusce accumsan mollis eros. Pellentesque a diam sit amet mi ullamcorper vehicula. Integer adipiscing risus a sem.

Nullam quis massa sit amet nibh viverra malesuada. Nunc sem lacus, accumsan quis, faucibus non, congue vel, arcu. Ut scelerisque hendrerit tellus. Integer sagittis.



Bootstrap Forms

```
<div class="container">
    <div class="row">
        <h3>New Customer</h3>
        <div class="form-horizontal">
            <fieldset>
                <div class="form-group">
                    <label for="txtFirstName" class="col-lg-2 control-label">First Name:</label>
                    <div class="col-lg-6">
                        <input id="txtFirstName" type="text" class="form-control" >
                    </div>
                </div>
                <div class="form-group">
                    <label for="txtLastName" class="col-lg-2 control-label">Last Name:</label>
                    <div class="col-lg-6">
                        <input id="txtLastName" type="text" class="form-control" >
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-lg-offset-2">
                        <input id="cmdSave" type="button" class="button" value="Save"  />
                    </div>
                </div>
            </fieldset>
        </div>
        <hr />
        <a href="#/">Return to customers list</a>
    </div>
</div>
```

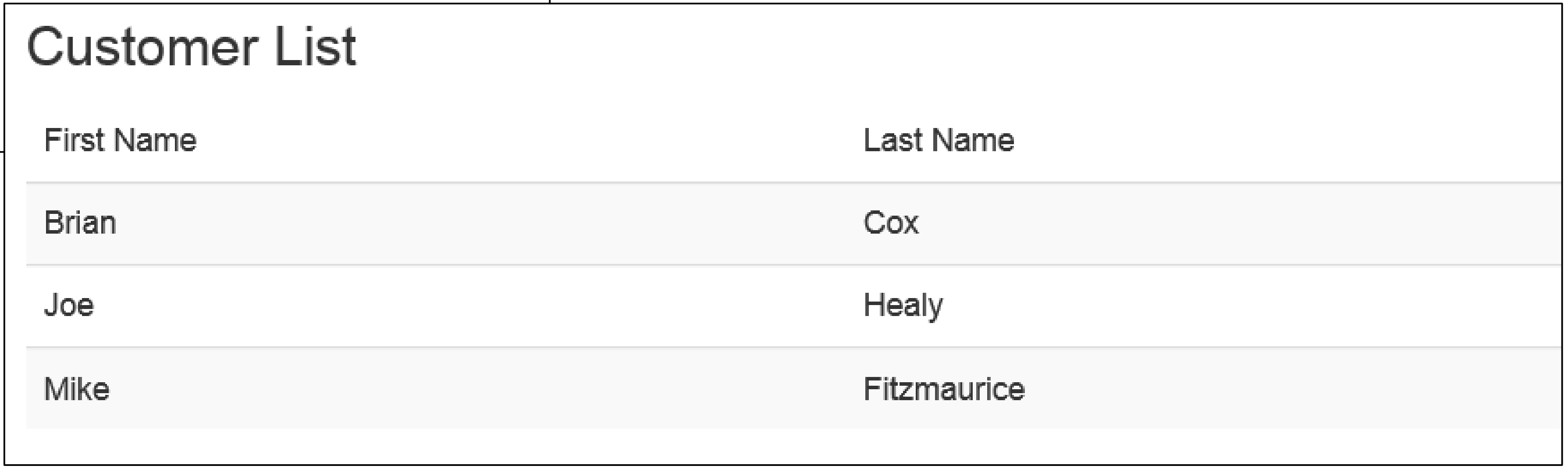

New Customer

Bootstrap Table Classes

```html
<div class="container">
    <div class="row">
        <h3>Customer List</h3>

        <table class="table table-striped table-responsive ">
            <thead>
                <tr><td>First Name</td><td>Last Name</td></tr>
            </thead>
            <tbody>
                <tr><td>Brian</td><td>Cox</td></tr>
                <tr><td>Joe</td><td>Healy</td></tr>
                <tr><td>Mike</td><td>Fitzmaurice</td></tr>
            </tbody>
        </table>

    </div>
</div>
```



Customer List

First Name	Last Name
Brian	Cox
Joe	Healy
Mike	Fitzmaurice

DEMO

Bootstrap

Summary

Enterprise JavaScript

JavaScript
TypeScript

Enterprise Services Architecture

REST and OData
Promises
Web API

Enterprise Frameworks

Angular
Bootstrap

Microsoft