- What is Angularjs?

Angularjs is a structured framework for dynamic web Apps. It lets you use HTML as your template language & lets you extend HTML's syntax to express your application's components clearly.

- A complete client - side solution.
- great for Building CRUD operations.
  - Data - binding
  - basic - templating
  - directives
  - form validation
  - routing / Deep - linking
  - reusable components
  - dependency Injection

* Removes the following from Before -

- Removed callbacks
- manipulating HTML DOM programmatically.
- writing tons of initialization code just to get started.

Data binding , deep linking and dependency injection.

Use service like $http and $ routeProvider, modify content w/ filters. Add two- way data blinding , and work with routes and templates.

Features

Directives
MVC
Data binding
Expressions

Angularjs vs Angular

Different paradigm
MVC vs components
More stable
Less tooling

Working with modules
Module(), ng-app
Controller (), ng-controller

Angularjs services

$http
Runs with a server
Returns a promise.

Angular website : https :// Angularjs.org

Declare your app : ng-app

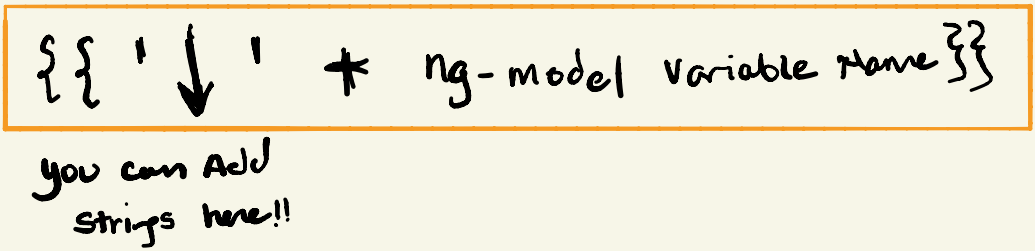
Directive is like command, and command or directives in Angularjs.

Always begin with the ng-prefix. They look like HTML attributes.

Directive: ng-model : creates a variables in your application, or your application scope, and it goes inside an form element. The variable can then be accessed within your scripts as well as within your template.

Two-way: data binding , in order to display data, we can use double curly braces in our HTML to create an expression.

Expression = {{ }} to show value of variables in HTML.



Module() is a container for different parts of your application. And inside A module,we can have different controllers.

Index.html

```
<HTML ng-app="myApp">

</HTML>

<body ng-controller="mycontroller">

</body>
```

App.js

```
Var app =
        Angular.module
                ('myApp');
```

```
Var myApp =
    Angular.module
        ('myApp', [] );
```

LIST OF dependencies.

App.js

```
myApp.controller('mycontroller', function mycontroller
                                    ($scope) {

    $scope.artist = "";

}
```

ng-src="" for any img source

---

ng-show

ng-hide

Both of these classes to show or Hide the element which means that element Stice exist in Dom.

ng-if

create element only IF the expression you give is true.

ng-repeat — loop through Array.

```
<UL>
  <li class="" ng-repeat="item in artists">
        <div>
          <h5> {{ item.Name }} </h5>
        </div>
  </li>
</UL>
```

- Angular Services: $HTTP

```
MyApp.controller('mycontroller', MyController($scope, $http){
    $HTTP.get('JS/data.JS').then(function(res)
    {   $scope.artists = res.data;
    });
```

Need to pass in dependency.

Basic Filters

- currency
- number
- date
- lowercase
- uppercase

* Use | and : characters

```
{{ Item.Name | UpperCase}}
```

- Array Filters

limit To: qty: start

Filter: keyword

order By : key: reverse.

- Deep Linking

ng Route

$ route provider

ng-view

```
myApp.config([ '$routeProvider', function ($routeProvider)
{
    $routeProvider.when('/', {
        templateUrl: 'JS/partials/search.tlpML',
        Controller: 'SearchController'
    }];
}]);
```

```
Var myApp = Angular.module('myApp',
                ['ngroute', 'mycontrollers']
```

→ Binding & directives

→ ng-App directives → Commond

{{ }} expression.

Need to Add Angular.js script tag in html tag.

Application Scope.

Modules and controllers

module is a container for different parts of your Applications, and inside a module, we can have different controllers.

Controllers ↙ ng-controller are piece of code that can handle some specific functionality within a module.

- A controller can also be defined with different dependencies.

- dependencies is just a name for something that controller needs in order to work.

One of the most common ways to initialize components is by using a special variable called scope.

scope → $scope

scope is a global object that we can use to communicate between js and html.

If we insert a variable in the scope, that means that we can use it within our html.

[ ] → list dependencies.

```
var myApp = angular.module('myApp', []);
myApp.controller('myController', function MyController
($scope) {
    $scope.artist = { json object }.
});
```

ng-src = img src

- ng-show will display an element if value of expression is true.
- ng-hide will hide an element.
- ng-if this will work similarly as v-if in vue.js, where it will not create element.
- ng-repeat : loop through items

In Angularjs, you can use services you can add your own or use a few of the ones provided by Angularjs.

- $http services let you handle Ajax Request to load files like json documents.
- request to a file doesn't normally stop the browser from doing other things

This why we call Ajax asynchronous.
Javascript uses something called promises.
which means that when request is ready
we can execute some code Afterwords.
- Returns a Promise.

```
$http.get(' ').then(function(response){
    $scope.artists= response.data;
});
});
```

We can split up our code into different
sections through something deep linking.

Now, this is the process of making different
URLs load up different content.

ng-route

Route library

src = "lib/angular/angular-route.min.js"

<div ng-view>

```
var mycontrollers = angular.module('mycontrollers', []);

mycontrollers.controller('SearchController', function($scope, $http){

$http.get('JS/data.json').then(function(response){
    $scope.artists= response.data;
    $scope.artistorder= 'name';
};
});
```

```
var myApp = angular.module('myApp', [
        'ngRoute',
        'myControllers'
]);
```

route parameter:

$routeparams as variable that is going to hold all information that comes from the route.

```
$scope.resetForce = function() {
    $scope.force = 0;
}
```

Whenever we define an ng-controller attribute on element, what happens behind the scenes is that Angular creates a new scope on top of that element, and it creates a controller as well & inject it with this new scope we just created.

```
<div ng-controller="playgroundController">
    <p>{{$scope.force}}</p>
    <div ng-controller="internalController">
        <p>{{$scope.force}}</p>
    </div>
</div>
```

So, internal controller has Access to parent controller scope.

But Playground controller Doesn't have access to internal controller $scope.

$scope is an object that we can extend & add attributes to

$ Vs. $$

- $scope.$parent
- $scope.$$watchers

Okay to use

$scope.$ Variables

one dollar sign variables are public and can be used & relied on in your Angular APP

- $scope.$parent
- $scope.$watch
- $scope.$on
- $scope.$root
- $scope.$apply

two dollar sign ($$) variables are not public (private), should not be used in our code.

- $scope.$$watchers
- $scope.$$postDigest
- $scope.$$phase
- $scope.$$destroyed

$scope inheritance work

* Prototype inheritance means when we try to access property on the son scope like, for example, scope dot force we will first check if property exists on the son scope.

So, here we can see the son scope, it has no attributes on it besides what is its prototype. which is the parent scope on the right side & the parent scope has force attribute with value of 30.

On other hand, when we're assigning something to the son scope, we do not alter the prototype. we just create a new variable on the son scope itself.

best pratice, we can use an object on the scope, & always access variables on top of that object. It is very common in Angular to use the controller itself for such an object, & bind controller to the scope.

The variable vm is the object that we're going to use in order to write is to bind this vm to our scope using :

$scope.playground = vm. ↖↘ same

```
angular.module('learndigest', []);

angular.module('learndigest').controller('playgroundCtrl',
{                                        function($scope)
    var vm = this;

      vm.force = 30;

    vm.resetForce = function () { vm.force = 0; }

});

angular.module('learndigest').controller('internalCtrl',
                                         function ($scope)
{
    var vm = this;
    vm.playground = $scope.$parent.vm;
    vm.secret = "shhh" + vm.playground.force;
});
```

There are some directives in Angular that create a new scope when we use them. for - example, ng- repeat or ng-if.

ng-if creates its own scope. when we change value, we only change that newly * created internal scope.

- We now understand that every scope has a parent scope. so, where do we stop. → root scope

When we create a new ng-app, root scope is generated. With it.

every scope is a descendant of root scope.

$watch. Method can be when you want to check if a certain values has changed & ran some code when it changes.

```
$scope.$watch('vm.force', function() {
        console.log('force value has changed', vm.force);
    });
});

$scope.$watch('vm.force', function(newValue, oldValue) {


});
```

```
$scope.$watch(function() { return vm.force % 2 === 0; },
                                    function(newValue, oldValue)
{
        console.log(newValue ? 'even' : 'odd')
})
});

$scope.$watch('vm.userProperties', function() {
        console.log("");
}, true);
});
```

↑ deep watch → watches every properties.

```
myApp.run(['$rootScope', '$location', function($rootScope, $location)
{
    $rootScope.$on('$routeChangeError',
                   function(event, next, previous, error)
    {
        if(error == 'Auth_Required')
        {
            $rootScope.message = 'Sorry ——— you';
            $location.path('/login');
        }
    });
});
```

- too many watchers make the Digest slow
- Behind scenes : Angular Adds $$hashkey : A to keep track of the object.

- Allow indicating which watchers should only be evaluated once.

```
<div ng-repeat = "item in $ctrl.list">
    <div > {{ ::item.name }} </div>
</div>
```

ng-include : directive includes HTML from an external file. The included content will included as childnodes of the specified element.

```
<element ng-include = "filename" onload = "expression"
                    autoscroll = "expression" >
</element>
```

File name : A file name, written with apostrophes, or an expression which Returns a file name.

onload : optional. An expression to evaluate when the included file is loaded.

autoscroll : optional: whether or not the included section should be able to scroll into a specific view.

ng-include vs component

ng-include

todo.html

{{todo.name}}

Initializes ^ takes 50%-60% longer.

100% longer digest cycles.

component

<todo todo="todo"></todo>

- Alot of faster.

<input type="text" ng-model="ctrl.show.name">

Watch vs ng-change

watch is slow.

ng-change is faster.

<input type="text" ng-model="$ctrl.show.name"
ng-change="$ctrl.nameChanged()">

ng-change : Benefits

- only triggers for changes made by the user.
- Expresses intent more clearly.
- Easier to trace
- more performant: saves on watches

Component:

```
angular.module('app').component('toggle', {
    templateUrl: '....',
    bindings: {
        callback: '&'
    },
    controller: function() {
        //.....
        this.callback();
    }
});
```

When using the ampersand binding, Angular doesn't just do two-way bindings the regular way, it takes the expression, in this case, mycallback Value, & saves it.

When called, Angular evaluates that expression on the parent scope. this is because, when using ampersand, it's possible to pass something other than function calls, even though that's what developers do, 99% of the time. It possible to pass an expression & not just a function & that would still work. Because Ampersand Binding is so generic

Password component

```
angular.module('app').component('passwordBox', {
    template:
        '<input type = password ng-model="ctrl.ngmodel"
            Placeholder ="{{ctrl.placeholder}}" />',
    bindings: {
        ngModel: '=',   <- two way binding (equal sign)
        Placeholder: '@'   special Binding using the @ sign
    }
});
```

Services And Factories.

Similar use cases.

Both create an injectable

Both are invoked once to create singleton.

Essentially, Factories are functions that return the object while services are constructor functions of the object.

```
Function MyCtrl( SomeService) {
    SomeService.someFunction();
}
```

Factory Implementation

```
angular.module('app').factory('SomeService', Function()
{
    return {
        someFunction: function() {}
    };
});
```

//@ - Text bind (getter)

//= - Two way bind.

//& one way bind

```
angular.module('app').service('someService', function()
{
    this.someFunction = function() {};

});
```

Service Function will be invoked once by Angular using the new keyword.

- ES5 - use factories.
- ES6 - use services

ng-resource : $save

$scope : $apply, $watch, etc.

- ng-resource is Rest API caller from UI to Backend. Similar to $http provider.

angular.toJson()

Ignores properties that start with "$$".

Is safer in Angular app.

$q.defer()

```
return $http.get('options.get').then(function(response){
        return response.data;
});
```

So, when is $q.defer() good?

- when integrating with non-Angular libraries that use callbacks.

HTTP Interceptors

- Angular's way for handling cross-app HTTP concerns.
- most commonly used for authentication & error handling.
- Angular provides the $injector service.
- The injector is the programmatic way to access Angular's dependency injection at runtime.

Angular.bootstrap

```
app.component('modal', {
  template: [
    '<div ng-transclude="title"></div>',
    '<div ng-transclude="body"></div>'
  ].join('');
  transclude: {
    title: 'modalTitle',
    body: 'modalBody'
  },
  controller: function () {
    // stuff to make this component render
       as a modal.
  }
});
```

linky : Filter Turn URLs in pagraphs to safe links

Add angular-santize.js to the project

Require the 'ngSanitize' module

```
<div ng-bind-html = "blog.post | linky"></div>

<div ng-bind-html="blog.post | linky:'_blank':{rel:
                                    'nofollow'}"></div>
```

$location

- The $location is very useful service in Angularjs to read or change url in the browser. It use "window.location" in back & make URL available to our Application.

- It also synchronizes the URL with browser when user click on back or forward button, change URL address bar manually or click any Language.

- **url** - read and write
- var currentURL = $location.url();
  - if current path in address bar is :

    "http://abc.com/#test/location/url"

    - output : test/location/url

- **absUrl** - read only. returns the full path with all segments.
  - If the current path is :

    "http://localhost:42382/TestAngularJS.html"

    - output ; "http://localhost:42382/TestAngularJS.html"

Port - read only method. returns the port Id of current url.

http://localhost:42382/TestAngularJs.html

output: 42382.

Protocol - read only method.

Returns the current protocol.

Path - Read & Write method. It takes optional Parameter.

- http://abc.com/#test/location/path?test=new
  - output - "test/location/path"

```
<script>
angular.module('initExample', [])
  .controller('ExampleController', ['$scope', function($scope) {
    $scope.list = [['a', 'b'], ['c', 'd']];
  }]);
</script>

<div ng-controller="ExampleController">
  <div ng-repeat="innerList in list" ng-init="outerIndex=$index">
    <div ng-repeat="value in innerList" ng-init="innerIndex=$index">
      <span class="example-init">
        list[ {{outerIndex}} ][ {{innerIndex}} ] = {{value}}
      </span>
    </div>
  </div>
</div>
```

$q.defer() as deferred

$q is a built-in service which helps in executing asynchronous functions, & use their return values (or exception) when they are finished with processing.

$q is integrated with the $rootscope.scope, $scope model observation mechanism, which means faster propagation of resolution or rejection into your models & avoiding unnecessary browser repaints, which would result in flickering UI.

To use promises, we need to inject $q as dependency

Var defer = $q.defer(); // creates a new instance of deferred.

deferred object exposes a promise as well as the associated method to resolve that promise.

It is constructed using $q.defer() function & it exposes three main methods. resolve(), reject(), & notify().

- resolve(value) – this method resolves the derived promise with value.

- reject(reason) – this method rejects the promise with some message or reason

- notify(value) - this method provides updates on the states of the promise. It can be called any number of time before the promise is either successfully resolved or rejected with any reason.

- when deferred object is created, a new promise instance is also created & that can be accessed by deferred.promise promise object helps to access the result of deferred task, when it is finished.

In an AngularJs directive the scope allows you to access the data in the attributes of the element to which the directive is applied

```
<div my-customer name="customer xyz"></div>

angular.module('myModule', []).directive('myCustomer', function() {
    return {
        restrict: 'E',
        scope: {
            customerName: '@name'
        },
        controllerAs: 'vm',
        bindToController: true,
        controller: ['$http', function($http) {

            var vm = this;

            vm.doStuff = function(name) {

                console.log(vm.custorName);
            };
        }];
        link: function(scope, element, attrs) {

                Console.log(scope.customerName);
        }
    };
});
```

- directives can come in two forms
- either in Attributes.
- either as elements. <customer> </customer>

- <random-ninja> - hypen

  syntax for directive.

  or camel case. - randomNinja - inside directive names.

restrict : A , E , C , M , EA

- A - Attribute only
- E - element only
- C - can use it both ways

Scope : { } inside directive. directive has its own scope.

Isolated scope.

<random-ninja ninjas = "ninjas" > </random-ninja>

ninjas - Prop

$ Compile function of AngularJS

@ - you need curly braces.

= - you don't need curly braces.

func : '&' ← references the controller's function from directive.

```
myNinjaApp.directive('randomNinja', [function(){
    function linkFunction(scope, elem, attrs){
        elem.bind('click', function(){
            console.log(elem[0].innerHTML);
        });
    }
    return {
        restrict: 'A' // A, E, C, M, EA
        scope: {
            ninjas: '=',   // binding one value to another.
            title: '='
        },
        template: '<img ng-src="{{ ninjas[random].thumb }}">',
        templateUrl: 'views/random.html'
        link: linkFunction,
        controller: function($scope){
            $scope.random = Math.Floor(Math.random() * 4);
        },
    }
}]);
```

Prop name

Some thing