What is AngularJS

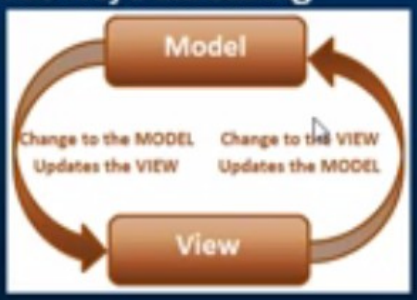
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

What is AngularJS

AngularJS is a JavaScript framework that helps build web applications

Benefits of AngularJS

- Dependency Injection
- Two Way Data-Binding



- Testing
- Model View Controller
- Directives, Filters etc...

AngularJS Module and Controller

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

What is a module in AngularJS

- A module is a container for different parts of your application i.e controllers, services, directives, filters, etc.
- You can think of a module as a Main() method in other types of applications.

How to create a module

Use the angular object's module() method to create a module.

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

What is a controller in angular

In angular a controller is a JavaScript function. The job of the controller is to build a model for the view to display

How to create a controller in angular

```
var myController = function ($scope) {
    $scope.message = "AngularJS Tutorial";
}
```

AngularJS Module and Controller

How to register the controller with the module

```
//Create the module
var myApp = angular.module("myModule", []);

//Create the controller
var myController = function ($scope) {
    $scope.message = "AngularJS Tutorial";
}

// Register the controller with the module
myApp.controller("myController", myController);
```

OR

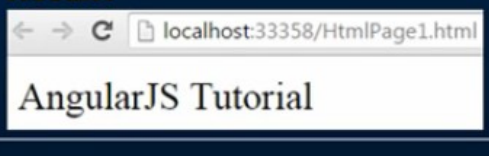
```
//Create the module
var myApp = angular.module("myModule", []);

// Creating the controller and registering with the module all done in one line
myApp.controller("myController", function ($scope) {
    $scope.message = "AngularJS Tutorial";
});
```

AngularJS Module and Controller

```
// Script.js
var myApp = angular.module("myModule", []);

myApp.controller("myController", function ($scope) {
    $scope.message = "AngularJS Tutorial";
});
```

```
<!doctype html>
<html ng-app="myModule">
<head>
    <script src="Scripts/angular.min.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body>
    <div ng-controller="myController">
        {{ message }}
    </div>
</body>
</html>
```

Result :



Controllers in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

Attaching a complex object to the scope

```
myApp.controller("myController", function ($scope) {
    var employee = {
        firstName: 'Mark',
        lastName: 'Hastings',
        gender: 'Male'
    };
    $scope.employee = employee;
});
```

With in the view, we can then retrieve the employee properties and display

```
<div ng-controller="myController">
    <div>First Name : {{ employee.firstName }}</div>
    <div>Last Name : {{ employee.lastName }}</div>
    <div>Gender : {{ employee.gender}}</div>
</div>
```

Controllers in AngularJS

What happens if the controller name is misspelled

When the controller name is misspelled, 2 things happen

- An error is raised. To see the error, use browser developer tools
- The binding expressions in the view that are in the scope of the controller will not be evaluated

What happens if a property name in the binding expression is misspelled

Expression evaluation in angular is forgiving, meaning if you misspell a property name in the binding expression, angular will not report any error. It will simply return null or undefined.

Controllers in AngularJS

How to create module, controller and register the controller with the module, all in one line

Use the method chaining mechanism as shown below

```
var myApp = angular
    .module("myModule", [])
    .controller("myController", function ($scope) {
        var employee = {
            firstName: 'Mark',
            lastName: 'Hastings',
            gender: 'Male'
        };
        $scope.employee = employee;
    });
```

AngularJS ng-src directive

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

Using a binding expression with the image src attribute, results in a 404 error

```
<img src="{{country.flag}}"/>
```

→ Failed to load resource: the server responded with a status of 404 (Not Found)

Country : United States of America
Capital : Washington, D.C.

Reason for the 404 error

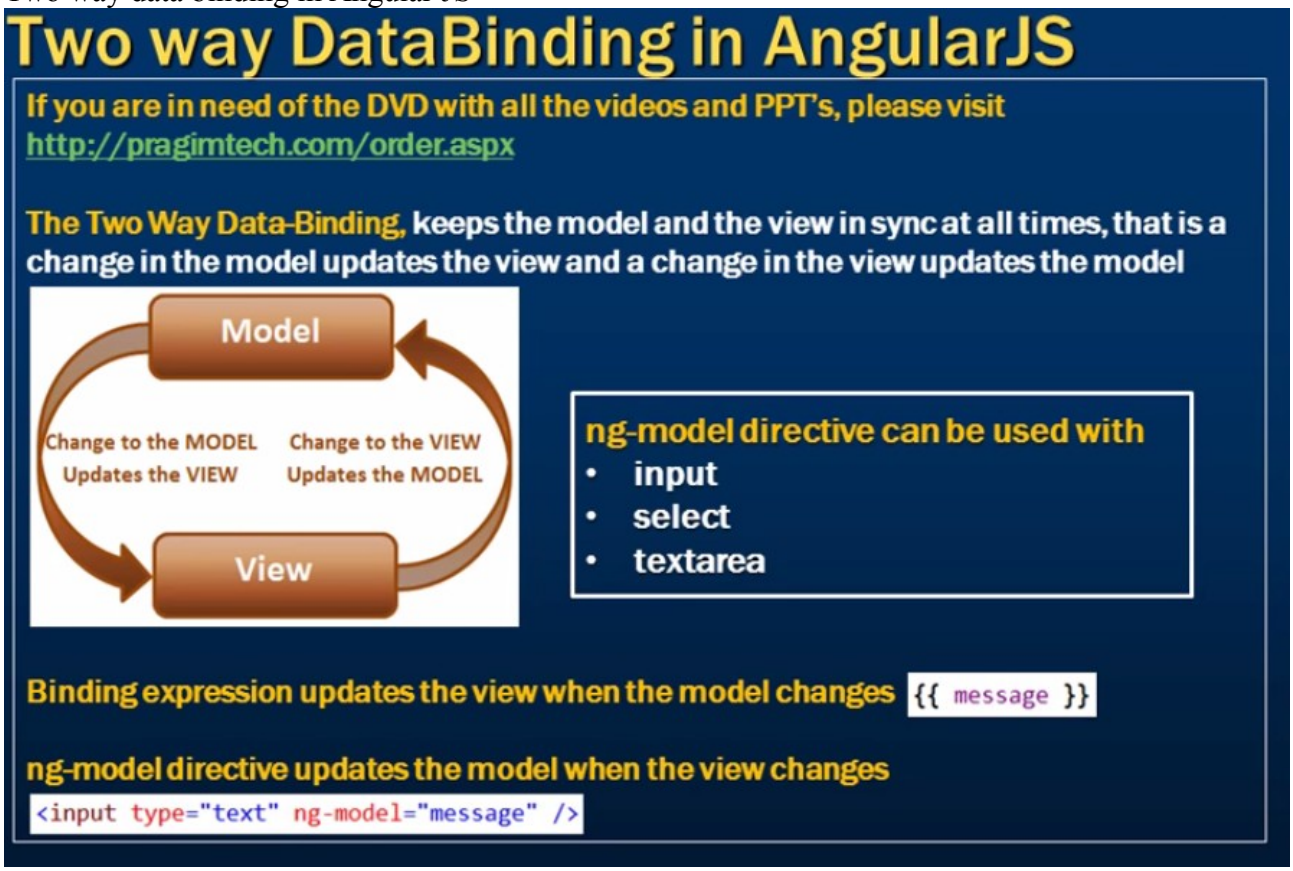
As soon as the DOM is parsed, an attempt is made is to fetch the image from the server. At this point, AngularJS binding expression that is specified with the src attribute is not evaluated. Hence 404 (not found) error.

To fix the 404 error use the ng-src directive : ng-src attribute ensures that a request is issued only after AngularJS has evaluated the binding expression

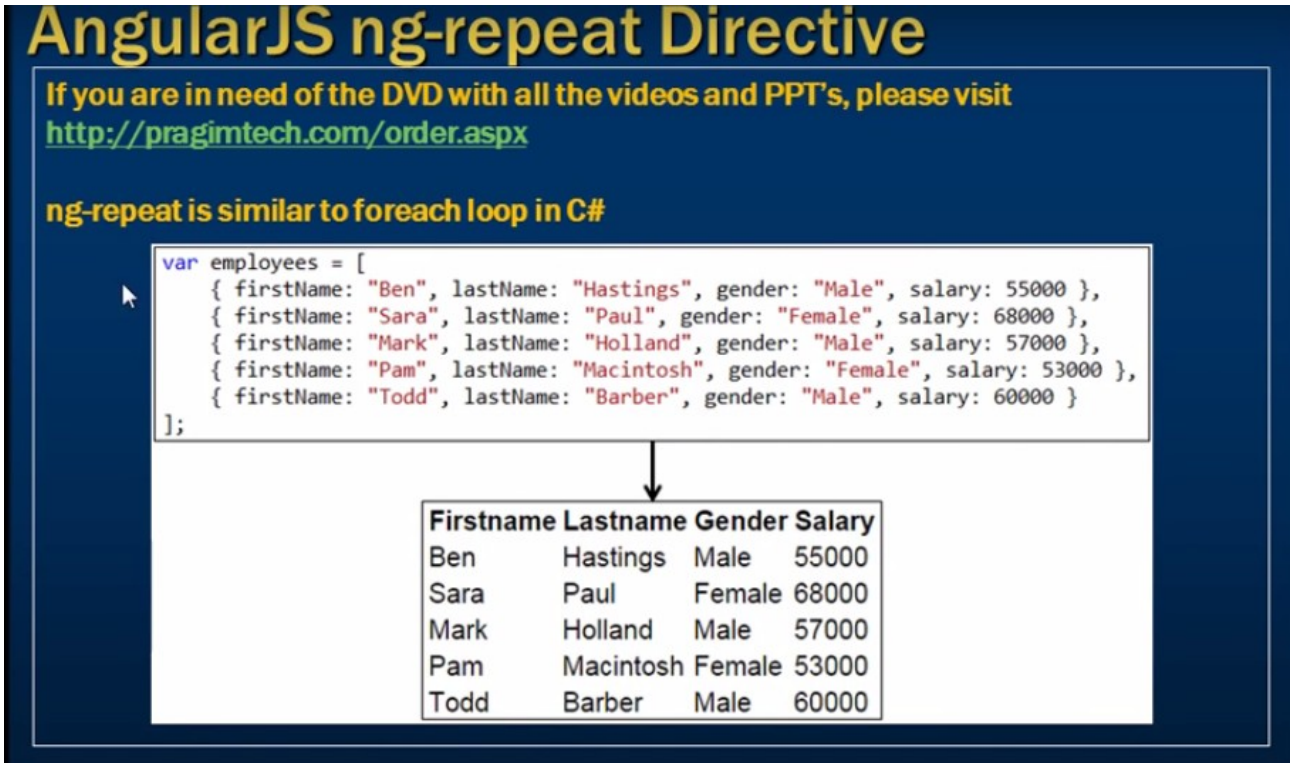
```html
<!doctype html>
<html ng-app="myModule">
<head>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
</head>
<body>
    <div ng-controller="myController">
        <div>
            Name : {{ country.name }}
        </div>
        <div>
            Capital : {{ country.capital }}
        </div>
        <div>
            <img ng-src="{{ country.flag }}"
                 alt="{{ country.name + ' Flag' }}"
                 style="height:100px; width:200px" />
        </div>
    </div>
</body>
</html>
```

```javascript
var myApp = angular
                .module("myModule", [])
                .controller("myController", function ($scope) {
                    var country = {
                        name: "USA",
                        capital: "Washington, D.C.",
                        flag: "/Images/usa-flag.png"
                    };

                    $scope.country = country;
                });
```

Two way data binding in Angular JS



ng-repeate



AngularJS ng-repeat Directive

Finding the index of an item in the collection :

- To find the index of an item in the collection use $index property
- To get the index of the parent element
  - Use $parent.$index or
  - Use ng-init="parentIndex = $index"

- UK - Index = 0
  - London - Parent Index = 0, Index = 0
  - Birmingham - Parent Index = 0, Index = 1
  - Manchester - Parent Index = 0, Index = 2
- USA - Index = 1
  - Los Angeles - Parent Index = 1, Index = 0
  - Chicago - Parent Index = 1, Index = 1
  - Houston - Parent Index = 1, Index = 2
- India - Index = 2
  - Hyderabad - Parent Index = 2, Index = 0
  - Chennai - Parent Index = 2, Index = 1
  - Mumbai - Parent Index = 2, Index = 2

```
    <script src="Scripts/Script.js"></script>
</head>
<body ng-app="myModule" style="font-family:Arial">
    <div ng-controller="myController">
        <ul>
            <li ng-repeat="country in countries">
                {{ country.name }} - Index = {{ $index }}
                <ul>
                    <li ng-repeat="city in country.cities">
                        {{ city.name }} - Parent Index = {{ $parent.$index }}, Index = {{ $index }}
                    </li>
                </ul>
            </li>
        </ul>
    </div>
</body>
</html>
```

```
    <script src="Scripts/Script.js"></script>
</head>
<body ng-app="myModule" style="font-family:Arial">
    <div ng-controller="myController">
        <ul>
            <li ng-repeat="country in countries" ng-init="parentIndex=$index">
                {{ country.name }} - Index = {{ $index }}
                <ul>
                    <li ng-repeat="city in country.cities">
                        {{ city.name }} - Parent Index = {{ parentIndex }}, Index = {{ $index }}
                    </li>
                </ul>
            </li>
        </ul>
    </div>
</body>
</html>
```

Handling events in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

1. Display the list of technologies in a table
2. Provide the ability to like and dislike a technology
3. Increment the likes and dislikes when the respective buttons are clicked

Name	Likes	Dislikes	Like/Dislike
C#	16	1	Like Dislike
ASP.NET	3	1	Like Dislike
SQL	16	2	Like Dislike
AngularJS	25	0	Like Dislike

Handling events in AngularJS

Script.js : In the controller function we have 2 methods to increment likes and dislikes. Both the functions have the technology object that we want to like or dislike as a parameter

```
var app = angular
            .module("myModule", [])
            .controller("myController", function ($scope) {

                var technologies = [
                    { name:"C#", likes:0, dislikes:0 },
                    { name: "ASP.NET", likes: 0, dislikes: 0 },
                    { name: "SQL", likes: 0, dislikes: 0 },
                    { name: "AngularJS", likes: 0, dislikes: 0 }
                ];

                $scope.technologies = technologies;

                $scope.incrementLikes = function (technology) {
                    technology.likes++;
                };

                $scope.incrementDislikes = function (technology) {
                    technology.dislikes++;
                };
            });
```

Handling events in AngularJS

```
<div ng-controller="myController">
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Likes</th>
                <th>Dislikes</th>
                <th>Like/Dislike</th>
            </tr>
        </thead>
        <tbody>
            <tr ng-repeat="technology in technologies">
                <td> {{ technology.name }} </td>
                <td style="text-align:center"> {{ technology.likes }} </td>
                <td style="text-align:center"> {{ technology.dislikes }} </td>
                <td>
                    <input type="button" ng-click="incrementLikes(technology)"
                           value="Like" />
                    <input type="button" ng-click="incrementDislikes(technology)"
                           value="Dislike" />
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

IncrementLikes() and incrementDislikes() functions are associated with the respective buttons. When any of these buttons are clicked, the corresponding technology object is automatically passed to the function, and the likes or dislikes property is incremented depending on which button is clicked

Handling events in AngularJS

Styles.css : Styles for table, td and th elements

```
table {
    border-collapse: collapse;
    font-family:Arial;
}

td {
    border: 1px solid black;
    padding: 5px;
}

th {
    border: 1px solid black;
    padding: 5px;
    text-align: left;
}
```

AngularJS Filters

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

Filters can do 3 different things
Format, Sort and Filter data

Filters can be used with a binding expression or a directive

To apply a filter use pipe (|) character

```
{{ expression | filterName:parameter }}
```

Filters for formatting data

Filter	Description
lowercase	Formats all characters to lowercase
uppercase	Formats all characters to uppercase
number	Formats a number as text. Includes comma as thousands separator and the number of decimal places can be specified
currency	Formats a number as a currency. $ is default. Custom currency and decimal places can be specified
date	Formats date to a string based on the requested format

AngularJS Filters

Angular Date formats

Format	Result
yyyy	4 digit year. Exampe 1998
yy	2 digit year. Example 1998 => 98
MMMM	January-December
MMM	Jan-Dec
MM	01-12
M	1-12 (No leading ZEROS)
dd	01 - 31
d	1 - 31 (No leading ZEROS)

Angular date format documentation
https://docs.angularjs.org/api/ng/filter/date

limitTo filter : Can be used to limit the number of rows or characters in a string

```
{{ expression | limitTo : limit : begin}}
```

AngularJS Filters

Demo

Rows to display : 3

Name	Date of Birth	Gender	Salary (number filter)	Salary (currency filter)
BEN	23/11/1980	Male	55,000.79	£55,000.8
SARA	05/05/1970	Female	68,000.00	£68,000.0
MARK	15/08/1974	Male	57,000.00	£57,000.0

```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", dateOfBirth: new Date("November 23, 1980"), gender: "Male", sa
                { name: "Sara", dateOfBirth: new Date("May 05, 1970"), gender: "Female", sala
                { name: "Mark", dateOfBirth: new Date("August 15, 1974"), gender: "Male", sal
                { name: "Pam", dateOfBirth: new Date("October 27, 1979"), gender: "Female", s
                { name: "Todd", dateOfBirth: new Date("December 30, 1983"), gender: "Male", s
            ];

            $scope.employees = employees;
            $scope.rowLimit = 3;
        });
```

```
<div ng-controller="myController">
    Rows to display : <input type="number" step="1" min="0" max="5" ng-model="rowLimit"
    <br /><br />
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Date of Birth</th>
                <th>Gender</th>
                <th>Salary (number)</th>
                <th>Salary (currency)</th>
            </tr>
        </thead>
        <tbody>
            <tr ng-repeat="employee in employees | limitTo:rowLimit">
                <td>{{ employee.name | uppercase }}</td>
                <td>{{ employee.dateOfBirth | date:"dd/MM/yyyy" }}</td>
                <td>{{ employee.gender | lowercase}}</td>
                <td>{{ employee.salary | number:2 }}</td>
                <td>{{ employee.salary | currency:"£":1 }}</td>
            </tr>
        </tbody>
    </table>
```

Sorting Data in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

To sort the data in Angular

Use orderBy filter

```
{{ orderBy_expression | orderBy : expression : reverse}}
```

Example : `ng-repeat="employee in employees | orderBy:'salary':false"`

To sort in ascending order, set reverse to false

To sort in descending order, set reverse to true

You can also use + and - to sort in ascending and descending order respectively

```
ng-repeat="employee in employees | orderBy:'+salary'"
```

Sort By : Name ASC

Name	Date of Birth	Gender	Salary
Ben	23/11/1980	Male	55000
Mark	15/08/1974	Male	57000
Pam	27/10/1979	Female	53000
Sara	05/05/1970	Female	68000
Todd	30/12/1983	Male	60000

Script.js - Controller Function

```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {
            var employees = [
                {   name: "Ben", dateOfBirth: new Date("November 23, 1980"),
                    gender: "Male", salary: 55000
                },
                {   name: "Sara", dateOfBirth: new Date("May 05, 1970"),
                    gender: "Female", salary: 68000
                },
                {   name: "Mark", dateOfBirth: new Date("August 15, 1974"),
                    gender: "Male", salary: 57000
                },
                {   name: "Pam", dateOfBirth: new Date("October 27, 1979"),
                    gender: "Female", salary: 53000
                },
                {   name: "Todd", dateOfBirth: new Date("December 30, 1983"),
                    gender: "Male", salary: 60000
                }
            ];
            $scope.employees = employees;
            $scope.sortColumn = "name";
        });
```

HtmlPage1.html

```
<div ng-controller="myController">
    Sort By :
    <select ng-model="sortColumn">
        <option value="name">Name ASC</option>
        <option value="+dateOfBirth">Date of Birth ASC</option>
        <option value="+gender">Gender ASC</option>
        <option value="-salary">Salary DESC</option>
    </select>
    <br /><br />
    <table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Date of Birth</th>
                <th>Gender</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tbody>
            <tr ng-repeat="employee in employees | orderBy:sortColumn">
                <td>{{ employee.name }}</td>
                <td>{{ employee.dateOfBirth | date:"dd/MM/yyyy" }}</td>
                <td>{{ employee.gender }}</td>
                <td>{{ employee.salary  }}</td>
            </tr>
        </tbody>
    </table>
</div>
```

Bidirectional Sort in AngularJS

AngularJS Sort Rows by Table Header

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

Name	Date of Birth ▼	Gender	Salary
Todd	30/12/1983	Male	60000
Ben	23/11/1980	Male	55000
Pam	27/10/1979	Female	53000
Mark	15/08/1974	Male	57000
Sara	05/05/1970	Female	68000

Script.js - Controller

```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {
            var employees = [
            { name: "Ben", dateOfBirth: new Date("November 23, 1980"), gender: "Male", salary: 55000 },
            { name: "Sara", dateOfBirth: new Date("May 05, 1970"), gender: "Female", salary: 68000 },
            { name: "Mark", dateOfBirth: new Date("August 15, 1974"), gender: "Male", salary: 57000 },
            { name: "Pam", dateOfBirth: new Date("October 27, 1979"), gender: "Female", salary: 53000 },
            { name: "Todd", dateOfBirth: new Date("December 30, 1983"), gender: "Male", salary: 60000 }
            ];

            $scope.employees = employees; $scope.sortColumn = "name"; $scope.reverseSort = false;

            $scope.sortData = function (column) {
                $scope.reverseSort = ($scope.sortColumn == column) ? !$scope.reverseSort : false;
                $scope.sortColumn = column;
            }

            $scope.getSortClass = function (column) {
                if ($scope.sortColumn == column) {
                    return $scope.reverseSort ? 'arrow-down' : 'arrow-up';
                }

                return '';
            }
        });
```

HtmlPage1.html - View

```
<table>
    <thead>
        <tr>
            <th ng-click="sortData('name')">
                Name <div ng-class="getSortClass('name')"></div>
            </th>
            <th ng-click="sortData('dateOfBirth')">
                Date of Birth <div ng-class="getSortClass('dateOfBirth')"></div>
            </th>
            <th ng-click="sortData('gender')">
                Gender <div ng-class="getSortClass('gender')"></div>
            </th>
            <th ng-click="sortData('salary')">
                Salary <div ng-class="getSortClass('salary')"></div>
            </th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="employee in employees | orderBy:sortColumn:reverseSort">
            <td> {{ employee.name }} </td>
            <td> {{ employee.dateOfBirth | date:"dd/MM/yyyy" }} </td>
            <td> {{ employee.gender }} </td>
            <td> {{ employee.salary  }} </td>
        </tr>
    </tbody>
</table>
```

Styles.css

```css
body {
    font-family: Arial;
}

table {
    border-collapse: collapse;
}

td {
    border: 1px solid black;
    padding: 5px;
}

th {
    border: 1px solid black;
    padding: 5px;
    text-align: left;
    /*cursor property displays hand symbol
        when hovered over the th element*/
    cursor: pointer;
}
```

```css
/*This class displays the UP arrow*/
.arrow-up {
    width: 0;
    height: 0;
    border-left: 5px solid transparent;
    border-right: 5px solid transparent;
    border-bottom: 10px solid black;
    display:inline-block;
}

/*This class displays the DOWN arrow*/
.arrow-down {
    width: 0;
    height: 0;
    border-left: 5px solid transparent;
    border-right: 5px solid transparent;
    border-top: 10px solid black;
    display:inline-block;
}
```

Search Filter

Search filter in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

As we type in the search textbox, all the columns in the table must be searched and only the matching rows should be displayed

Search : Search employees

Name	Gender	Salary	City
Ben	Male	55000	London
Sara	Female	68000	Chennai
Mark	Male	57000	London
Pam	Female	53000	Chennai
Todd	Male	60000	London

Script.js : Controller

```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: "Male", salary: 55000, city: "London" },
                { name: "Sara", gender: "Female", salary: 68000, city: "Chennai" },
                { name: "Mark", gender: "Male", salary: 57000, city: "London" },
                { name: "Pam", gender: "Female", salary: 53000, city: "Chennai" },
                { name: "Todd", gender: "Male", salary: 60000, city: "London" },
            ];

            $scope.employees = employees;
        });
```

HtmlPage1.html - View

```
Search : <input type="text" ng-model="searchText" placeholder="Search employees" />
<br /><br />
<table>
    <thead>
        <tr>
            <th>Name</th>
            <th>Gender</th>
            <th>Salary</th>
            <th>City</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="employee in employees | filter:searchText">
            <td> {{ employee.name }} </td>
            <td> {{ employee.gender }} </td>
            <td> {{ employee.salary  }} </td>
            <td> {{ employee.city  }} </td>
        </tr>
    </tbody>
</table>
```

Angularjs filter by multiple properties

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

Multiple search textboxes. Each textbox searching a specific property


```
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="text" ng-model="searchText.name" placeholder="Search name" />
        <input type="text" ng-model="searchText.city" placeholder="Search city" />
        <input type="checkbox" ng-model="exactMatch"/> Exact Match
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>Salary</th>
                    <th>City</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | filter:searchText:exactMatch">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender }} </td>
                    <td> {{ employee.salary  }} </td>
                    <td> {{ employee.city  }} </td>
```

Angularjs filter by multiple properties


```
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <input type="text" ng-model="searchText" placeholder="Search city & name" />
        <br /><br />
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>Salary</th>
                    <th>City</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees | filter:search">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender }} </td>
                    <td> {{ employee.salary  }} </td>
                    <td> {{ employee.city  }} </td>
                </tr>
            </tbody>
```

```
        { name: "Mark", gender: "Male", salary: 57000, city: "London" },
        { name: "Pam", gender: "Female", salary: 53000, city: "Chennai" },
        { name: "Todd", gender: "Male", salary: 60000, city: "London" },
    ];

    $scope.employees = employees;

    $scope.search = function (item) {
        if ($scope.searchText == undefined) {
            return true;
        }
        else {
            if(item.name.toLowerCase().indexOf($scope.searchText.toLowerCase()) != -1 ||
                item.city.toLowerCase().indexOf($scope.searchText.toLowerCase()) != -1)
            {
                return true;
            }
        }

        return false;
    }
});
```

Create a Custom Filter in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

Custom filter in AngularJS

- Is a function that returns a function
- Use the filter function to create a custom filter

Example : Create a custom filter to convert integer values 1, 2, 3 to Male, Female and Not disclosed respectively

Name	Gender	Salary
Ben	1	55000
Sara	2	68000
Mark	1	57000
Pam	2	53000
Todd	3	60000

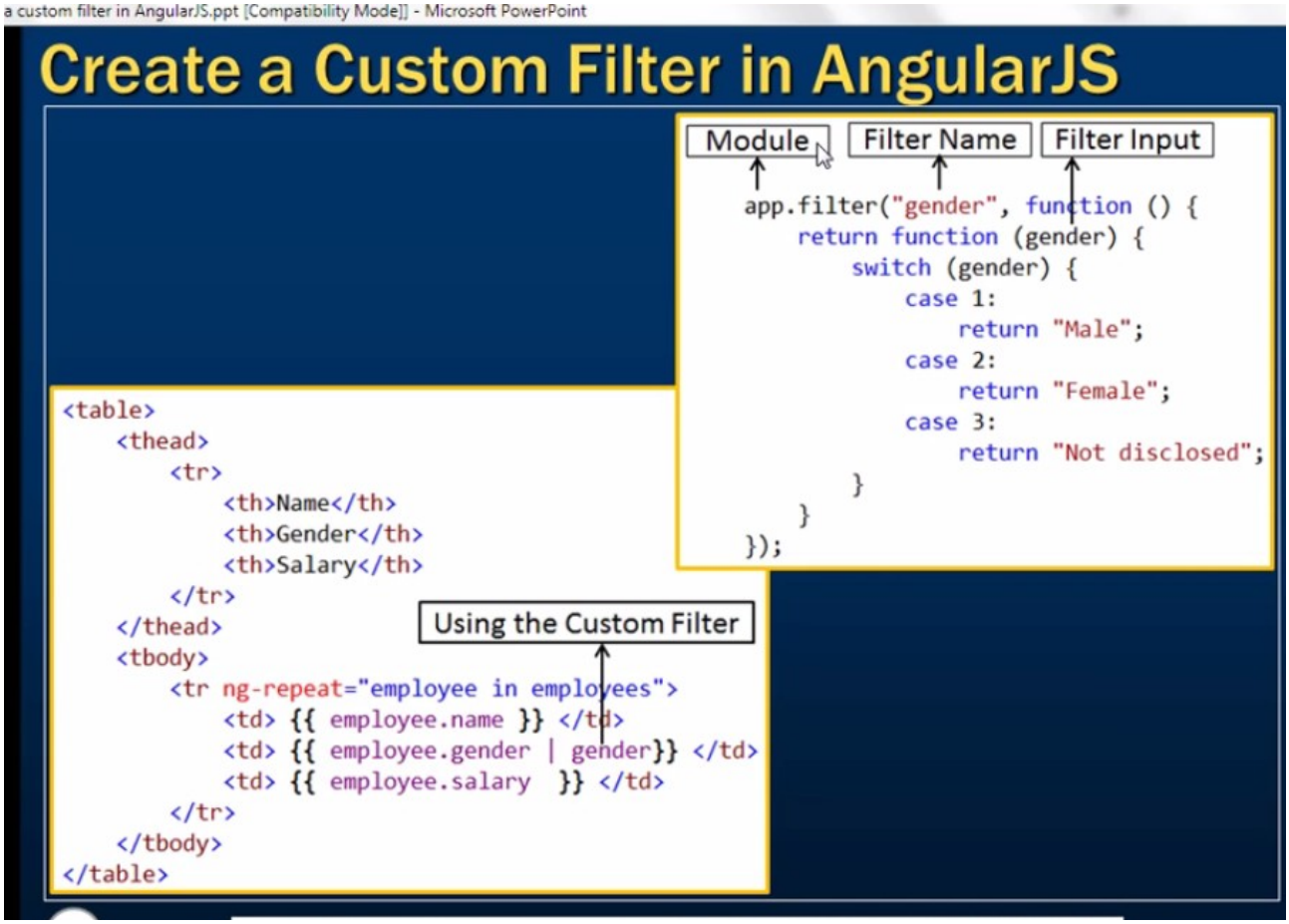
Name	Gender	Salary
Ben	Male	55000
Sara	Female	68000
Mark	Male	57000
Pam	Female	53000
Todd	Not disclosed	60000

First Approach:

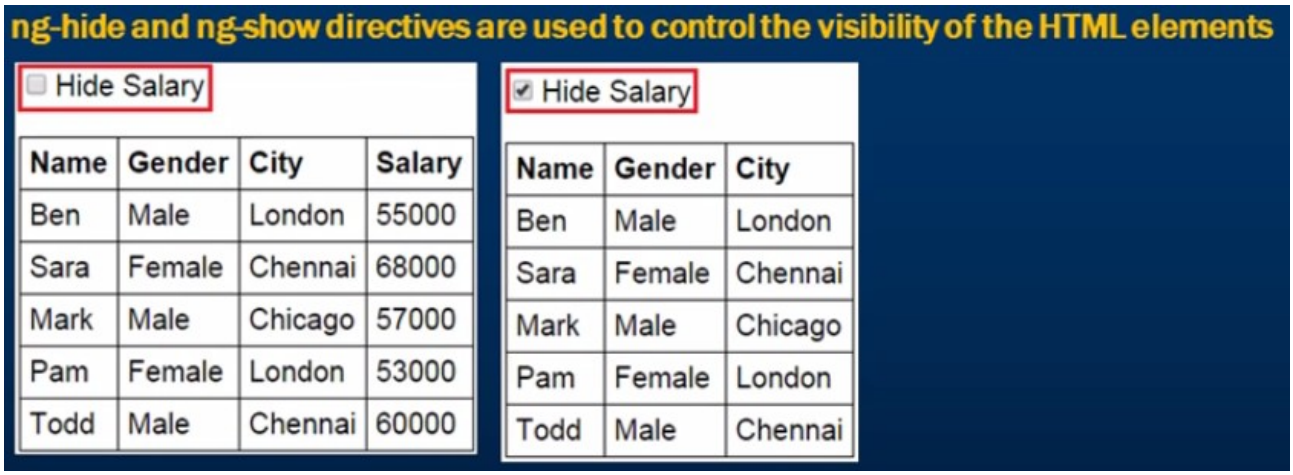
```
var app = angular
        .module("myModule", [])
        .filter("gender", function () {
            return function (gender) {
                switch (gender) {
                    case 1:
                        return "Male";
                    case 2:
                        return "Female";
                    case 3:
                        return "Not disclosed";
                }
            }
        })
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: 1, salary: 55000 },
                { name: "Sara", gender: 2, salary: 68000 },
                { name: "Mark", gender: 1, salary: 57000 },
                { name: "Pam", gender: 2, salary: 53000 },
                { name: "Todd", gender: 3, salary: 60000 }
```

```
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>Salary</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td> {{ employee.name }} </td>
                    <td> {{ employee.gender | gender }} </td>
                    <td> {{ employee.salary  }} </td>
                </tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

Second Approach: write filter in another JS file



ng-hide and ng-show in AngularJS



```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: "Male", city: "London", salary: 55000 },
                { name: "Sara", gender: "Female", city: "Chennai", salary: 68000 },
                { name: "Mark", gender: "Male", city: "Chicago", salary: 57000 },
                { name: "Pam", gender: "Female", city: "London", salary: 53000 },
                { name: "Todd", gender: "Male", city: "Chennai", salary: 60000 }
            ];

            $scope.employees = employees;
        });
```

```
<input type="checkbox" ng-model="hideSalary" />Hide Salary
<br /><br />
<table>
    <thead>
        <tr>
            <th>Name</th>
            <th>Gender</th>
            <th>City</th>
            <th ng-hide="hideSalary">Salary</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="employee in employees">
            <td> {{ employee.name }} </td>
            <td> {{ employee.gender}} </td>
            <td> {{ employee.city}} </td>
            <td ng-hide="hideSalary"> {{ employee.salary  }} </td>
        </tr>
    </tbody>
</table>
```

Mask and unmask Salary column values

☐ Hide Salary

Name	Gender	City	Salary
Ben	Male	London	55000
Sara	Female	Chennai	68000
Mark	Male	Chicago	57000
Pam	Female	London	53000
Todd	Male	Chennai	60000

☑ Hide Salary

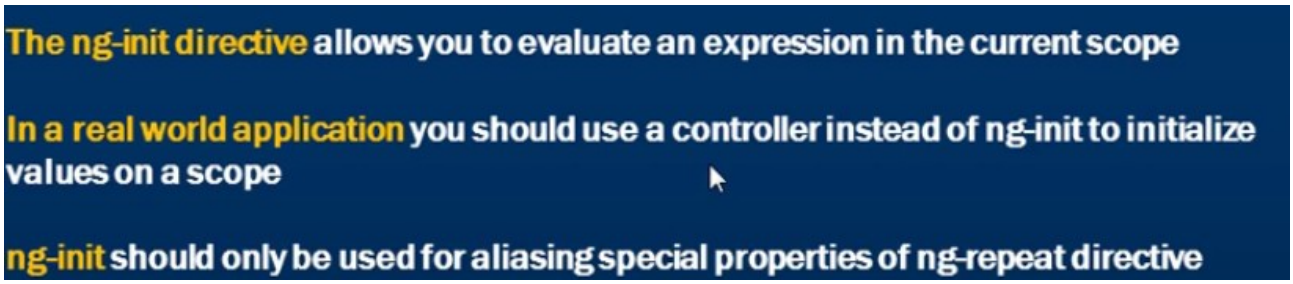
Name	Gender	City	Salary
Ben	Male	London	#####
Sara	Female	Chennai	#####
Mark	Male	Chicago	#####
Pam	Female	London	#####
Todd	Male	Chennai	#####

```
<input type="checkbox" ng-model="hideSalary" />Hide Salary
<br /><br />
<table>
    <thead>
        <tr>
            <th>Name</th>
            <th>Gender</th>
            <th>City</th>
            <th ng-hide="hideSalary">Salary</th>
            <th ng-show="hideSalary">Salary</th>
        </tr>
    </thead>
    <tbody>
        <tr ng-repeat="employee in employees">
            <td> {{ employee.name }} </td>
            <td> {{ employee.gender}} </td>
            <td> {{ employee.city}} </td>
            <td ng-hide="hideSalary"> {{ employee.salary  }} </td>
            <td ng-show="hideSalary"> ##### </td>
        </tr>
    </tbody>
</table>
```

AngularJS ng-init Directive :

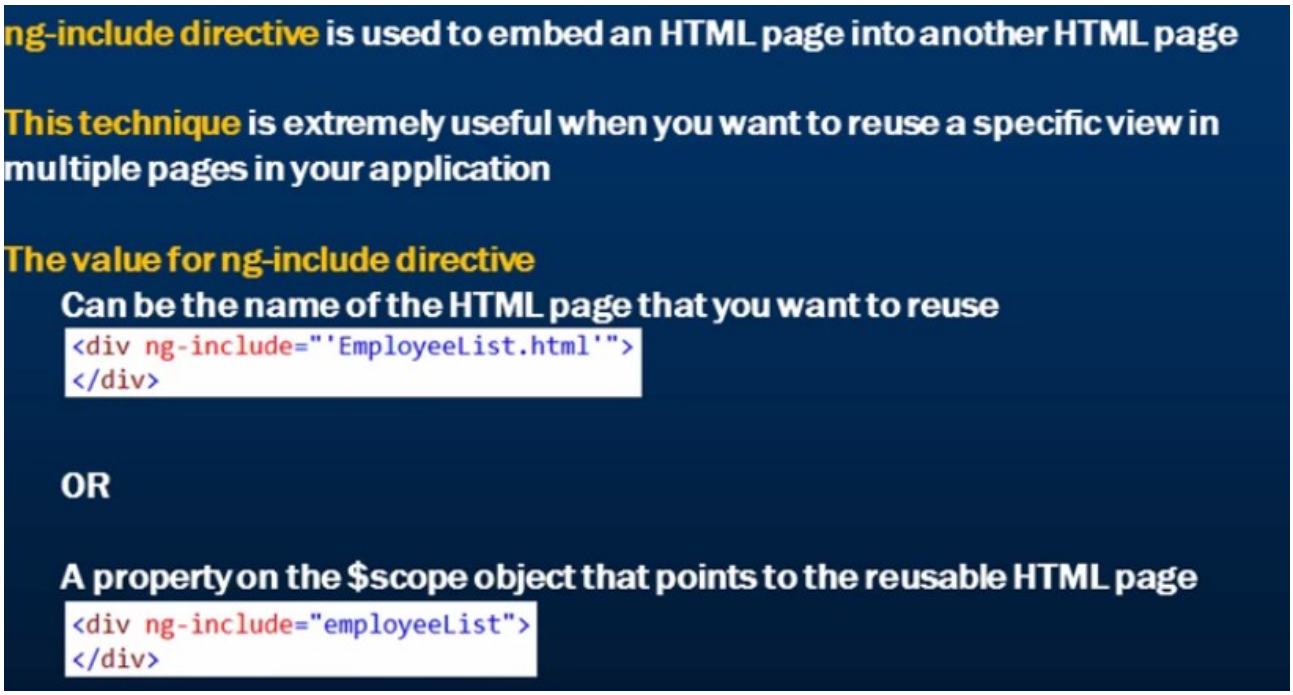
AngularJS ng-init Directive allows you to evaluate an expression in the current scope.

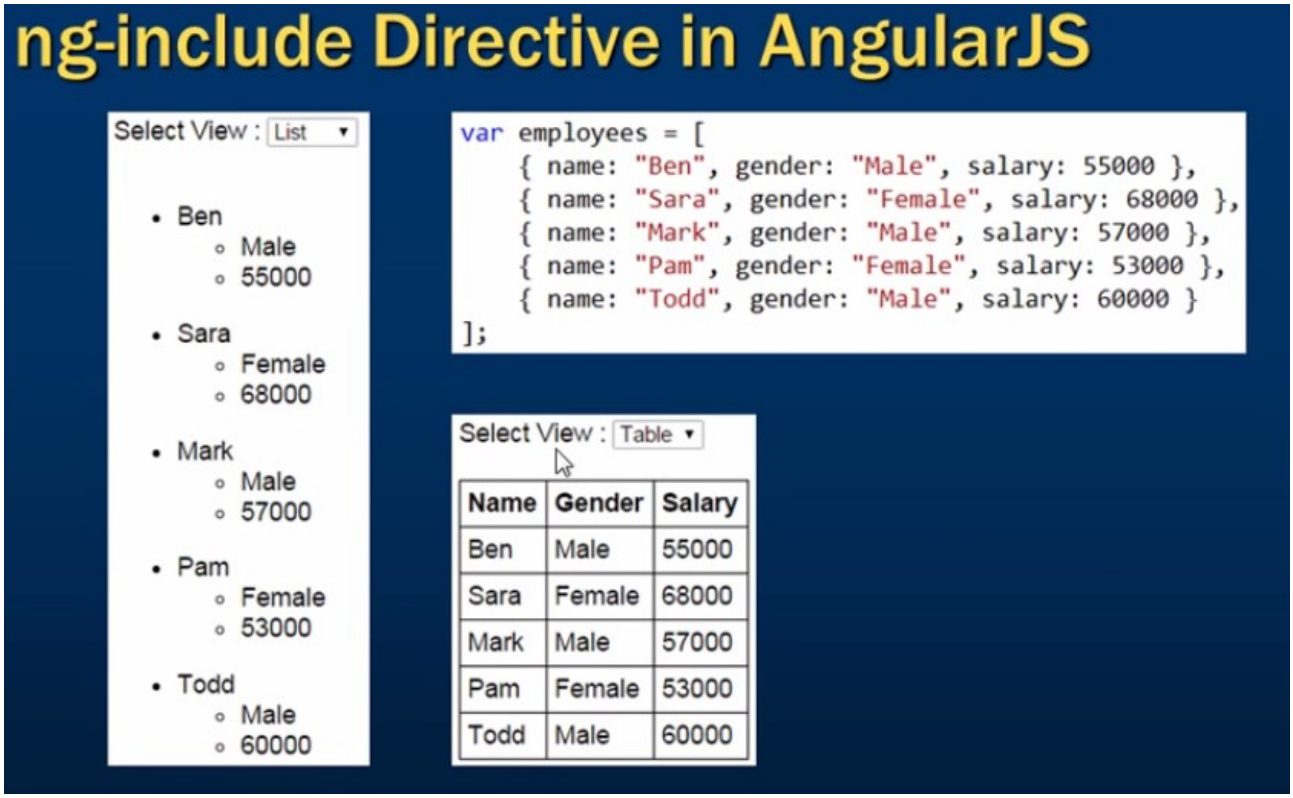
```
<body ng-app>
    <div ng-init="employees = [
                { name: 'Ben', gender: 'Male', city: 'London' },
                { name: 'Sara', gender: 'Female', city: 'Chennai' },
                { name: 'Mark', gender: 'Male', city: 'Chicago' },
                { name: 'Pam', gender: 'Female', city: 'London' },
                { name: 'Todd', gender: 'Male', city: 'Chennai' }
            ]">

        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Gender</th>
                    <th>City</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="employee in employees">
                    <td>{{employee.name}}</td>
                    <td>{{employee.gender}}</td>
                    <td>{{employee.city}}</td>
                </tr>
```




```
<body ng-app="myModule">
    <div ng-controller="myController">
       <ul>
           <li ng-repeat="country in countries" ng-init="parentIndex = $index">
               {{country.name}}
               <ul>
                   <li ng-repeat="city in country.cities">
                       {{city.name}} - Parent Index = {{ parentIndex }}, Index = {{$index}}
                   </li>
               </ul>
           </li>
       </ul>

    </div>
</body>
```

AngularJS ng-include Directive :





```
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {

            var employees = [
                { name: "Ben", gender: "Male", salary: 55000 },
                { name: "Sara", gender: "Female", salary: 68000 },
                { name: "Mark", gender: "Male", salary: 57000 },
                { name: "Pam", gender: "Female", salary: 53000 },
                { name: "Todd", gender: "Male", salary: 60000 }
            ];

            $scope.employees = employees;
            $scope.employeeView = "EmployeeTable.html";
        });
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-app="myModule">
    <div ng-controller="myController">
        Select View :
        <select ng-model="employeeView">
            <option value="EmployeeTable.html">Table</option>
            <option value="EmployeeList.html">List</option>
        </select>
        <br /><br />
        <div ng-include="employeeView"></div>
    </div>
</body>
</html>
```

$http service in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

$http service is used to make HTTP requests to remote server

$http service is a function that has a single input parameter i.e a configuration object

Example : The following example issues a GET request to the specified URL

```
$http({
    method: 'GET',
    url: 'EmployeeService.asmx/GetAllEmployees'
});
```

Complete list of properties supported by the configuration object
https://docs.angularjs.org/api/ng/service/$http#usage

$http service in AngularJS

Shortcut methods like get, post, put, delete etc are also available to be used with $http service

Example : Using the shortcut method get()

```
$http.get('EmployeeService.asmx/GetAllEmployees')
```

$http service returns a promise object

```
$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployees');
```

Instead use the **then()** method

```
$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployees')
                        .then(function (response) {
                            $scope.employees = response.data;
                        });
```

Use the $log service to log the response object to the console

```
$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployees')
                        .then(function (response) {
                            $scope.employees = response.data;
                            $log.info(response);
                        });
```

$http service in AngularJS

If there is an error processing the request, the errorCallback function is called

```
$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployee')
                        .then(function (response) {
                            $scope.employees = response.data;
                        }, function (reason) {
                            $scope.error = reason.data;
                        });
```

You can also create separate functions and associate them as successCallback and errorCallback functions

```
var successCallBack = function (response) {
    $scope.employees = response.data;
};

var errorCallBack = function (reason) {
    $scope.error = reason.data;
}

$scope.employees = $http.get('EmployeeService.asmx/GetAllEmployees')
                        .then(successCallBack, errorCallBack);
```

Default Transformations provided by Angular's http service

- If the data property of the request configuration object contains a JavaScript object, it is automatically converted into JSON object
- If JSON response is detected, it is automatically converted into a JavaScript object

AngularJS Services

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

What is a service in AngularJS

A service in Angular is simply an object that provide some sort of service that can be reused with in an angular application

Why do we need services

Services encapsulate reusable logic that does not belong anywhere else (i.e Directives, Filters, Views, Models, & Controllers)

What are the benefits of using services

- Reusability
- Dependency Injection
- Testability

Next Video : How to create a custom service, register and use it

Create custom service in AngularJS

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

Your String	AngularVideoTutorial
Result	Angular Video Tutorial

```
// All logic in the controller. No custom service used
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope) {
            $scope.transformString = function (input) {
                if (!input)
                    return input;

                var output = "";

                for (var i = 0; i < input.length; i++) {
                    if (i > 0 && input[i] == input[i].toUpperCase()) {
                        output = output + " ";
                    }

                    output = output + input[i];
                }

                $scope.output = output;
            };
        });
```

```
// factory method is used to create and register the service with Angular
app.factory('stringService', function () {
    return {
        processString: function (input) {
            if (!input)
                return input;

            var output = "";

            for (var i = 0; i < input.length; i++) {
                if (i > 0 && input[i] == input[i].toUpperCase()) {
                    output = output + " ";
                }

                output = output + input[i];
            }

            return output;
        }
    };
});
```

```
// The controller is now simple and clean. Notice the stringService injection
var app = angular
        .module("myModule", [])
        .controller("myController", function ($scope, stringService) {
            $scope.transformString = function (input) {
                $scope.output = stringService.processString(input);
            };
        });
```

AngularJS anchorscroll example

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

$anchorscroll service is used to jump to a specified element on the page

$location service hash method appends hash fragments to the URL

$anchorscroll() method reads the hash fragment in the URL and jumps to that element on the page

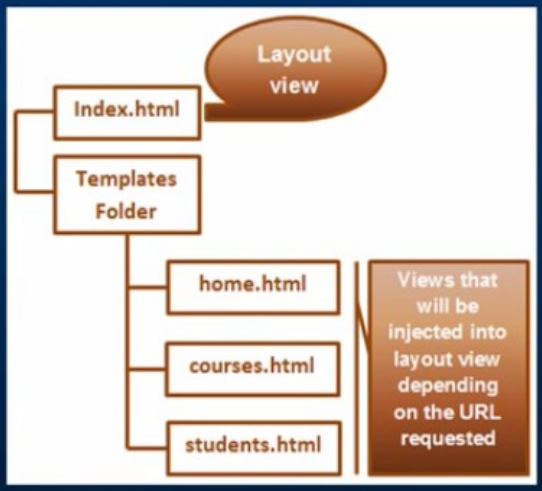
yOffset property specifies the vertical scroll-offset

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" ng-app="demoApp">
<head>
    <title></title>
    <script src="Scripts/angular.js"></script>
    <script src="Scripts/Script.js"></script>
    <link href="Styles.css" rel="stylesheet" />
</head>
<body ng-controller="demoController">
    <button id="top" ng-click="scrollTo('bottom')">Go to bottom of the page</button>
    <br /><br />
    <div>...</div>
    <br />
    <button id="bottom" ng-click="scrollTo('top')">Go to top of the page</button>
</body>
</html>
```

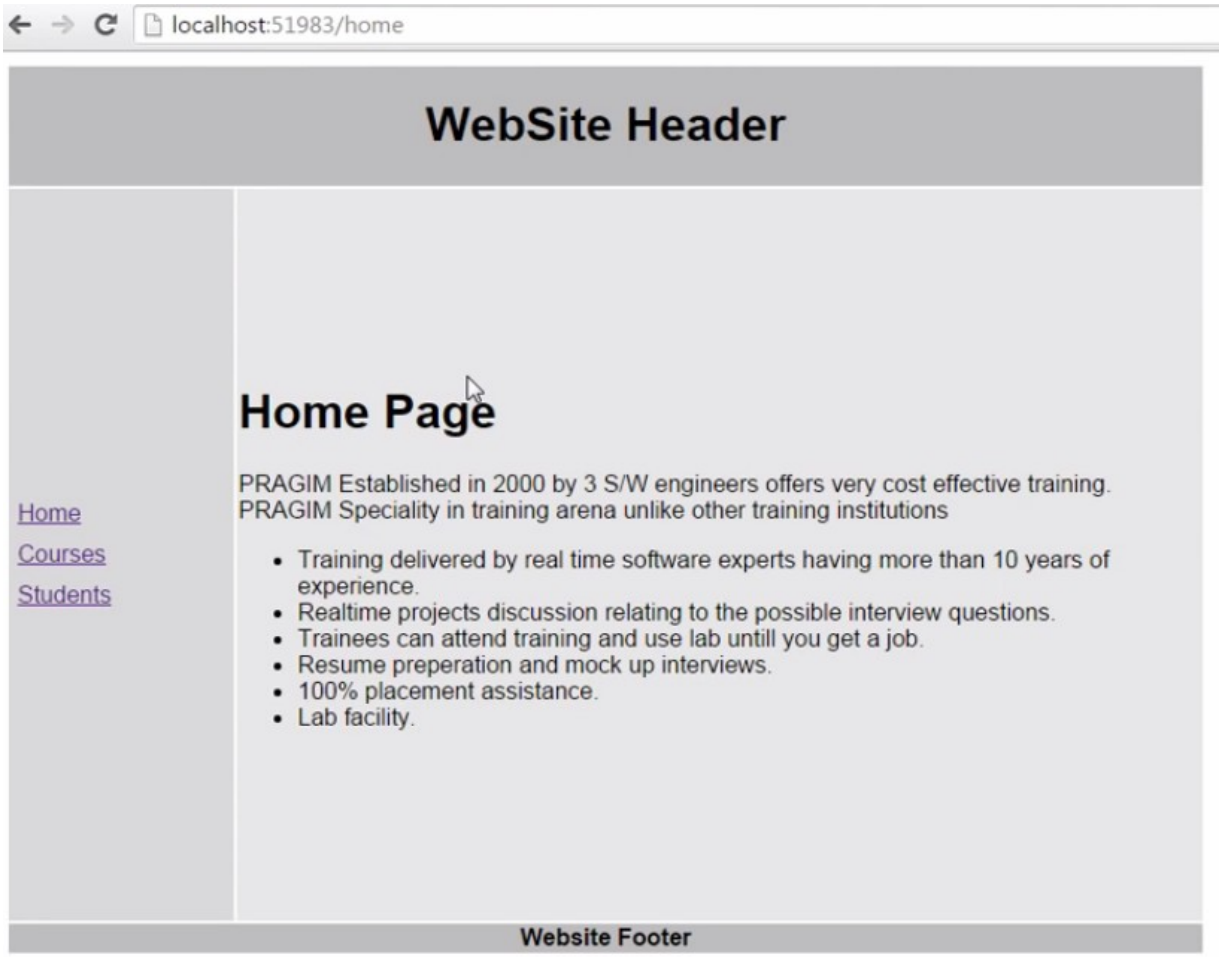
```javascript
// <reference path="angular.js" />

var demoApp = angular.module("demoApp", [])
                    .controller("demoController", function ($scope, $location, $anchorScroll)
                        $scope.scrollTo = function (scrollLocation) {
                            $location.hash(scrollLocation);
                            $anchorScroll.yOffset = 20;
                            $anchorScroll();
                        }
                    });
```

AngularJS Routing Tutorial

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx



Please note : The AngularJS Route module is present in a separate JavaScript file.



Index.html

```
<!DOCTYPE html>
<html ng-app="Demo">
<head>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.js"></script>
    <script src="http://localhost:8080/SampleProject/script.js"></script>
    <link href="http://localhost:8080/SampleProject/Style.css" rel="stylesheet" type="text/css"/>

    <meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <table style="font-family: Arial">
        <tr>
            <td colspan="2" class ="header">
                <h1> WebSite Header </h1>
            </td>
        </tr>
        <tr>
            <td  class ="leftMenu">
                <a href="#/home">Home</a>
                <a href="#/courses">Courses</a>
                <a href="#/students">Students</a>
            </td>
            <td  class ="mainContent">
                <ng-view></ng-view>
            </td>
        </tr>
        <tr>
            <td colspan="2" class ="footer">
                <h1> WebSite Footer </h1>
            </td>
        </tr>
    </table>
</body>
</html>
```

```
.header {
    width: 800px;
    height: 80px;
    text-align: center;
    background-color: #BDBDBD;
}

.footer {
    text-align: center;
    background-color: #BDBDBD;
}

.leftMenu {
    width: 150px;
    height: 500px;
    background-color: #D8D8D8;
}

.mainContent {
    width: 650px;
    height: 500px;
    background-color: #E6E6E6;
}
a{
    display: block;padding: 5px;
}
```

```
HOME.html
=========================
<h1>{{message}}</h1>
<div>
    Welcome to ng-binding.
</div>

<ul>
<li>The binding goes both ways.</li>
<li>If the user changes the value inside the input field, </li>
<li>the AngularJS property will also change its value:</li>
<li>The ng-model directive can provide type validation</li>
<li> for application data (number, e-mail, required):</li>
<li>In the example above,</li>
</ul>

CORSES.html
=========================
<h1> Courses we offer </h1>

<ul>
    <li ng-repeat="course in courses">
    {{course}}
    </li>

</ul>
STUDENT.html
================================
<h1> List of students </h1>

<ul>
    <li ng-repeat="student in students">
    {{student.name}}
    </li>

</ul>
```

```
*/
var app = angular
            .module("Demo", ["ngRoute"])
            .config(function($routeProvider) {
                $routeProvider
                    .when("/home", {
                        templateUrl: "Templates/home.html",
                        controller: "homeController"
                    })
                    .when("/courses", {
                        templateUrl: "Templates/courses.html",
                        controller: "courseController"
                    })
                    .when("/students", {
                        templateUrl: "Templates/students.html",
                        controller: "studentController"
                    })
                })
                .controller("homeController", function($scope) {
                    $scope.message="Home Page";
                })
                .controller("courseController", function($scope) {
                    $scope.courses = ["C#","JAVA","SQL Server","Spring"];
                })
                .controller("studentController", function($scope) {
                    $scope.students = [{name:"Rams",age:26,sex:"Male"},
                                       {name:"Ranga",age:26,sex:"Male"},
                                       {name:"Anu",age:26,sex:"Male"},
                                       {name:"Nithya",age:26,sex:"Male"},
                                       {name:"Nivetha",age:26,sex:"Female"}
                                       ];
                })
```

AngularJS CONTROLLER AS Syntax

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

Controller Code with $scope

```
app.controller("mainController", function ($scope) {
    $scope.message = "Hello Angular";
});
```

View Code with $scope

```
<h1 ng-controller="mainController">{{message}}</h1>
```

Controller Code with CONTROLLER AS syntax

```
app.controller("mainController", function () {
    this.message = "Hello Angular";
});
```

View Code with CONTROLLER AS syntax

```
<h1 ng-controller="mainController as main">{{main.message}}</h1>
```

AngularJS CONTROLLER AS Syntax

If you use "this" keyword in "then()" function as shown below, you would not get the result you expect. That's because "this" keyword points to the "window object" when the control comes to "then()" function

```
.controller("studentsController", function ($http) {
    $http.get("StudentService.asmx/GetAllStudents")
                        .then(function (response) {
                            this.students = response.data;
                        })
})
```

AngularJS CONTROLLER AS Syntax

When defining routes CONTROLLER AS syntax can be used in one of these 2 ways

```
$routeProvider
    .when("/home", {
        templateUrl: "Templates/home.html",
        controller: "homeController as homeCtrl"
    })
```

OR

```
$routeProvider
    .when("/home", {
        templateUrl: "Templates/home.html",
        controller: "homeController",
        controllerAs: "homeCtrl"
    })
```

Next Video : How the CONTROLLER AS syntax can make our code more readable when working with nested scopes

Nested scopes & ControllerAs Syntax

India
India - Maharashtra
India - Maharashtra - Mumbai

```
<div ng-controller="countryController">
    {{name}}
    <div ng-controller="stateController">
        {{$parent.name}} - {{name}}
        <div ng-controller="cityController">
            {{$parent.$parent.name}} - {{$parent.name}} - {{name}}
        </div>
    </div>
</div>
```

Nested scopes & ControllerAs Syntax

```
var app = angular
            .module("Demo", [])
            .controller("countryController", function () {
                this.name = "India";
            })
            .controller("stateController", function () {
                this.name = "Maharashtra";
            })
            .controller("cityController", function () {
                this.name = "Mumbai";
            });
```

India
India - Maharashtra
India - Maharashtra - Mumbai

```
<div ng-controller="countryController as countryCtrl">
    {{countryCtrl.name}}
    <div ng-controller="stateController as stateCtrl">
        {{countryCtrl.name}} - {{stateCtrl.name}}
        <div ng-controller="cityController as cityCtrl">
            {{countryCtrl.name}} - {{stateCtrl.name}} - {{cityCtrl.name}}
        </div>
    </div>
</div>
```

AngularJS CONTROLLER AS vs $scope

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

CONTROLLER AS syntax is new and is officially released in 1.2.0. $scope is the old technique and is available since the initial version of angular is released.

You can use either one of thes techniques. Both have their own uses. For example, CONTROLLER AS syntax makes your code more readable when working with nested scopes.

If you want to use $scope it has to be injected into controller function, where as with CONTROLLER AS syntax there is no need for such injection, unless you need it for something else

Angular Case Insensitive Match

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

Routes are case sensitive by default. To make the route case-insensitive set caseInsensitiveMatch property to true.

```
$routeProvider
    .when("/home", {
        templateUrl: "Templates/home.html",
        controller: "homeController",
        controllerAs: "homeCtrl",
        caseInsensitiveMatch: true
    })
```

To make all routes case-insensitive set caseInsensitiveMatch property on $routeProvider

```
$routeProvider.caseInsensitiveMatch = true;
```

Inline Templates

View content can also come from an inline template. To use an inline template use template property.

```
$routeProvider
    .when("/home", {
        template: "<h1>Inline Template in action</h1>",
        controller: "homeController",
        controllerAs: "homeCtrl"
    })
```

AngularJS Route Reload

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

Angular route service reload() method is useful when you want to reload just the current route instead of the entire app

```
// Controller Code
.controller("studentsController", function ($http, $route) {
    var vm = this;

    vm.reloadData = function () {
        $route.reload();
    }

    $http.get("StudentService.asmx/GetAllStudents")
                        .then(function (response) {
                            vm.students = response.data;
                        })
})
```

```
<!--View HTML-->
<button ng-click="studentsCtrl.reloadData()">Reload</button>
```

$scope v/s $rootScope

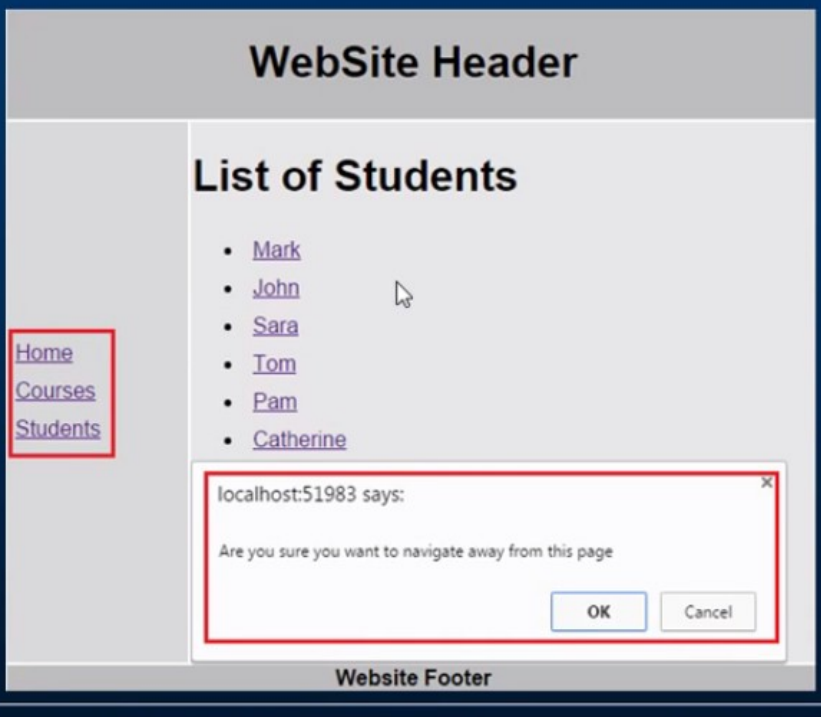
```
<!--View HTML-->
<div ng-controller="redColourController">
    Root Scope Colour : {{rootScopeColour}} <br />
    Red Colour Controller : {{redColour}} <br />
    Green Colour Controller :
        <span style="color:red" ng-show="greenColour == undefined">
            greenColour is undefined
        </span>
</div>

<br />

<div ng-controller="greenColourController">
    Root Scope Colour : {{rootScopeColour}} <br />
    Green Colour Controller : {{greenColour}} <br />
    Red Colour Controller :
        <span style="color:red" ng-show="redColour == undefined">
            redColour is undefined
        </span>
</div>
```

Root Scope Colour : I am Root Scope Colour
Red Colour Controller : I am Red Colour
Green Colour Controller : greenColour is undefined

Root Scope Colour : I am Root Scope Colour
Green Colour Controller : I am Green Colour
Red Colour Controller : redColour is undefined

AngularJS Cancel Route Change

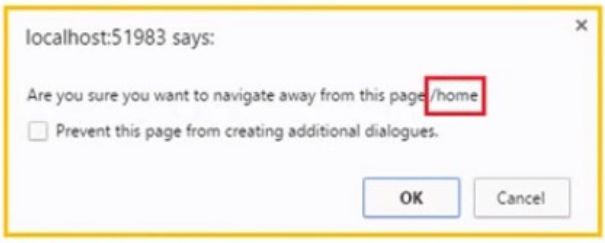
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx



AngularJS Cancel Route Change

Include the route that the user is trying to navigate to in the confirmation message

```
.controller("studentsController", function ($http, $route, $scope) {
    var vm = this;

    $scope.$on("$routeChangeStart", function (event, next, current) {
        if (!confirm("Are you sure you want to navigate away from this page to "
                            + next.$$route.originalPath)) {
            event.preventDefault();
        }
    });

    vm.reloadData = function () {
        $route.reload();
    }

    $http.get("StudentService.asmx/GetAllStudents")
                .then(function (response) {
                    vm.students = response.data;
                })
})
```



AngularJS Cancel Route Change

You can also cancel route change by handling $locationChangeStart event

```
.controller("studentsController", function ($http, $route, $scope) {
    var vm = this;

    $scope.$on("$locationChangeStart", function (event, next, current) {
        if (!confirm("Are you sure you want to navigate away from this page to "
            + next)) {
            event.preventDefault();
        }
    });

    vm.reloadData = function () {
        $route.reload();
    }

    $http.get("StudentService.asmx/GetAllStudents")
                .then(function (response) {
                    vm.students = response.data;
                })
})
```

Next Video : Different events that are triggered when a route change occurs

AngularJS Route Change Events

If you are in need of the DVD with all the videos and PPT's, please visit http://pragimtech.com/order.aspx

When route navigation occurs in an Angular application, the following events are triggered

1. $locationChangeStart
2. $routeChangeStart
3. $locationChangeSuccess
4. $routeChangeSuccess

AngularJS Route Change Events

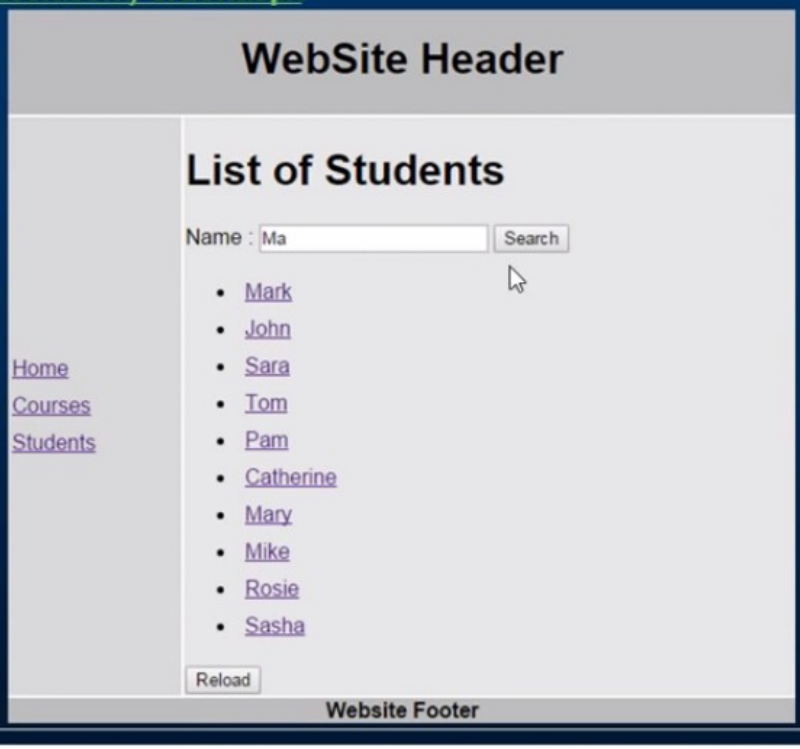
```
.controller("studentsController", function ($http, $route, $rootScope, $log) {
    var vm = this;

    $rootScope.$on("$locationChangeStart", function () {
        $log.debug("$locationChangeStart fired");
    });

    $rootScope.$on("$routeChangeStart", function () {
        $log.debug("$routeChangeStart fired");
    });

    $rootScope.$on("$locationChangeSuccess", function () {
        $log.debug("$locationChangeSuccess fired");
    });

    $rootScope.$on("$routeChangeSuccess", function () {
        $log.debug("$routeChangeSuccess fired");
    });

    vm.reloadData = function () {
        $route.reload();
    }

    $http.get("StudentService.asmx/GetAllStudents")
                .then(function (response) {
                    vm.students = response.data;
                })
})
```

AngularJS Route Change Events

```
.controller("studentsController", function ($http, $route, $scope, $log) {
    var vm = this;

    $scope.$on("$routeChangeStart", function (event, next, current) {
        $log.debug("RouteChangeStart Event");
        $log.debug(event);
        $log.debug(next);
        $log.debug(current);
    });

    $scope.$on("$locationChangeStart", function (event, next, current) {
        $log.debug("LocationChangeStart Event");
        $log.debug(event);
        $log.debug(next);
        $log.debug(current);
    });

    vm.reloadData = function () {
        $route.reload();
    }

    $http.get("StudentService.asmx/GetAllStudents")
                .then(function (response) {
                    vm.students = response.data;
                })
})
```

AngularJS optional URL parameters

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx



AngularJS optional URL parameters

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

AngularJS Route Resolve

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

```
.when("/students", {
    templateUrl: "Templates/students.html",
    controller: "studentsController",
    controllerAs: "studentsCtrl",
    resolve: {
        studentslist: function ($http) {
            return $http.get("StudentService.asmx/GetAllStudents")
                    .then(function (response) {
                        return response.data;
                    })
        }
    }
})
```

AngularJS Route Resolve

If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

```
.controller("studentsController", function (studentslist, $route, $location) {
    var vm = this;

    vm.studentSearch = function () {
        if (vm.name)
            $location.url("/studentsSearch/" + vm.name)
        else
            $location.url("/studentsSearch")
    }

    vm.reloadData = function () {
        $route.reload();
    }

    vm.students = studentslist;
})
```

AngularJS Route Resolve

So in summary

1. The resolve property contains one or more promises that must resolve successfully before transitioning to the new route
2. The property names used in the resolve property can then be injected into the controller. This means the controller does not have to fetch the data