ANGULARJS: Javascript (built on vanilla js, compact, easy to do stuff)

1. Dom manipulation diff from application logic
2. Separation of concerns (MVC pattern)
3. Make single page application (asp.net, php)
4. Robust, enterprise-scale js client-side applications
5. Additional components, directives, plugins can be added)

All application logic in one module. If second argument is provided, we are creating a new module. Empty square bracket means we have no dependency. No second argument means we are accessing existing module.

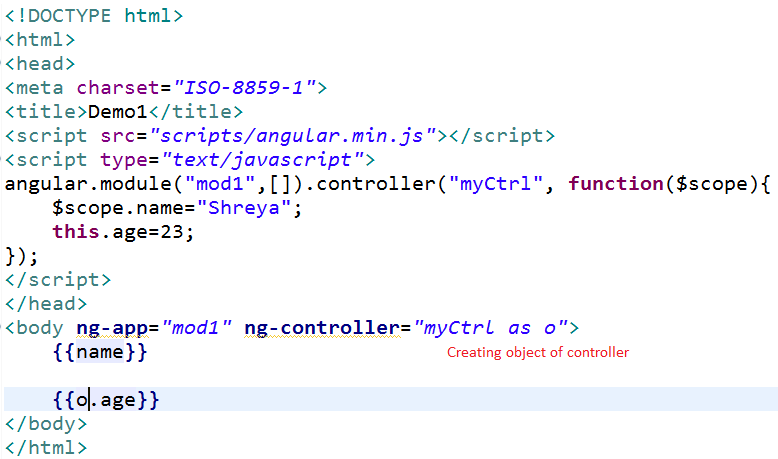
Ng-app: angular application, root where angular starts executing.

Usually one ng-app per webpage, can have more than 1 also. Can be attached to any html element. Ng-app attached to root module.

Angular expression: javascript code/ expression. Cannot have conditional statement or loops.

Angular module created. Template: html with angular mark-up. Ng-\* attributes. Evaluating expressions, data-binding markup. Compiles template (for errors), loads template in memory, transform template with data. Renders to dom tree.

**Controllers:**  Contains application logic. Usually part of angular module. Allows us to send/receive data b/w views and application logic. Attached to dom using ng-controller attribute. Controller works as a class. Mostly work with web services to pull data from database and then populate view based on it.



.run: method executed after module is created.

Open source by google, mvc architecture, front-end application

3) It is a Framework. Functions, classes, modules, dependencies defined in library. Framework will take care of building your app. Jquery: library

EmpName

Designation

Submit

**Using jquery:**

$(‘empName’).val(): val method will fetch details

$(‘designation’).val()

**Using javascript:**

Document.getElementByID(“empName”);

Document.getElementByID(“designation”);

**Library**: has methods which can be used

**Framework**: container to develop application

**Features**

1. Client side MVC (model: data to be displayed, view: display the data, controller: set up the data)
2. Two way data binding ( from view automatically update model object)
3. Dependency injection (dependencies in angular : to call web-service, some services provided by angular ($http, $location), such services injected in code)
4. Routing (In single page app development, home link opens a new page, services opens new page, contact opens other page. All links displayed on same page)
5. Directives (Some pre-defined directives (base functionality like h1 means heading in html), like ng-app, ng-controller, ng-bind, ng-model , ng-class as html does not know how to fetch data from controller)
6. Filters (To filter the data)
7. Controllers (to control the flow of application, between view and model, fetch data from db and display on webpage and data sent to db)
8. Services ($http)
9. Testability (both unit, integration)
10. Build and deployment (can be deployed on server too)

Database

Rest endpoint

Json

Angular application

Data cannot be fetched directly from db, so uses rest services to fetch the data from db.

1. Angular simplifies application development
2. Not every app is a good fit for angular
3. Built using CRUD in mind (data-binding, templating, directives, form validations, routing, deep-linking, reusable components, dependency injection)
4. Best suited for single page applications

Cannot be used for:

1. Games, GUI editors with tricky dom manipulations
2. Diff from crud

Core concepts:

1. Template: html with additional markup (html page)
2. Directives: extend html with custom attributes and elements
3. Module: container for diff parts of app including controller, directive, services, filters
4. Scope: context where model is stored so that controllers, directives, expressions can access it. Diff scope for accessing data.
5. Model: data shown to user in view
6. Expression: access variable and functions from scope
7. View: what user sees (template)
8. Data binding: sync between model and view
9. Controller: business logic behind view (intermediate between model and view)

Include angular library to project

**Angular.module:** function in ng module. Used to create , register and retrieve angular modules.

Angular.module(name, [requires dependencies], [configFn]);

When passed 2 or more params, new module is created. If only one argument is passed, existing module is retrieved.

Var app= angular.module(“mod1”,[]); app is instance of module mod1.

Var app1= angular.module(“mod1”,[‘ngResource’]); inject ngResource not part of angular, but needs to be used in angular application, so can be injected. Create mod1 injecting ngResource.

This mod1 can be used in other modules then

**Template :** Written in html that contains angular specific elements and attributes.

**Directive**: attribute , element or class that augments an existing dom element or represents a reusable dom component

**Ng-app:** root element of application. Placed near root element of page eg: body or html

<html ng-app=”mod1”>

**Scope**: $scope service used to put data inside the scope.

App.controller(“ctrl1”, function($scope){});

$scope is a service which we are injecting in the app controller. Scope is created automatically when compiled html template is executed. Scope is an object which refers to application model.

Nginit directive allows you to evaluate an expresiion in current scope.

Eg:

<body ng-app ng-init="label1='text'">

<input type="text" ng-model="label1" placeholder="Enter text"/>

<input type="text" ng-model="style1" placeholder="Enter style"/>

<button style="{{style1}}">{{label1}}</button>

{{style1}}

</body>

**Controller:** Data can be created within the controller using scope also which is the ideal way to do.

Controller is a javascript constructor function and is attached to dom via ng-controller directive. A new child scope will be available as injectable parameter to controller’s constructor function as $scope.

Changing value in controller displayed on view: 1 way data-binding: Done using {{}}

Changing value in view, updates the model i.e. backend data: **2 way**. Done using ng-model

**Array :**

<body ng-app ng-init="emps=['Ram','shyam']">

<ol>

<li ng-repeat="emp in emps">{{emp}}</li>

</ol>

</body>

Eg2:

<script type="text/javascript">

angular.module("mod1",[]).controller("ctrl1", function($scope){

$scope.emps=['Ram','shyam'];

});

</script>

</head>

<body ng-app="mod1" ng-controller="ctrl1">

<ol>

<li ng-repeat="emp in emps">{{emp}}</li>

</ol>

</body>

**JSON object**

<script type="text/javascript">

angular.module("mod1",[]).controller("ctrl1", function($scope){

$scope.emp={name:'Ram',id:1};

});

</script>

</head>

<body ng-app="mod1" ng-controller="ctrl1">

Welcome {{emp.name}}

</body>

**Array of json**

<script type="text/javascript">

angular.module("mod1",[]).controller("ctrl1", function($scope){

$scope.emp=[{name:'Ram',id:1},{name:'Raj',id:2}];

});

</script>

</head>

<body ng-app="mod1" ng-controller="ctrl1">

<ol>

<li ng-repeat="e in emp">

{{e.name}}

</li>

</ol>

</body>

**Working with external file**

Get json data from external file like mydata.dat or employee.json, use $http service. All predefined services starts with $. Injected in controller to fetch data.

Success and error are two callback methods.

The response object has these properties:

•data – {string|Object} – The response body transformed with the transform functions.

•status – {number} – HTTP status code of the response.

•headers – {function([headerName])} – Header getter function.

•config – {Object} – The configuration object that was used to generate the request.

•statusText – {string} – HTTP status text of the response.

**Eg:File has array**

<script type="text/javascript">

angular.module("mod1",[]).controller("ctrl1", function($scope, $http){

$http.get("./data/emp.dat").success(function(result){

console.log(result);

$scope.emp= result;

}).error(function(errMsg){

console.log(errMsg);

});

});

</script>

</head>

<body ng-app="mod1" ng-controller="ctrl1">

<ol>

<li ng-repeat="e in emp">

{{e.name}}

</li>

</ol>

</body>

**Eg2: File has json object**

<script type="text/javascript">

angular.module("myModule",[]).controller("myController", function($scope, $http){

$http.get("http://localhost:8080/AngularJSDemos/data/employee.json").then(function success(response){

$scope.employees= response.data.records;

});

})

</script>

</head>

<body ng-controller="myController">

<ul>

<li ng-repeat="employee in employees">{{employee.name + "-" + employee.marks}}</li>

</ul>

</body>

**Sample app:**

<script type="text/javascript">

angular.module("mod1",[]).controller("ctrl1", function($scope, $http){

var url="http://localhost:8081/AngularJSDemos/data/";

$scope.userId;

$scope.user={};

$scope.get= function(){

$http.get(url).success(function(result){

$scope.users= result;

}).error(function(errMsg){

console.log(errMsg);

});

}

$scope.getUser= function(){

$http.get(url+ $scope.userId).success(function(result){

$scope.user= result;

}).error(function(errMsg){

console.log(errMsg);

});

}

$scope.saveUser= function(){

$http.post(url, $scope.user).success(function(result){

$scope.successMsgForSave= result;

}).error(function(errMsg){

console.log(errMsg);

});

}

$scope.updateUser= function(){

$http.put(url+$scope.user.id, $scope.user).success(function(result){

$scope.updateData= result;

}).error(function(errMsg){

console.log(errMsg);

});

}

$scope.deleteUser= function(){

$http.delete(url+$scope.user.id).success(function(result){

$scope.successMsg= "Deleted record for "+ $scope.user.id;

}).error(function(errMsg){

console.log(errMsg);

});

}

});

</script>

</head>

<body ng-app="mod1" ng-controller="ctrl1">

Get users:

<input type="button" ng-click="get()"/>

<ol>

<li ng-repeat="e in users">

{{e.name}}

</li>

</ol>

</body>

**Minified version:**

Uglifyjs yourjsfile.js –o newfilename.js

**Ng-click:** To add event click on button. To call function inside the controller.

**Ng-show and ng-hide:**  to display or hide something on webpage

**Ng-class:** to dynamically set css on html element by data binding an expression that represents all classes to be added.

<style>

.c1{

color:red;

}

</style>

</head>

<body ng-app ng-init="users=[{ 'name': 'Rm' },{ 'name' : 'sm' }]">

Get users:

<input type="text" ng-model="val"/>

<ol>

<li ng-repeat="e in users" ng-class='val'>

{{e.name}}

</li>

</ol>

</body>

**Script, ng-template, Ng-include:**

Script directive: Load the content of the script element into the template cache and a cache name for the template must be assigned through element id. Type of script must be specified as text/ng-template.

Eg1:

<script type="text/ng-template" id="test.html">

<h1>This is the inline template</h1>

</script>

</head>

<body ng-app>

<div ng-include="'test.html'"></div>

Hello

</body>

Eg2:

var url="http://public-api.wordpress.com/rest/v1/sites/wtmpeachtest.wordpress.com/posts?callback=JSON\_CALLBACK";

var responsePromise = $http.jsonp( url).success(function(json) {

alert(json.found);

$scope.homeMessage= json.found;

});

library: angular.js as cdn or locally

1. Module: angular.module("sample",[]);//empty array means creating a module. If not providing, implies retrieving the module defined somewhere else.

2. <body ng-app="sample">

ng-app: angular application. Root where application starts. 1 ng-app per webpage. ng-app cannot be nested. Can be attached to any html element (body/ html tag)

3.Angular expressions:

{{2+3}}

control flow statements, loops not alowed.

4.<script src="scripts/angular.min.js"></script>

<script type="text/javascript">

angular.module("mod1",[]).controller("controller1",function($scope){

$scope.name="Kartik";

});

</script>

</head>

<body ng-app="mod1" ng-controller="controller1">

{{2+3}}

{{["Ram","Shyam"][1]}}

{{name}}

</body>

Body has ng-app attribute. Anything starting with ng is angular directive. Every template is compiled by html compiler provided by angularjs. Then loaded into the browser memory and transforms template with data. (data binding, evaluate expressions). Then we get DOM out of it. Browser understands dom. Called View

2.<script type="text/javascript">

angular.module("mod1", []).controller("controller1",function(){

this.name="Kartik";

});

</script>

</head>

<body ng-app="mod1" ng-controller="controller1 as o">

{{2+3}}

{{["Ram","Shyam"][1]}}

{{o.name}}

</body>

Controller allow us to send and receive data b/w the views and controllers. Controller attached to dom element using ng-controller attribute/ directive

Controller(class) instantiated using "controller1 as o"

Controller creates model and sends it to the view. Controller works closely with rest, gets data and places it in view. View works closely with controller. Not with web services.The memory area created for controller instance, logical context created called scope.

When script is minified, don't want to change $scope, so give it in string. replaces c at runtime.

<script type="text/javascript">

angular.module("mod1", []).controller("controller1",["$scope",function(c){

c.name="Kartik";

}]);

</script>

</head>

<body ng-app="mod1" ng-controller="controller1">

{{name}}

</body>

angular js adds classes ng-scope and ng-binding automaticallt with ng-controller,

<script src="scripts/angular.min.js"></script>

<script type="text/javascript">

angular.module("mod1", []).controller("controller1",function($scope){

$scope.age=10;

$scope.getAge= function(){

return this.age \* 20;

}

});

</script>

</head>

<body ng-app="mod1" ng-controller="controller1 as o">

{{getAge()}}

</body>

o added to scope object

$scope.o= this;

eg:

<script type="text/javascript">

angular.module("mod1", []).controller("controller1",function($scope){

$scope.o = this;

this.age=10;

this.getAge= function(){

return this.age \* 20;

};

});

</script>

</head>

<body ng-app="mod1" ng-controller="controller1 as o">

{{o.age}}

</body>

Run is invoked after services initialized and $injector is created.

Binding:

Unit tests: karma

Integration testing: protractor

Npm test: start karma unit test runner. Needs karma.conf.js as the configuration file for karma

Npm run protractor: run protractor end to end tests. Uses selenium standalone server and jdk

Npm run update-webdriver: installs driver needed by protractor. To cancel, Ctrl+c. Instead Delete node\_modules and run npm install again

Npm start: run the development web server

Ng-app means angular directive with the name ngApp. Angularjs evaluates the expression and inserts result in the dom and it is done in the context of the current model scope, not in the global context

Bootstrap angular application: using ng-app or creating injector which will be used for dependency injection

AngularJS service is a singleton object created by service factory and service factories are functions created by service provider. Service providers are constructor functions. When instantiated must have $get property which holds the service factory function.

When we request for a service, $injector is responsible for finding correct service provider , instantiating it and then calling $get service factory method to get the instance if the service. Provider is a constructor function with a $get property. $provide is a service with additional helper methods to register services without specifying the provider.

1. Provider: naming convention is name of the service it provides followed by the Provider.

Eg: $log service has $logProvider. We can configure how service will act and what kind of service is created. Eg. $logProvider has a method debugEnabled which specifies whether $log service will debug messages to console or not. If provider is an object: $get method invoked using $injector.invoke() methods when an instance needs to be created. If it is a constructor: New instance of provider is created using $injector.instantiate().

1. Constant
2. Value
3. Factory
4. Service
5. decorator

**Controller:**

Controller definition:

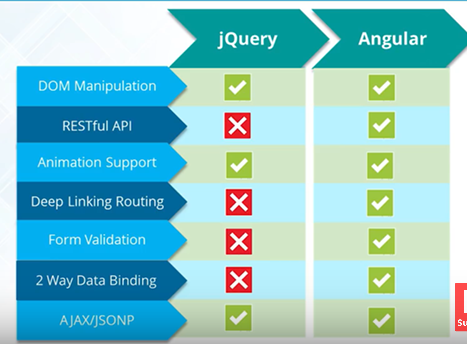
Mod.controller(“ctrl1”, function(){

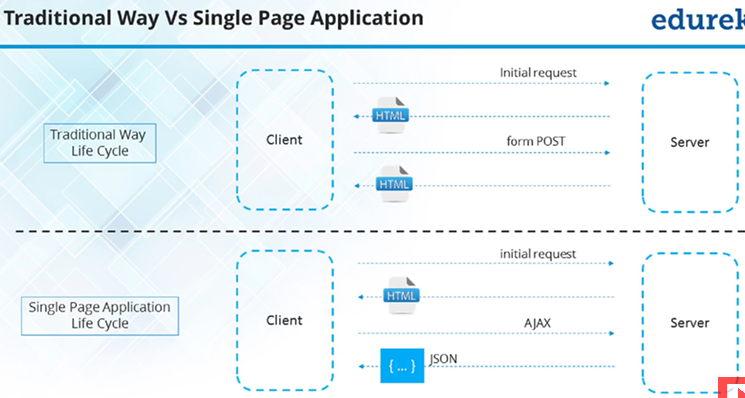
});

<div ng-controller=”ctrl1”>

// Context of this controller. Memory assigned. Context, logical assigned in dom memory and assigned to div element. This context is called scope. Controller instantiated when ng-controller used. View also has access to scope.

</div>



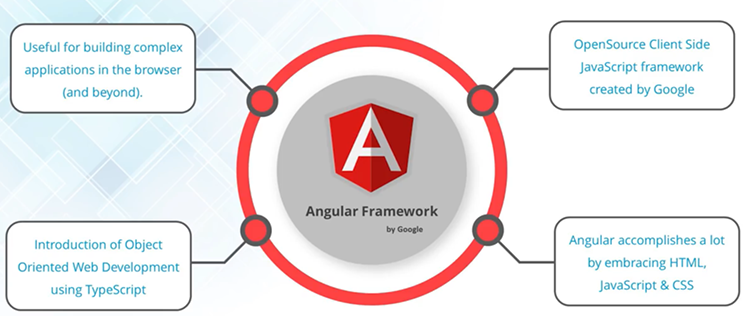


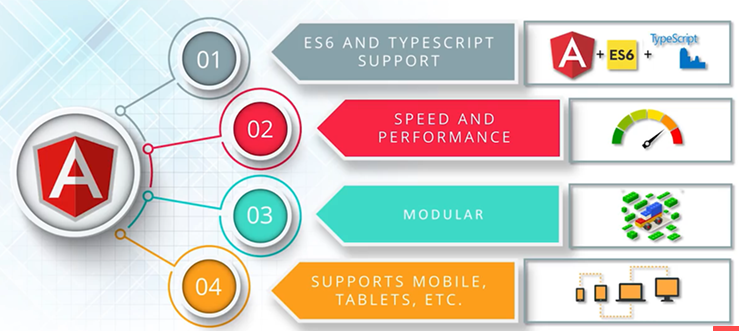
**Installation**

**Use Node.js instead of cmd**

1. npm install –g @angular/cli
2. ng new my-app
3. cd my-app
4. ng serve : compile and start the dev env, open up the browser, refresh changes, recompile.
5. First install typescript (npm install typescript –g)
6. Tsc –v : version for typescript

Basic data types of javascript, null, undefined, array, array manipulation, object, object manipulation. Typescript : superset of javascript. Valid js is valid ts. Can Transpile ts into js. And can run on web browser. Typescript: can work on types, interfaces, classes, decorators, import/export or modularity.





Data type in typescript:

1. **Boolean**: let isDone : Boolean = false;
2. **Number:** 6, 0xf00d, 0b101011, 0o744, decimal
3. **String:** let color: string = “blue”;

Use ` if span multiple lines.

let sentence : string = ` Hello, my name is ${name}.

I am 10 years old .`;

1. **Array:** let list: number[] = [1,2,3];

Or let list1: Array<number> =[3,4,5];

1. **Enum:** enum color {Red, Green, Orange}

Let c: color = color.Red;

**Modularity:**

Every file is a module. To work with objects, you have to import that object from that file.

Import {object} from ‘./file.ts’;

Import \* from ‘./file.ts’

Var x= ‘test’

If we want to use x in other file, we need to export it, else no one can use it. So each file is independent

Can create a class:

Class MyClass{

Private X: string= ‘Test variable’; // x = ‘Test’

Public Y: number[] = [2,3];

Protected z: Point;

constructor(){

}

func(){

}

onSelect(){}

}

interface Point{

X: string;

Y: number; }

Class tester extends Test{

Constructor(){

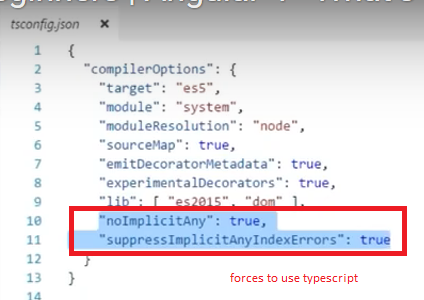
Super();

}

}

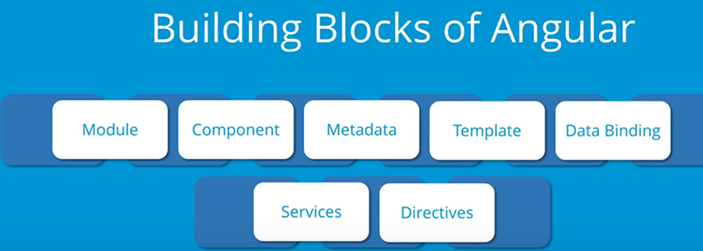
**Folder Structure:**

1. .angular-cli.json: Has metadata for cli to manage, run, test and build this project
2. Karma-conf.js: for testing
3. Protractor-conf.js: end to end testing
4. Tsconfig.js: which version of js it should compile to, what modularity it should use.



1. Package.json: name of the project, dependencies: needed for production, devDependencies: not needed for production but for dev
2. Index.html: <app-root>: Component that we have created and this is core of angular.
3. Main.ts: get the application bootstrapped. Imported module and bootstrapped it
4. App.module.ts: declare components, modules, services etc, bootstrapping component

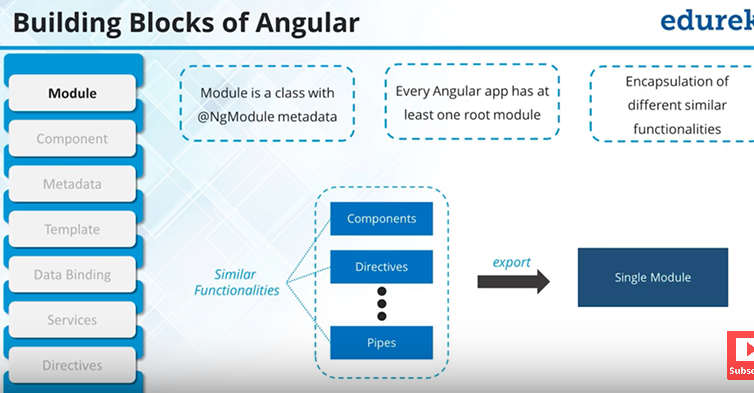
Modules have components

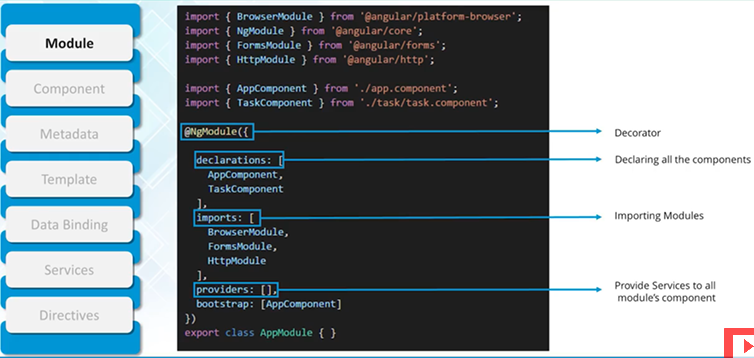


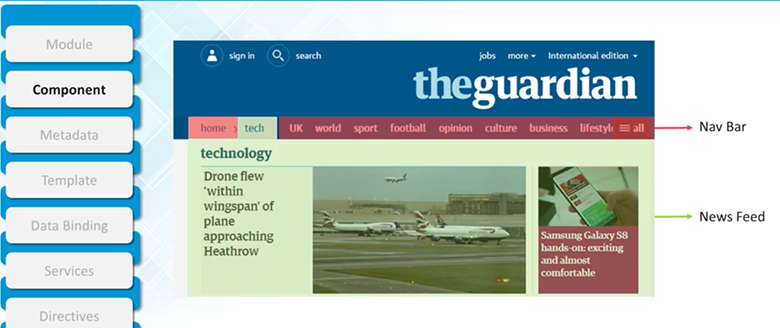
Dat-binding: updating views, ..

services: dependency , can be injected as needed

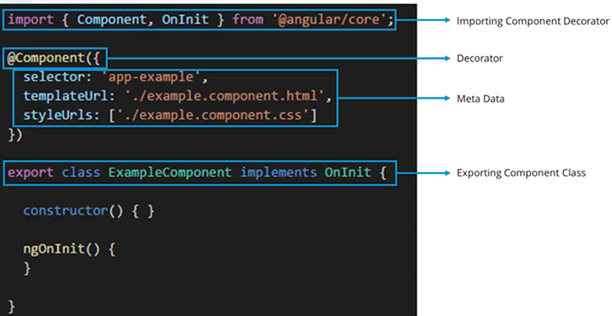
Angular module: Not file based but combination of components, services, pipes, directives to create one single module.





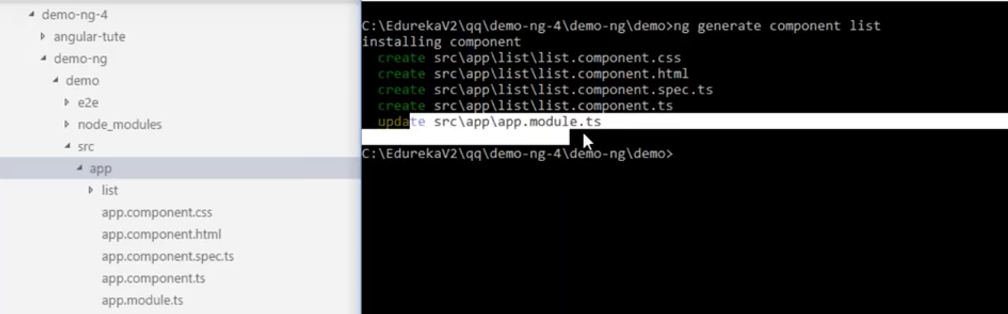


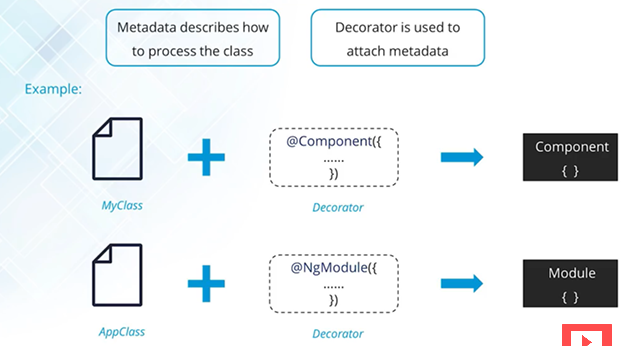
Component: various sections like nav bar, header, footer, tech component. Created as a class

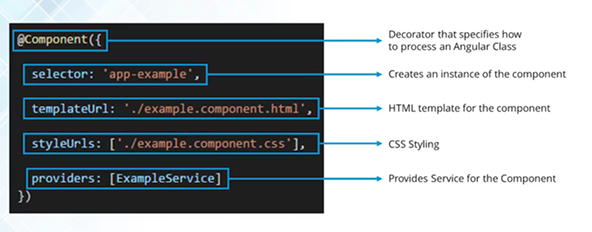


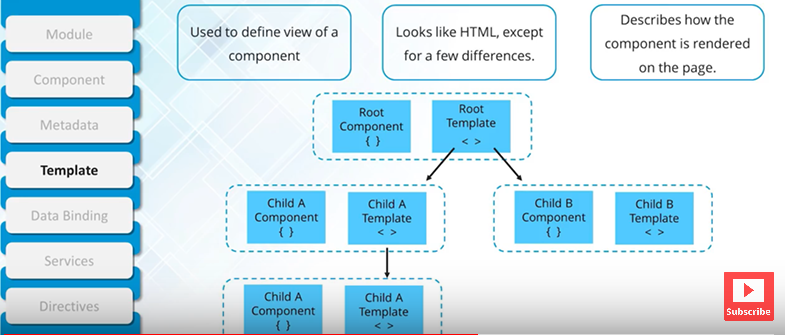
No semi-colon after @Component class else will break app.

1. Ng generate component list // A component list would be created inside app folder.

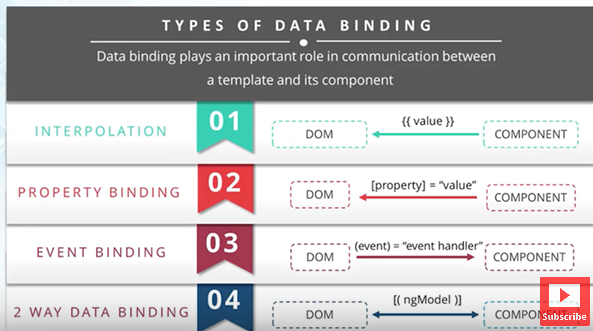








Application is based on components and like component tree. Components are isolated. Input and output methods to work with parent child and services to work along with siblings.



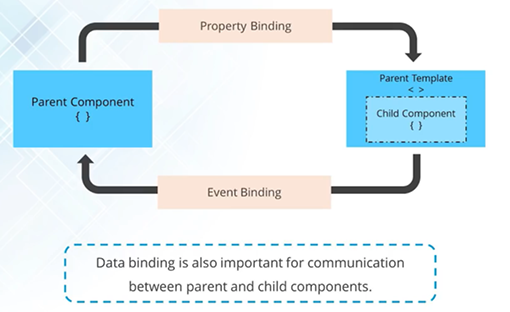
**Example of property binding**

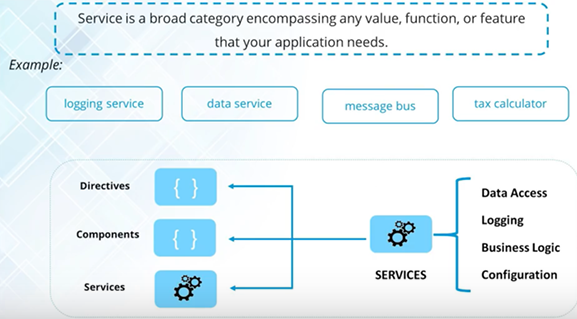
In ts file:

Let str: string = ‘apple’;

In html file:

<img [src]=’str’>



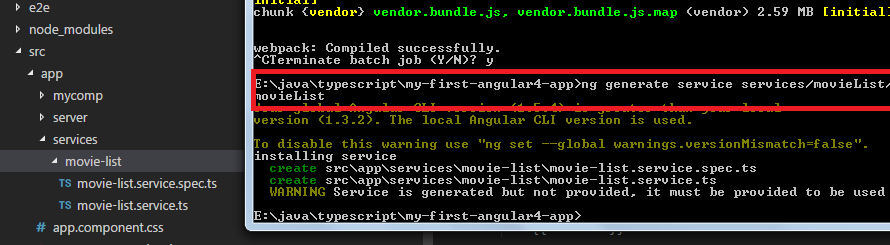


Generally created as singleton. (@Injectable makes it a singleton) Termed as dependency injection.

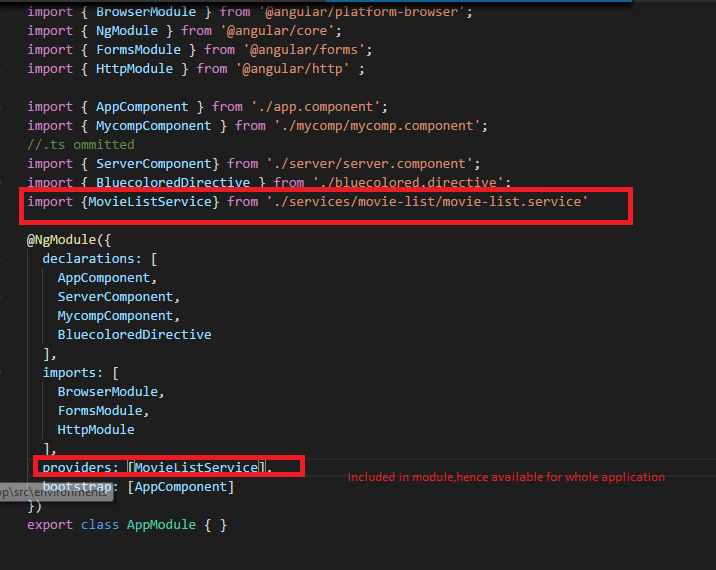
Inject service to service.

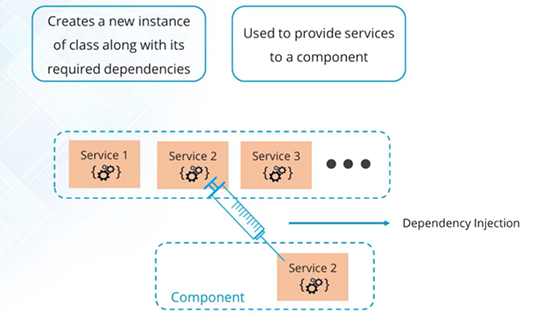


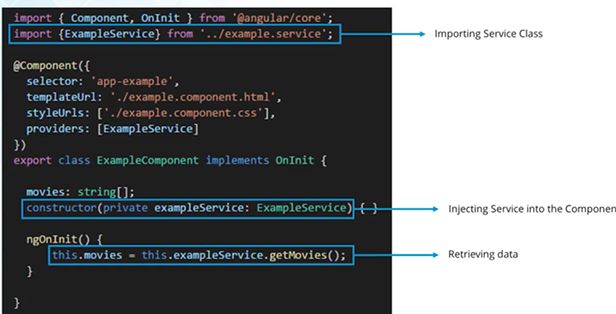
Ng generate service services/moviesList/movielist

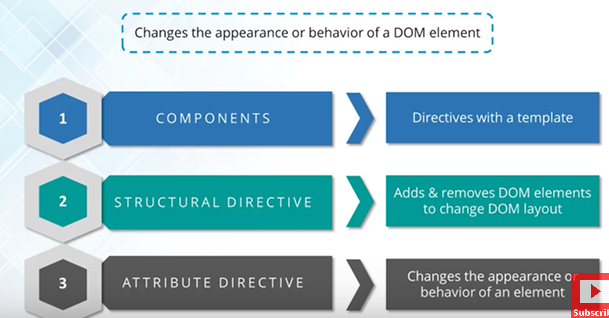


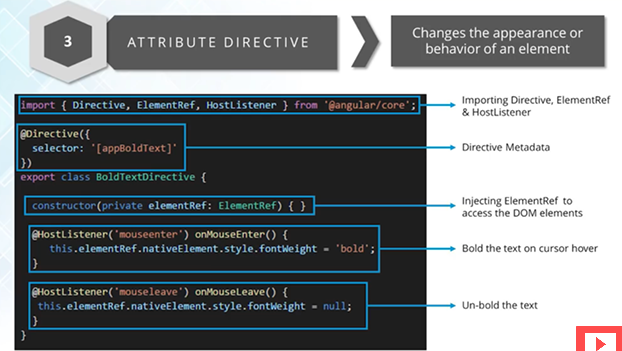
Service created. Module or component can provide the service. Module level means available for whole application

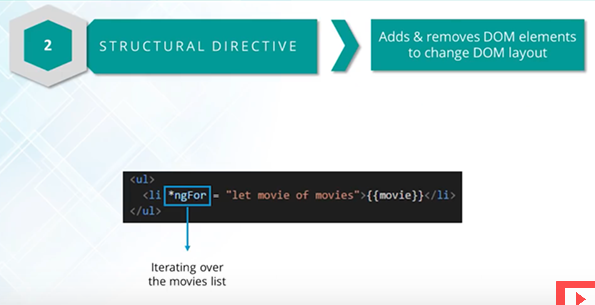












Ng generate directive bluecolored

HostListener: will listen to

elementRef: get reference to element where directive is assigned,

