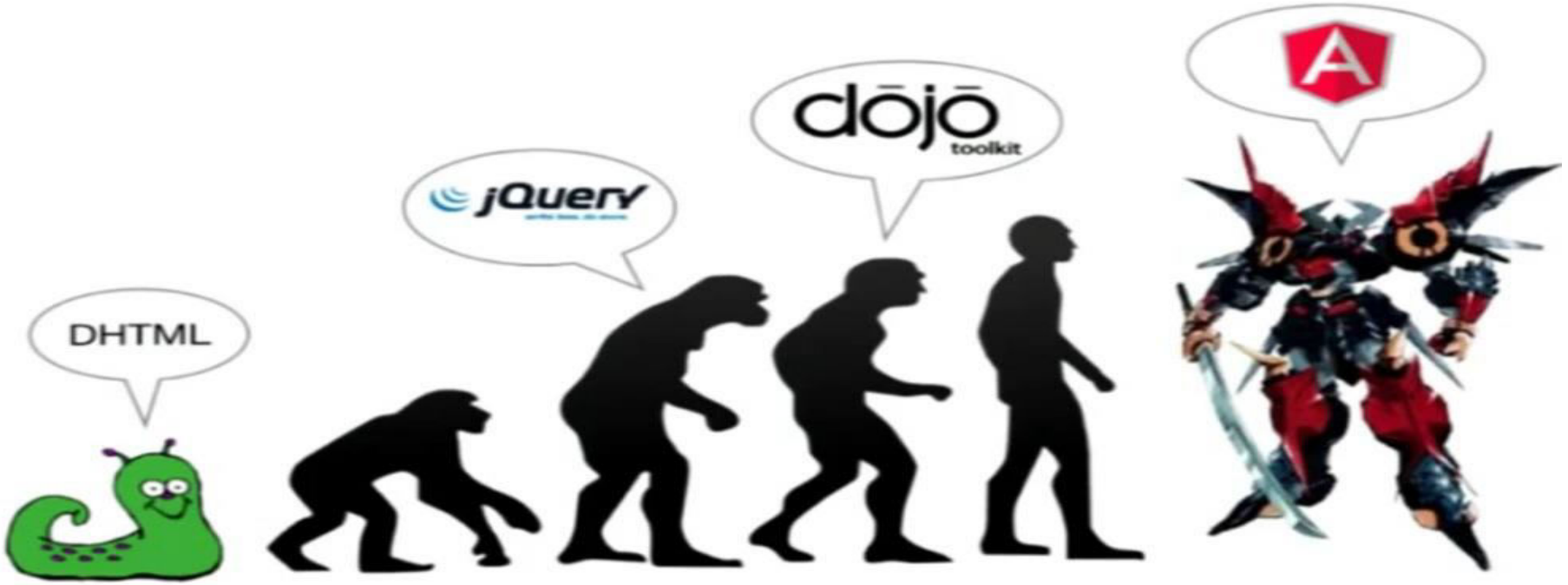
UNDERSTANDING

SIVA SUBRAMANIAN

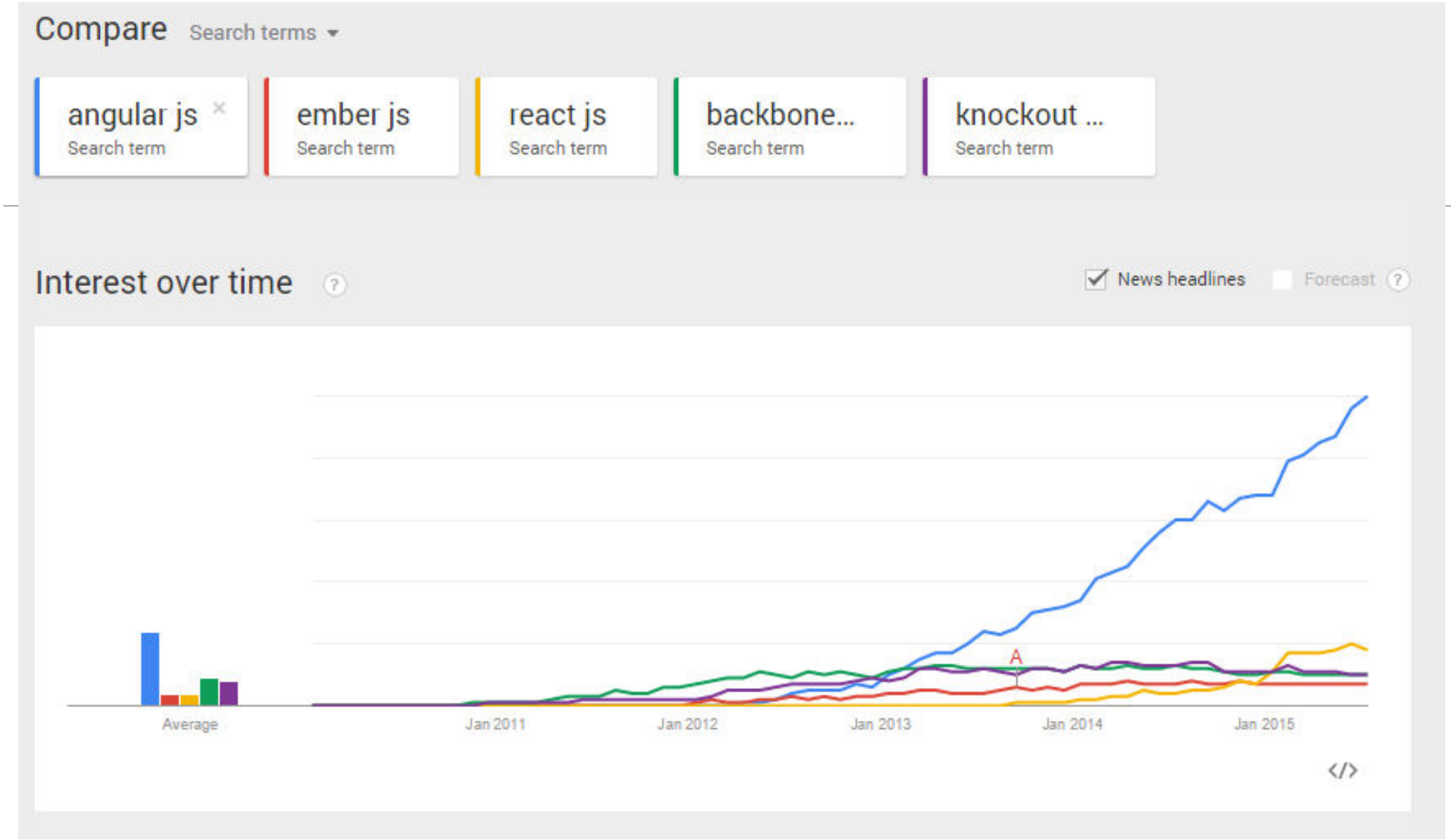
Software Engineer @ Market Simplified India Ltd.

Evolution of Web Apps



What is Angular JS?

Angular JS is an **open-source JavaScript Framework** maintained by **Google** for building **Single Page Applications (SPAs).** Its goal is to augment browser-based application with **Model-View-Controller (MVC)** capability.


Compare
Search terms
angular js
Search term
ember js
Search term
react js
Search term
backbone...
Search term
knockout ...
Search term
Interest over time
News headlines
Forecast
A
Average
Jan 2011
Jan 2012
Jan 2013
Jan 2014
Jan 2015

What is Angular JS?

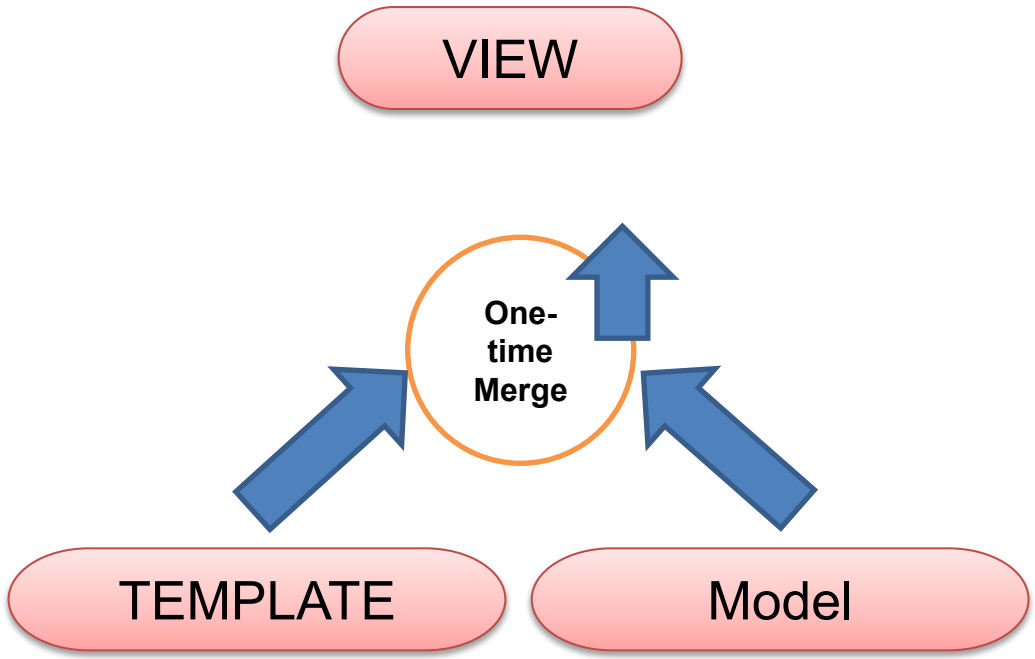
- Client-side JS Framework for **SPA**
  - Not just a single piece of a puzzle but **full client side solution**
- **Model-View-Controller** framework
- For front-end of your application
- Steroids for your UI

Core Features of **Angular JS**

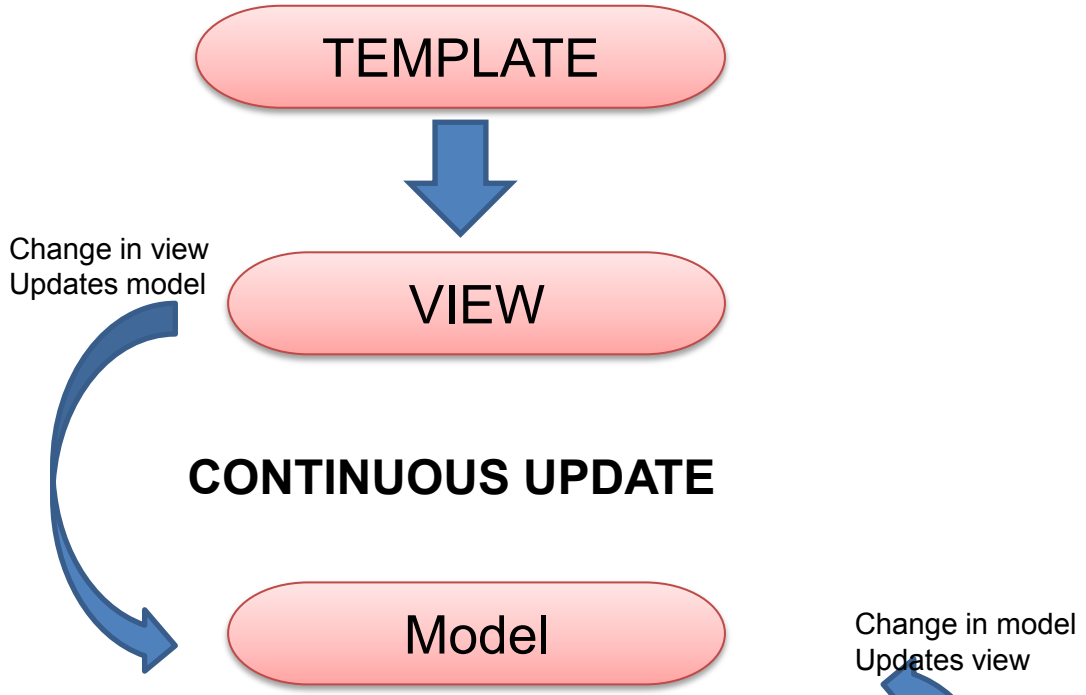
DISCLAIMER : THERE WILL BE CODE SNIPPETS IN THE SLIDES WHICH ARE ONLY FOR DEMO PURPOSE AND MIGHT NOT BE TOTALLY ACCURATE SYNTACTICALLY. IT IS ONLY MEANT TO GIVE A BRIEF OVERVIEW OF HOW THINGS WORK.

Two-way Data Binding

ONE WAY DATA BINDING



TWO WAY DATA BINDING



MVC Framework

- The whole application has 3 major components Model, View and Controller
- Model is the data layer
- View is the UI layer
- Controller is the logic layer
- Actually, MVVM (Model-View-ViewModel) architecture

Templates

Templates

- Templates are plain old HTML
- Extended HTML Vocabulary to contain instructions on how to combine model data into view
- Its **NOT** HTML string manipulation *(one of the major differences from other frameworks)*

Templates

```
function AlbumCtrl($scope) {
    scope.images = [
        {"thumbnail":"img/image_01.png", "description":"Image 01 description"},
        {"thumbnail":"img/image_02.png", "description":"Image 02 description"},
        {"thumbnail":"img/image_03.png", "description":"Image 03 description"}
    ];
}
```

```
<div ng-controller="AlbumCtrl">
    <ul>
        <li ng-repeat="image in images">
            <img ng-src="{{image.thumbnail}}" alt="{{image.description}}">
        </li>
    </ul>
</div>
```

Dependency Injection

- Built-in Dependency Injection Subsystem
- Easier to understand and test
- Modular Development
- Just ask for things that you want to use (built-in or custom services)

Dependency Injection

```
angular.
        module('MyAppModule',
          ['ModuleIDownloaded', 'MyAnotherModule']
        );
```

```
function AlbumCtrl($scope, $location, MyAwesomeService) {
    // Write something Amazing Here...
}
```

Directives

- Extend HTML vocabulary to give
- them superpowers
- In-built and custom directives

Core Concepts of **Angular JS**

Modules

- Container for different parts of your application – controllers, services, filters, etc.
- Declaratively specify how an application is to be bootstrapped
- Builds reusable component packages
- Can be loaded in any order (or even in parallel)

Modules

```
// declaring a module
var myApp = angular.module('myAwesomeApp',[dependencies, ...]);

// Different components of a module
// Config Block
myApp.config(function(injectables, ...){
    // Can have as many as we want
    // Can only inject "Providers"
    // Define configurations for the module
    // Gets executed during configuration phase
});

// Run Block
myApp.run(function(injectables, ...){
    // Can have as many as we want
    // Can only inject "Instances"
    // Gets executed to kickstart the application
});
```

Modules

```
// Controllers
myApp.controller('myAwesomeController', function(injectables){

});

//Factories, Directives and Filters
myApp.
    factory('myFactory', function(){return 123;}).
    directive('myDirective', .....).
    filter('myFilter', .....)
```

Modules

```
<body ng-app>
    <!-- Will make angularjs available but will not -->
    <!-- bootstrap any module -->
    <!-- Your awesome app HTML here -->
</body>

<body ng-app="myApp">
    <!-- Auto-bootstrap myApp module -->
    <!-- Your awesome app HTML here -->
</body>
```

Modules

```
1  <script>
2      angular.bootstrap(document,['myApp', otherModulesMayBe...]);
3  </script>
```

Modules

- A module for each feature
- A module for reusable features
- An application level module which will depend on above modules and will be auto-bootstrapped.

Controllers

- Logic behind the view
- Constructs the Model and publishes it to the View
- Instantiate the ViewModel object or “$scope”
- Set up the initial state of the $scope
- Add behavior to $scope

Controllers

Writing Controllers

```
<body ng-app>
    <div ng-controller="myAwesomeController">
        Hello {{name}}
    </div>
</body>
<script src="angular.js"></script>
<script>
    function myAwesomeController($scope) {
        $scope.name = "World!";
    }
</script>
```

Writing Controllers

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

controllers = {
    myAwesomeController : function($scope) {
        $scope.name = "World";
    },
    anotherController : function($scope) {
        // Something
    },
    oneMoreController : function($scope){
        // Something
    }
}

app.controller(controllers);
```

```
<body ng-app="myAwesomeApp">
    <div ng-controller="myAwesomeController">
        Hello {{name}}
        <div ng-controller="anotherController"
        >
            <!-- Something Else -->
        </div>
    </div>
    <div ng-controller="oneMoreController"></
    div>
</body>
```

Writing Controllers

```
// app.js
var app = angular.module('myAwesomeApp');

// myAwesomecontroller.js
app.controller('myAwesomeController',
    ['$scope', otherInjectables,
    function($scope, otherInjectables){
        $scope.name = "World";
    }]);

// anotherController.js
app.controller('myAwesomeController',
    ['$scope', otherInjectables,
    function($scope, otherInjectables){
        $scope.name = "World";
    }]);
```

```
<body ng-app="myAwesomeApp">
    <div ng-controller="myAwesomeController">
        Hello {{name}}
        <div ng-controller="anotherController"
        >
            <!-- Something Else -->
        </div>
    </div>
    <div ng-controller="oneMoreController"></
    div>
</body>
```

View

- What the users see
- HTML Template that is merged with the model and finally rendered into the browser DOM

Model

- Data that is merged with the HTML template to produce views

Model

- Data that is merged with the HTML template to produce views

```
<body ng-app="myAwesomeApp">
    <div ng-controller="myAwesomeController">
        Hello {{name}}
        <input type="text" ng-model="age"
        value="22">
    </div>
</body>
```

```
Scope {
    name : 'World', //string
    value : 1234, //number
    // Object Hash
    person : {
        firstName : 'John',
        lastName : 'Doe'
    },
    age: 22
}
```

Scope

- An object that refers to the application's data-model
- $scope
- Execution context for expression
- Contains data, behaviors and other APIs to manage model mutation and events.

Scope : Scope Hierarchy

```
angular.module('myApp').
    run(function($rootScope){
        $rootScope.name = "World";
    }).
    controller('ControllerOne', ['$scope',
        function($scope){
            $scope.name = "John Doe";
        }]).
    controller('ControllerTwo', ['$scope',
        function($scope){
            $scope.names = ['Ramesh', 'Suresh'];
        }])
```

Scope : Scope Hierarchy

```
<body ng-app>
    Hello {{name}}
    <div ng-controller="ControllerOne">
        Hello {{name}}
        <div ng-controller="ControllerTwo">
            <ul>
                <li ng-repeat="name in names">
                    Hello {{name}}
                </li>
            </ul>
        </div>
    </div>
</body>
```

Hello World
Hello John Doe
Hello Ramesh
Hello Suresh

Scope : $watch

- API to observe model mutation

```
<input type="checkbox" ng-model="checkValue">

{{counter}}
```

```
$scope.counter = 0;

$scope.$watch('checkValue', function(newValue, oldValue){
    $scope.counter++;
})
```

Scope : $digest cycle

- A cycle that processes all watcher functions
- Asynchronous Dirty Checking cycle
- Not to be called directly, instead we use $apply

Scope : $apply

- Explicitly evaluate expressions in angular from outside angular
- Executes $digest after expression evaluation

Scope : Events

- **$emit('somethingHappened', args)** – dispatches event upwards the scope
- **$broadcast('somethingHappened', args)** – dispatches event downwards the scope
- **$on('somethingHappened', listenerFunction)** – listens to event fires and executes listener function.

Filters

- Formats the value of an expression for display
- Change form of data
- Return a subset of list according to some rule

Filters

```
<input type="text" ng-model="searchText">

<div ng-repeat="friend in friends | filter:searchText">
    {{friend.name}} -- {{friend.phone}}

</div>
```

Search:

Name	Phone
John	555-1276
Mary	800-BIG-MARY
Mike	555-4321
Adam	555-5678
Julie	555-8765
Juliette	555-5678

Search: ju

Name	Phone
Julie	555-8765
Juliette	555-5678

Filters : Custom

```
$scope.names = [
    'Ram Shrestha',
    'John Doe',
    'Barack Obama'
]
```

```
<div ng-repeat="name in names | seperate">
    First Name : {{name.fname}}
    Last Name : {{name.lname}}
</div>
```

Filters : Custom

```
angular.module('myApp').
    filter('seperate', function(){

        return function(input) {
            var result = [];
            angular.foreach(input,function(item){
                seperated = item.split(' ');
                newForm = {
                    fname : seperated[0],
                    lname : seperated[1]
                };

                result.push(newForm);
            });

            return result;
        }
    })
```

Services

- Injectable objects that can be used to organize and share code and functions across the application
- Could be used to share utility functions
- Angular provides useful services like $http to make AJAX requests
- We can also make custom services

Services

```
angular.module('myApp').
    factory('AwesomeService', function(){
        return {
            doSomethingAwesome : function() {
                // Something Awesome
            },
            doSomethingFunny : function(){
                // Something Funny
            }
        }
    });
```

```
angular.module('myApp').
    controller('myController', ['AwesomeService',
        function(AwesomeService){

            AwesomeService.doSomethingFunny();

            AwesomeService.doSomethingAwesome();
        }])
```

Directives

- Coolest feature of angular js
- Markers on DOM elements (attributes, element names, class names, comment) that attach specified behaviors to that DOM element, or even transform the element
- Superpowers for your DOM

Directives : In-built directives

- Ng-app
- Ng-bind
- Ng-model
- Ng-class
- Ng-controller
- Ng-show /Ng-Hide
- Ng-if
- Ng-switch

Directives : Custom Directives

```
1  <!-- Element Name -->
2  <my-dir></my-dir>
3
4  <!-- Attribute  -->
5  <span my-dir="expression"></span>
6
7  <!-- Class Name -->
8  <span class="my-dir : expression;"></span>
9
```

Directives : Custom Directives

```
1  <input type="text" auto-complete="autocompleteURL">
```

Directives : Custom Directives

```
angular.module('myApp').
    directive('autoComplete', function(){
        return {
            restrict : 'A',
            template : "<input type='text' />"+
                "<div ng-repeat='item in options'>"+
                "</div>",
            scope : {
                url : '='
            },
            transclude : true,
            link : function(scope, element, attr){
                element.on('change', function(event){
                    scope.options = getOptionsFromURL(scope.url);
                })
            }

        }
    })
```

That's All For Today!!

- Routing using angular-ui-router
- Scope Life Cycle
- $resource
- Services/Factories/Providers
- Ng-include and $templateCache
- Custom Directives

SIVA SUBRAMANIAN

shiva1991@live.com