**What is angular2?**

Angualr2 is an open source javascript framework to build web applications in html and js and has been conceived as a mobile first approach.

**Why to use angular2?**

* Angular 2 is simpler than angular 1 and its fewer concepts make it easier to understand.
* You can update the large data sets with minimal memory overhead
* It will speed up the initial load through server side rendering

Features.

* Angular 2 is faster and easier than angular1
* It supports latest the version of browser and also supports old browsers including ie9 and android 4.1
* Angular 2 is mainly focused on mobile apps.
* It is a cross platform framework
* Code structure is very simplified than the previous of angular

**Advantages:**

* if an application is a heavy load then angular2 keeps it fully ui responsive
* it uses server side rendering for fast views on mobile.
* It works well with ECMAScript and other languages that compile to js
* It used dependency injection to maintain applications without writing too long code
* Everything will be the component based approach

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Disadvantages:**

* Since angular 2 is a newly introduced framework there is less online community support

Angular2 used typescript which is a primary language for developing of angular applications

The typescript is a super set of javascript which is migrated to typescript and code written in typescript make less prone to run time errors.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Module:**

The module component is characterized by a block of code which can be used to perform a single task. You can export the value of sometimes from the code such as a class.

The angular apps are called as modules and build your application using many modules.

Export class AppComponent{ }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Component:**

A component is a controller class with a template which mainly deals with a view of the application and logic on the page. It is a bit code that can be used throughout an application.

You can associate css styles using component inline styles style url and template inline styles to a component.

@Component({  
 **selector**: **'my-app'**, *//custom component* **templateUrl**: **'app/ts/app.component.html'**,  
 **directives**:[PlaylistCompoenet]  
  
 *//<h1>{{}mainHeading}</h1> one way data binding*})

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Template:** The components view can be defined by using the template which tells angular how to disply the component for instance below simple template shows how to display the name.

<div>

Name is:{{name}}

</div>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Metadata:**

Metadata is a way of processing the class. Consider we have one component called mycomponent which will be a class until we tell angular that it’s a component. You can use metadata to the class to tell angular that my component is a component. The metadata can be attached to ts by using the decorator

@Component({

selector : 'mylist',

template : '<h2>Name is Harry</h2>'

directives : [MyComponentDetails]

})

export class ListComponent{...}

@component --🡪component

Selector, template and directives are metadata in angualrjs

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Data binding:**

Data binding is a process of coordinating application data values by declaring bindings between sources and target html elements. It combines the template parts with components parts and template html is bound with markup to connect bothe sides. There are four types of data binding:

* Interpolation: It displays the component value within the div tages.
* Property binding: it passes the property from the parent to property of the child.
* Event binding: It fires when you click on the components method name.
* Two-way binding: this form binds property and event by using the ngModel directive in a single notation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Service:**

Services are js functions that are responsible for doing a specific task only. Angular services are injected using dependency injection mechanism. Services includes the value function or feature which is required by the application.

* Logging service, data services, message service, the configuration of application

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Directives:** The directive is a class that represents the metadata. There are three tyes of directives:

* Component directive: It creates custom controller by using view and controller and used as custom html element
* Decorator directive: it decorates the elements using additional behavior.
* Template directive: It converts html into a reusable template

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dependency Injection is a design pattern that passes an object as dependencies in different components across the application. It creates new a instance of class along with its required dependencies.

You must remember the below points while using Dependency Injection:

* The Dependency Injection is stimulated into the framework and can be used everywhere.
* The *injector* mechanism maintains service instance and can be created using a *provider*.
* The *provider* is a way of creating a service.
* You can register the *providers* along with injectors.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In component :

1.@NgModule: @ is decorator

2.declatations: 🡪 component names

3.imports 🡪 we imporated some modules we insert it

4. providers same as injectors

5.bootstrap: tell us which one is top level component for our application

Component creations:

**import** { Component } **from '@angular/core'**;  
  
@Component({  
 **selector**: **'app-root'**,  
 **templateUrl**: **'./app.component.html'**,  
 **styleUrls**: [**'./app.component.css'**]  
})  
**export class** AppComponent {  
 **title** = **'Rental'**;  
}

ng g component <name> to create new component

OnInit🡪 ebility to give something this components

OnInit to ngOnInit

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
property binding

-------------------------------------------------

Features of angular 2

**Components:** the earlier version of angular had a focus of controllers but now has changed the focus

To having components over controllers. Components help to build the applications into many modules. This helps in better maintaining the applications over a period of time.

**Typescript:** the newer version of angular is based on ts. This is a superset of js and is maintained by Microsoft

**Services:** Services are a set of code that can be shared by different components of an applications. So for example if you had a data component that picked data from a database you could have it as a shared service that could be used across multiple applications.

In addition angular2 has better event-handling capabilities, powerful templates and better support for --mobile devices.

Components of angularjs2:

**Modules**- this is used to break up the application into logical pieces of code. Each piece of code or module is designed to perform a single task.

**Component** – this can be used to bring the modules together.

Templates – this is used to define the views of an angular application.

Metadata- this can be used toad more data to an angularjs class.

Service: this is used to create components which can be shared across the entire applications

Npm: this is known as the node package manage that is used to work with the open source repositories. Angularjs as a framework had dependencies on other components. And npm can be used to download these dependencies and attach them to you project.

Each application consists of components. Each component is a logical boundary of functionality for the application. You need to have layered services, which are used to share the functionality across components.

Component

Component

Component

Services

Following is the anatomy of a component a component consists of:

Class: this is like a c or java class which consists of properties and methods

Metadata- this is used to decorate the class and extend the functionality of the class

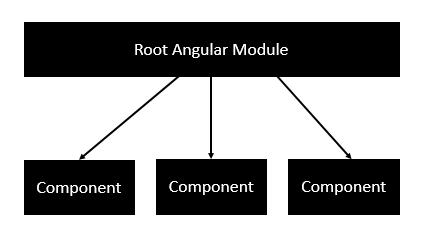
Template- this is used to define the html view which is displayed in the application

Component

Metadata

class

Template



Following is an example of a root module.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

Angular2- directives:

A directive is a custom html element that is used to extend the power of html angular2 had following directives that get called as part of the browser module

1. Ngif
2. Ngfor

NgIf

The ngif element is used to add elements to the html code if it evaluates to true else it will not add the elelments to the html code

\*ngIf = 'expression'

If the expression evaluates to true then the corresponding gets added else the elements are not added.

ngFor

the ngfor element is used to elements based on the condition of the for loop.

\*ngFor = 'let variable of variablelist'

The variable is a temporaty variable to display the values in the variablist

import { Component } from '@angular/core';

@Component({

selector: 'demo-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

appTitle: string = 'Welcome';

appList: any[] = [ {

"ID": "1",

"Name" : "One"

},

{

"ID": "2",

"Name" : "Two"

} ];

}

<div \*ngFor = 'let lst of appList'>

<ul>

<li>{{lst.ID}}</li>

<li>{{lst.Name}}</li>

</ul>

</div>

Metadata:

Metadata is used to decorate a class so that it can configure the expected behavior of the class.

**Annotations** − These are decorators at the class level. This is an array and an example having both the @Component and @Routes decorator.

Following is a sample code, which is present in the app.component.ts file.

@Component({

selector: 'demo-app',

templateUrl: 'app/app.component.html'

})

Data binding:

Two-way binding was a functionality in Angular JS, but has been removed from Angular 2.x onwards. But now, since the event of classes in Angular 2, we can bind to properties in AngularJS class.

Suppose if you had a class with a class name, a property which had a type and value.

export class className {

property: propertytype = value;

}

You could then bind the property of an html tag to the property of the class.

<html tag htmlproperty = 'property'>

Angular2- cli

Command line interface can be used to create our angularjs application it also help in creating a unit and ene to end tests for the applications.

Angular dependency injection

Dependency injection is the ability to add the functionality of components at runtime.

Create a separate class which had the injectable decorator the injectable decorator allows the functionality of this class to be injected and used in any angularjs module

@Injectable()

export class classname {

}

@Component ({

providers : [classname]

})

Angular2 allows you to work with any third-party controls.

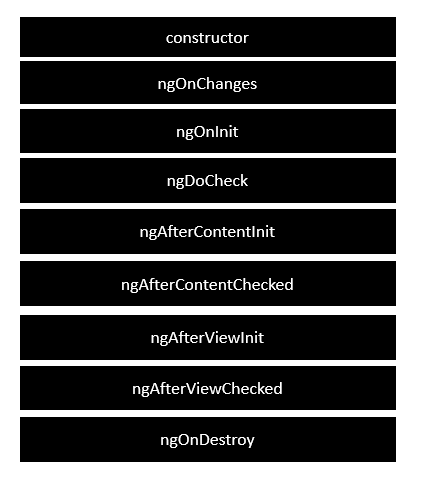
Install the component using the npm command

Npm install ng2-pagination –save

Angualrjs2 pipes

* **'Pipename'** − This is the name of the pipe.
* **Pipeclass** − This is name of the class assigned to the custom pipe.
* **Transform** − This is the function to work with the pipe.
* **Parameters** − This are the parameters which are passed to the pipe.
* **Returntype** − This is the return type of the pipe.

Angularjs lifecycle hooks:



ngOnChanges: when the value for data bound property changes then this method is called,

ngOnInt - this is called whenveer the initialization of the directive after angular first displays the data-bound properties happens

* **ngDoCheck** − This is for the detection and to act on changes that Angular can't or won't detect on its own.
* **ngAfterContentInit** − This is called in response after Angular projects external content into the component's view.
* **ngAfterContentChecked** − This is called in response after Angular checks the content projected into the component.
* **ngAfterViewInit** − This is called in response after Angular initializes the component's views and child views.
* **ngAfterViewChecked** − This is called in response after Angular checks the component's views and child views.
* **ngOnDestroy** − This is the cleanup phase just before Angular destroys the directive/component.

Kudvenkat notes

Advantages:

1,Improved modularity, testability, laudability,

1. With angular2 we can build a single application that works across mobile and desktop devices
2. Component based development: in angular everything is a component component are the building blocks of an angular application

ES 6 new features: classes modules arrow functions etc

Type script is superset of JavaScript

Typescript benefits:

Intellisence

Autocompletion

Code navigation

Advanced refactoring

Strong typing

Supports es6 features like class, interfaces

What is component in angular

A component in angular is a class with a template and a decorator composed of template, class, decorator

Template defines the user interface. Contains the html, directives and data bindings

Class contains the code required for the template

Decorator adds meta data to the class making it an angular component

Angular interpolation : moving component property to html property

1.one way data-binding –from component to view template

2.one way data bindng – from view template to component

3.two way data binding – from component to view template from view template to component

{{ title ? title : ‘no need’ }}

If we have title then display title property or else it will display no need

<img src=”{{image url}}”/>

Property binding

<img [src]=”image url”/>

Interpolation and property binding flows uni directional

Angular data binding sanitized malicious content before displaying it

\*ngIf – is called structural directive

-------------------------------------------------------------------------------------------------------------------------------------

Two-way data binding:

<**input type="text" [value]="name" (input)="name=*$event*.target.value"**/>

ngFor directive

ngFor directive may perform poorly with large lists.

A small change to the list may trigger a cascade of DOM manipulation

Steps to create custom pipe

Step1: create the custom pipe

Step2: declare the custom pipe in the module where we need it

Step3: use the custom pipe just like any other built in angular pipe