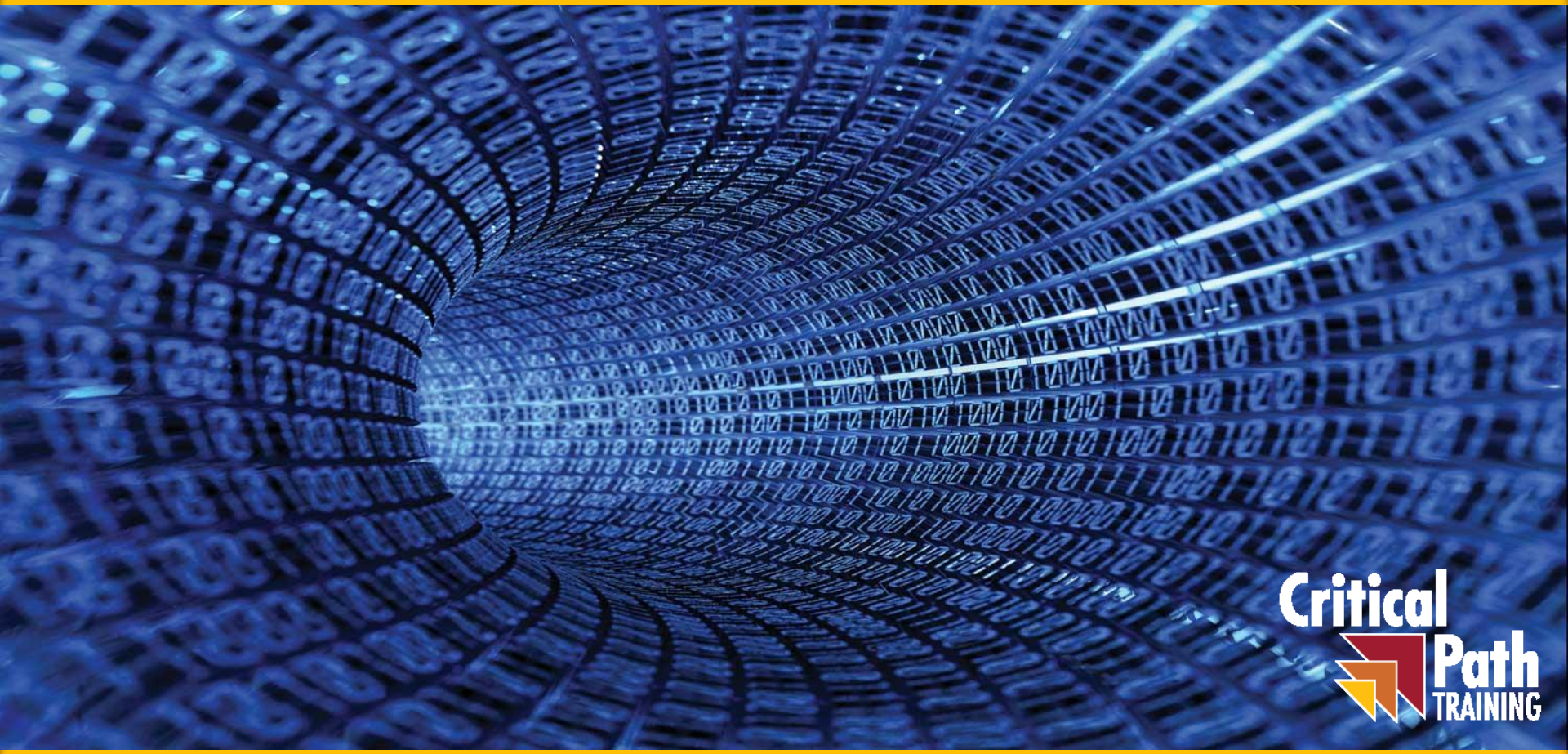
Developing SharePoint Add-ins with AngularJS



Agenda

- Introduction to AngularJS
- Directives and Modules
- Routes, Views and Controllers
- Angular Services

Introducing AngularJS

- What is AngularJS?
  - A JavaScript framework for building web applications
  - Based on Single-Page Application (SPA) model
  - Implements Model-View-Controller (MVC) Pattern
  - Check out the official site at http://angularjs.org/

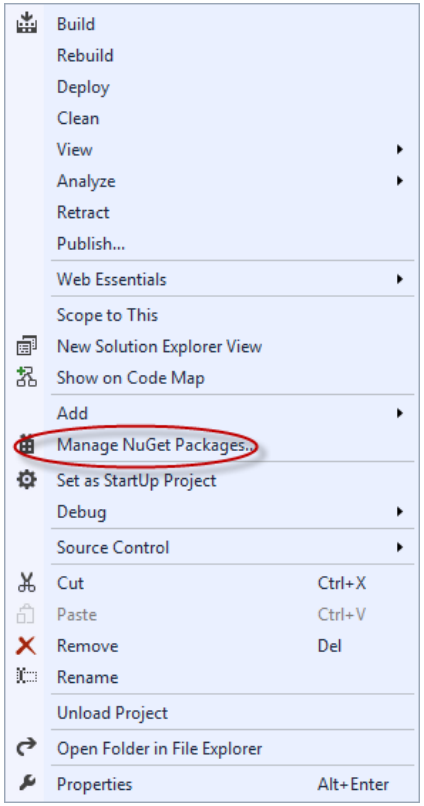
- Why is AngularJS so popular these days?
  - True framework instead of patchwork of libraries
  - Strong separation of concerns

Angular JS Features

- Directive
  - A shared unit of declarative functionality
- Module
  - A container for a reusable unit of code
- Controller
  - A JavaScript functions which processes incoming requests
- View
  - An HTML template that serves as a partial view on a page
- Model
  - JavaScript object containing domain-specific data prepared by controller
  - Object properties declaratively bound to HTML elements in the view
- Service
  - Built-in Angular services include $http, $window and $route
  - Custom services used to write code which is shared across controllers

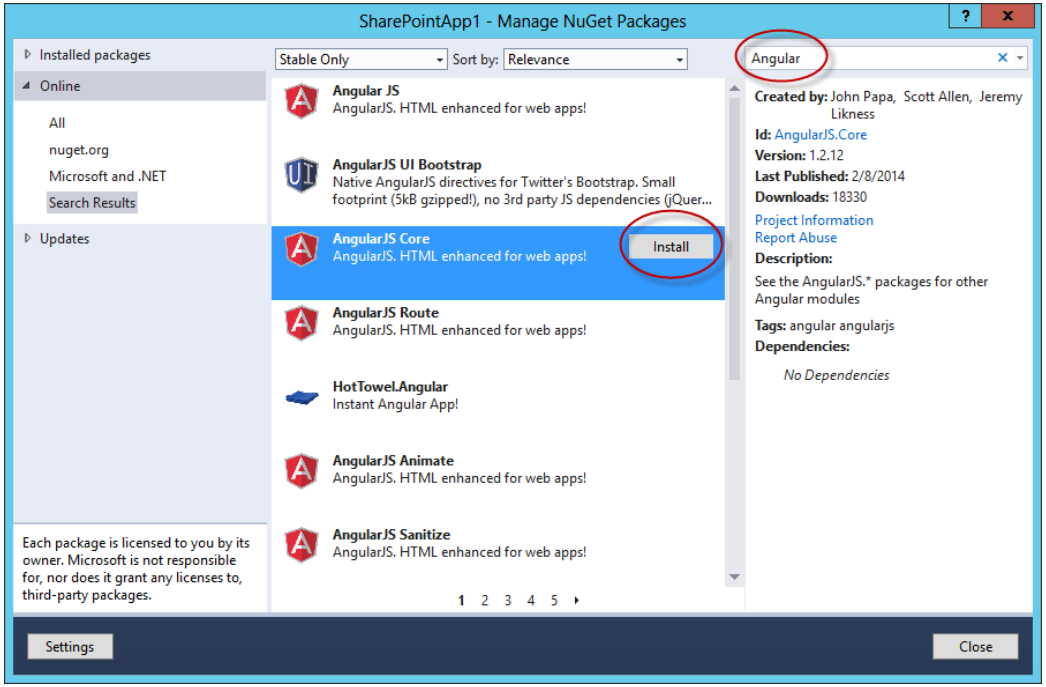
Adding Angular JS to an App

1. Manage NuGet Packages



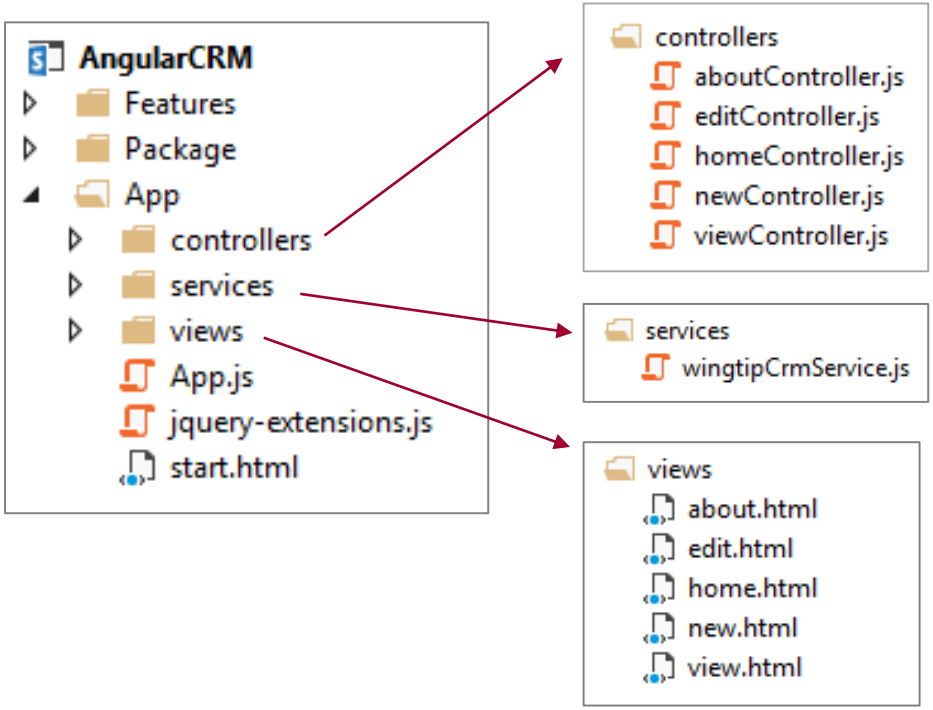
2. Search for “Angular”

3. Install Angular JS Core



SharePoint App Project Structure

- All application code maintained in **App** folder
  - App start page implemented using **start.html**
  - App initialization code maintained in **app.js**
  - Child folders added for controllers, services and views




DEMO
Starting an Angular JS App Project

Agenda

- ✓ Introduction to AngularJS
- ➢ Directives and Modules
- ▪ Routes, Views and Controllers
- ▪ Angular Services

Angular Directives

- AngularJS includes several built-in Directives
  - Directives created with custom attributes in HTML5
  - Custom attributes in HTML5 start with `data-`

- Angular Directives start with `data-ng` or `ng-`
  - You can use `data-ng-app` or `ng-app`
  - You can use `data-ng-controller` or `ng-controller`
  - You can use `data-ng-click` or `ng-click`

Key Angular Directives

- `data-ng-app`: initialize the Angular app
- `data-ng-controller`: designate controller scope
- `data-ng-view`: define placeholder for dynamic views
- `data-ng-bind`: one-way binding of HTML element to model
- `data-ng-model`: two-way binding of HTML element to model
- `data-ng-repeat`: `create` for-each loop
- `data-ng-click`: handle click event.
- `data-ng-cloak`: prevents view from displaying during start up
- `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

Understanding Modules

- Module represents a container of code
  - AngularJS provides several built-in Modules
  - Third parties libraries often created using Modules
- Named module can be created for app
  - App module named using **ng-app** Directive
  - App module initialized using **angular.module** function

```
<body ng-app="AngularCRM" >

  <div class="container">
    <div class="navbar navbar-default">...</div>
  </div>

  <div class="container">
    <div id="content-box" ng-view ></div>
  </div>

</body>
```

```
'use strict';

(function(){

  var crmApp = angular.module("AngularCRM", ['ngRoute'])

  crmApp.config(['$routeProvider', initializeApp]);

  function initializeApp($routeProvider) {
    // add code to initialize app
  }

})();
```


DEMO
Adding Angular Directives to a SharePoint-hosted App Start Page

Agenda

- ✓ Introduction to AngularJS
- ✓ Directives and Modules
- ➢ Routes, Views and Controllers
- ▪ Angular Services

Routes, View Template and Controllers

- What are Routes?
  - Route represents endpoint in the app's route map
  - Route configured with View template and Controller
- What is a View Template?
  - HTML fragment in .html file which acts as partial view
  - HTML in view template often created using Directives
- What is a controller?
  - JavaScript function which provides view logic
  - Controller creates and passes model to View Template

Defining Routes

- Steps to defining route map for an app
  - Add Angular JS Route NuGet Package
  - Reference the `ngRoute` Module
  - Define routes using the injected $routeProvider object

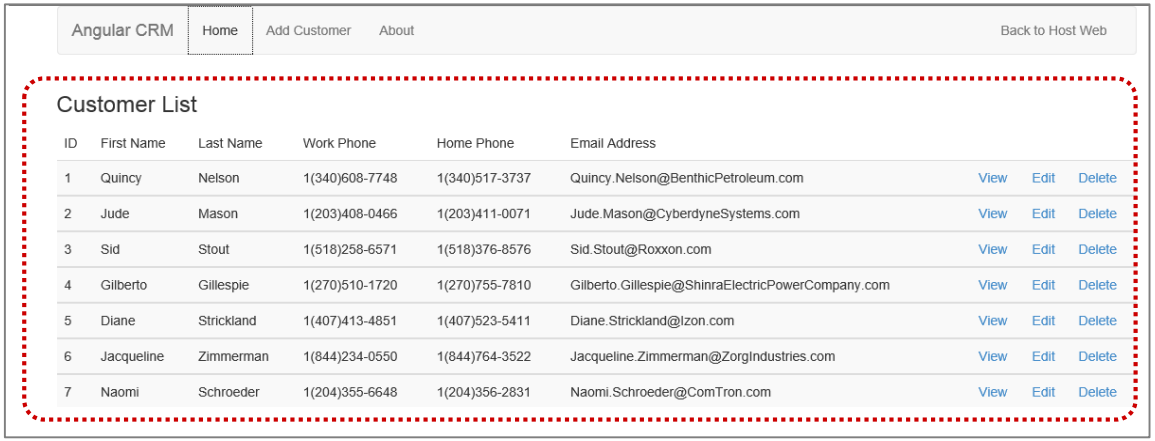
```
var crmApp = angular.module("AngularCRM", ['ngRoute']);

crmApp.config(['$routeProvider', initializeApp]);

function initializeApp($routeProvider) {

  // config app's route map
  $routeProvider
    .when("/",
         { templateUrl: 'views/home.html', controller: "homeController" })
    .when("/view/:id",
         { templateUrl: 'views/view.html', controller: "viewController" })
    .when("/edit/:id",
         { templateUrl: 'views/edit.html', controller: "editController" })
    .when("/new",
         { templateUrl: 'views/new.html', controller: "newController" })
    .when("/about",
         { templateUrl: 'views/about.html', controller: "aboutController" })
    .otherwise({ redirectTo: "/" });

}
```

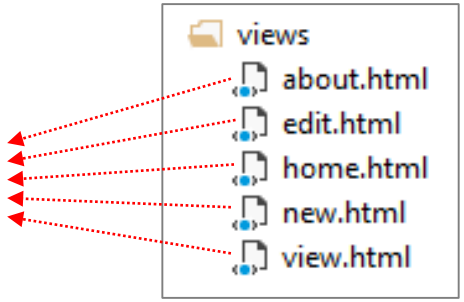
Dynamically Loading View Templates

- View placeholder element defined using ng-view attribute

```
<body ng-app="AngularCRM" >

  <div class="container">
    <div class="navbar navbar-default">...</div>
  </div>

  <div class="container">
    <div id="content-box" ng-view></div>
  </div>

</body>
```

- View templates are loaded into view placeholder element





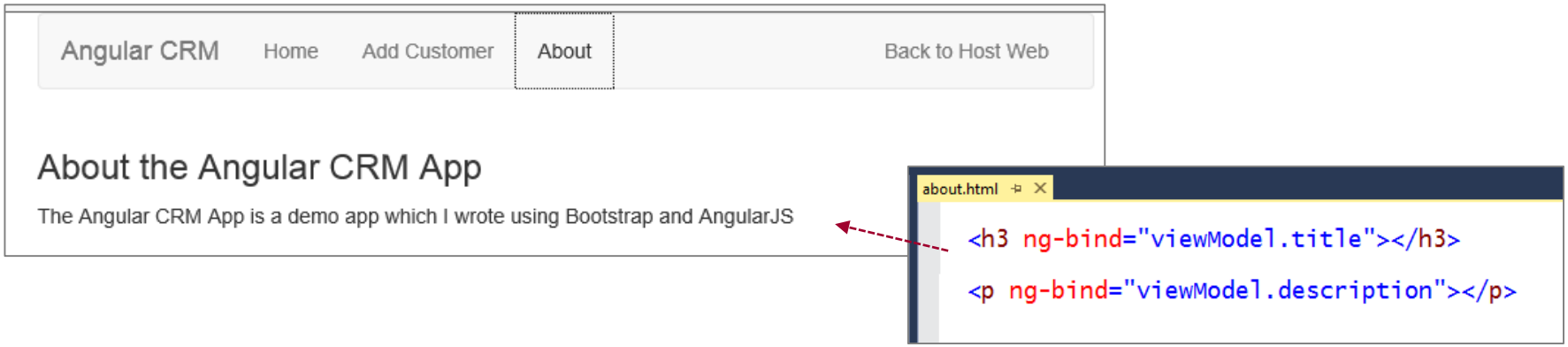
Understanding Controllers

- Controllers are implemented using JavaScript functions
  - Controller registered using **controller** function on app module
  - Controller typically creates object to serve as view model
  - Controllers makes view model accessible through $scope object

```
// acquire reference to app module
var app = angular.module('AngularCRM');

// register controller with app module
app.controller('aboutController', processRequest);

// implement controller function
function processRequest($scope) {

  // (1) create object to serve as view model
  var viewModel = {
    title: "About the Angular CRM App",
    description: "The Angular CRM App is a demo app which I wrote using Bootstrap and AngularJS"
  };

  // (2) add reference to $Scope to make make view model accessible to view template
  $scope.viewModel = viewModel;
}
```

Understanding View Templates

- View Templates are implemented using HTML
  - You add HTML and CSS as usual
  - Directives are added to enable data binding to model
  - Directives are added to wire up event handlers in controller
- Data binding directives
  - **ng-bind**: used to create one-way, read-only binding
  - **ng-model**: used to create two-way, read-write binding



Programming $scope in Controllers & Views

- Rule of thumb in dealing with $scope
  - Treat $scope as write-only in controllers
  - Treat $scope as read-only in templates
- Common misconception that $scope ***is the*** model
  - In truth, $scope is not the model
  - Instead, $scope references a separate object which is the model
- Don’t bind elements directly to $scope properties
  - Unexpected behavior occurs in child scopes
  - Instead, create separate JavaScript object for model
  - If **ng-bind** or **ng-model** value doesn't have a dot (.) - you’re doing it wrong

```
<!-- correct -->
<h3 ng-bind="viewModel.title"></h3>
```

```
<!-- wrong -->
<h3 ng-bind="title"></h3>
```

Filters

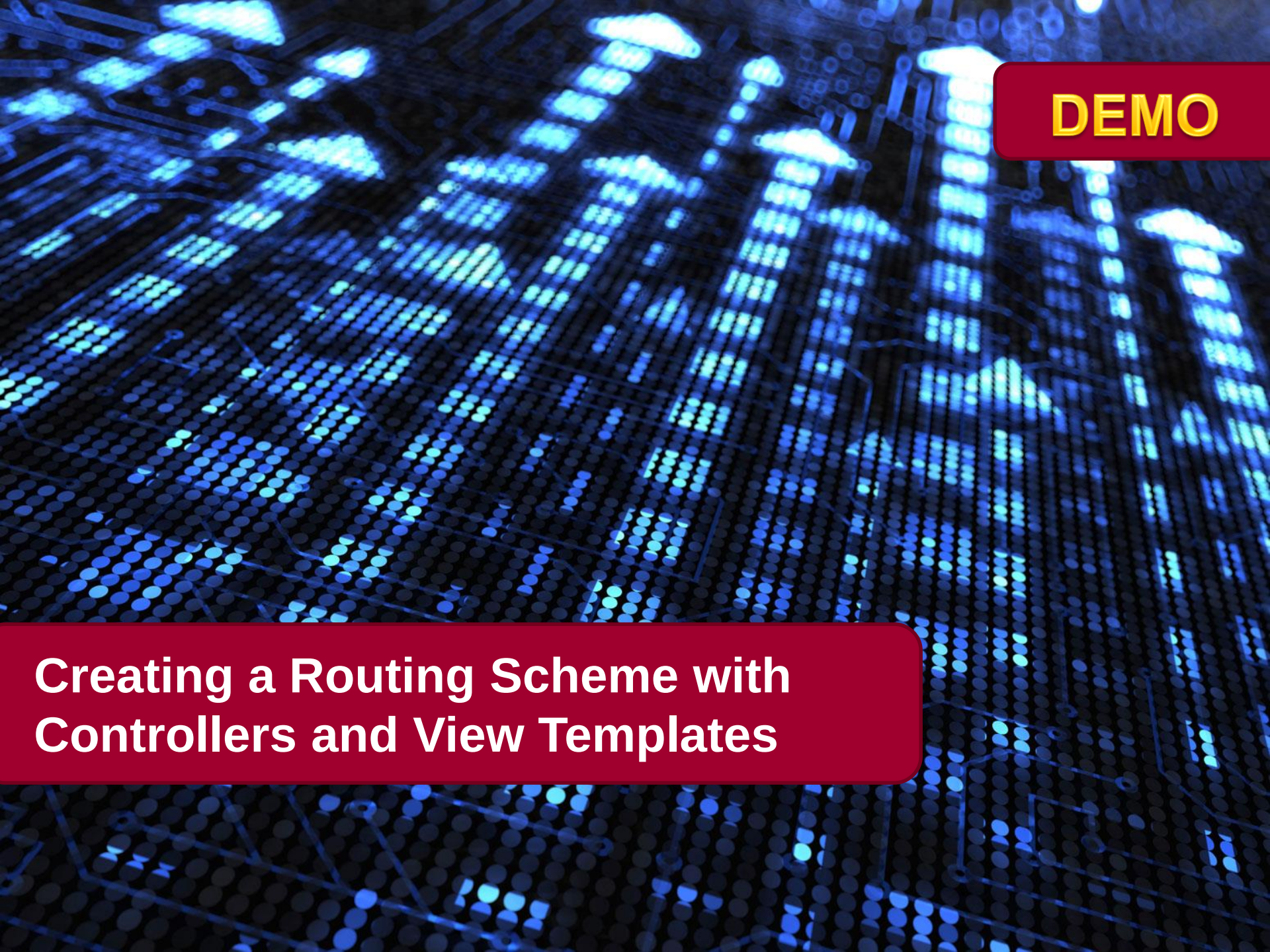
- Perform common operations on data bound elements
  - Takes the form of {{ expression | filter }}

```
<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


DEMO
Creating a Routing Scheme with Controllers and View Templates

Agenda

- ✓ Introduction to AngularJS
- ✓ Directives and Modules
- ✓ Routes, Views and Controllers
- ➢ Angular Services

Service Components included with AngularJS

- Angular includes many built-in service components
  - This tables lists some of the more commonly used services

Service	Purpose
$http	used to communicate with the remote HTTP servers using XMLHttpRequest object
$location	used to retrieve the URL in the browser address bar
$log	safely writes the message into the browser's console
$q	promise/deferred implementation
$window	reference to the browser's window object
$anchorScroll	scrolls to the related element
$filter	used for formatting data displayed to the user
$route	used for deep-linking URLs to controllers and views
$routeParams	allows you to retrieve the current set of route parameters

Custom Services in Angular

- What type of code should be written in a service?
  - Any code which is to be shared across controllers
  - Any code which calls to servers across the network
- How do you create a service?
  - Call **factory** method on App Module object to create a new service

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

- How do you use the service from a controller?
  - Pass it by name to any controller function to trigger code injection

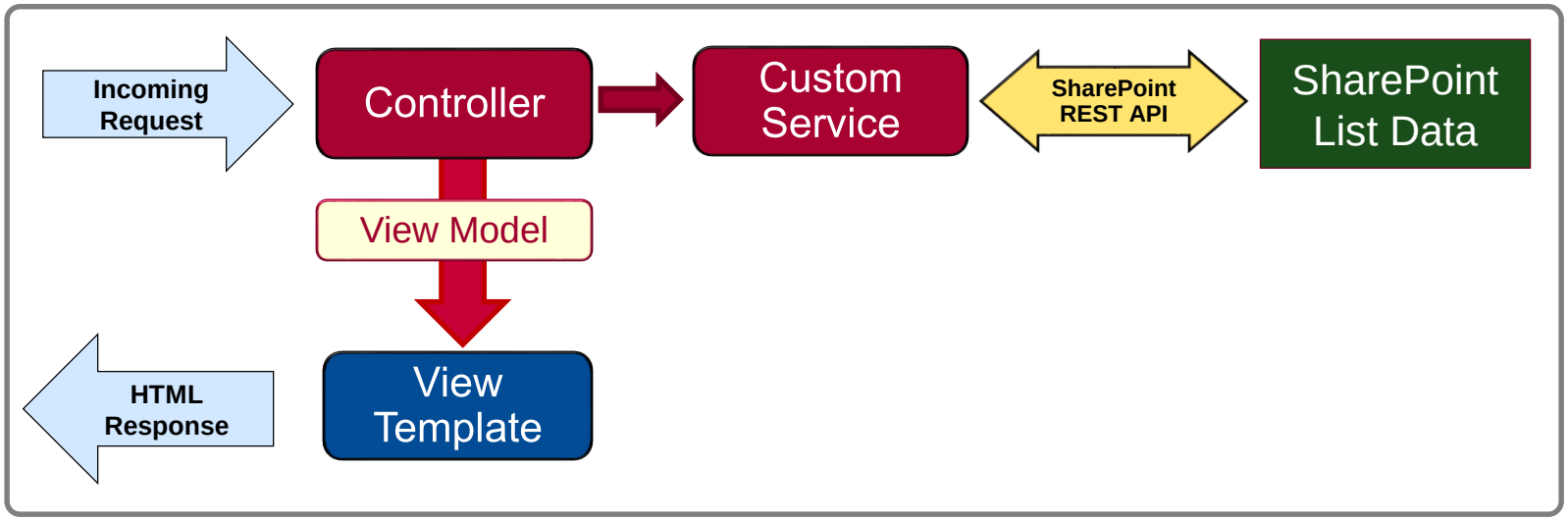
```
Wingtip.App.controller("myCtrl", ["$scope", "welcomeService",
function contactsCtrl($scope, welcomeService) {
      welcomeService.greet();
   }]
);
```

Best Practices with Services and Controllers

- Controllers should never reference the DOM
  - DOM manipulation done using custom Directives
- Controllers should define view behavior
  - What happens when user clicks Filter button?
  - What happens when user clicks Save button?
- Controllers should not contain any data access code
  - Code to call across network should be written in service(s)
- Services should rarely reference the DOM
  - Exception is service which interacts with user using modal dialog
  - Service logic should be completely decoupled from all views

Controller Processing Flow

1. Incoming request routed to Controller using app's route map
2. Controller calls data access function provided by custom service
3. Custom service calls across network to fetch SharePoint list data
4. Custom service returns SharePoint list data to Controller
5. Controller uses SharePoint list data to create model
6. Controller passes model to View Template using $scope
7. View Template binds to model data using Directives
8. View Templates renders HTML which is returned to client



wingtipCrmService

```
(function () {

  var app = angular.module('AngularCRM');

  app.factory("wingtipCrmService", createServiceObject);

  function createServiceObject($http) {
    // create service object
    var service = {};

    // set default headers for $http service
    $http.defaults.headers.common.Accept = 'application/json;odata=verbose;';

    // initialize app with SharePoint form digest value
    var requestDigest;
    $http({
      method: 'POST',
      url: "../_api/contextinfo",
      headers: { "Accept": "application/json; odata=verbose" }
    }).success(function (data) {
      requestDigest = data.d.GetContextWebInformation.FormDigestValue
    });

    service.getCustomers = function ()...

    service.getCustomer = function (id)...

    service.deleteCustomer = function (id)...

    service.addCustomer = function (FirstName, LastName, Company, WorkPhone, HomePhone, Email)...

    service.updateCustomer = function (id, FirstName, LastName, WorkPhone, HomePhone, Email, etag)...

    // return service object to angular framework
    return service;
  }

})();
```

Accessing the SharePoint REST API

```
service.getCustomers = function () {
  var restQueryUrl = "../_api/web/lists/getByTitle('Customers')/items/" +
                     "?$select=ID,Title,FirstName,WorkPhone,HomePhone,Email";
  return $http({
    method: 'GET',
    url: restQueryUrl,
    headers: { "Accept": "application/json; odata=verbose" }
  })
}

service.getCustomer = function (id) {
  var restQueryUrl = "../_api/web/lists/getByTitle('Customers')/items(" + id + ")/" +
                     "?$select=ID,Title,FirstName,WorkPhone,HomePhone,Email";
  return $http({
    method: 'GET',
    url: restQueryUrl,
    headers: { "Accept": "application/json; odata=verbose" }
  })
}

service.deleteCustomer = function (id) {
  var restQueryUrl = "../_api/web/lists/getByTitle('Customers')/items(" + id + ")";
  return $http({
    method: 'DELETE',
    url: restQueryUrl,
    headers: {
      "Accept": "application/json; odata=verbose",
      "X-RequestDigest": requestDigest,
      "If-Match": "*"
    }
  });
}
```


DEMO
Adding a Custom Service to Access a SharePoint List

Summary

- ✓ Introduction to AngularJS
- ✓ Directives and Modules
- ✓ Routes, Views and Controllers
- ✓ Angular Services