**Angular 2**

**Features of Angular 2**

Following are the key 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 application over a period of time.
* **TypeScript** − The newer version of Angular is based on TypeScript. This is a superset of JavaScript and is maintained by Microsoft.
* **Services** − Services are a set of code that can be shared by different components of an application. 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.

## Components of Angular 2

Angular 2 has the following components −

* **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 JS application.
* **Metadata** − This can be used to add more data to an Angular JS class.
* **Service** − This is used to create components which can be shared across the entire application.

**Why Angular 2**

Performance: Angular 2 is 5 times faster compared to AngularJS 1

Mobile Support: With Angular 2 we can build a single application that works across mobile and desktop devices.

Component Based Development: In Angular 2, everything is a component ,component are the building blocks of an Angular application

More language choices:

* ECMAScript 5
* ECMAScript 6 (Also called ES 2015)
* TypeScript
* Dart
* PureScript
* ELM, etc

Angular 2 itself, is built using TypeScript. TypeScript has great support of ECMAScript 6 standard.

**What is ECMAScript**

* The JavaScript language standard is officially called ECMAScript.
* ECMAScript 1 till ECMAScript 7 were released over the past several years.
* Most modern browser available today supports ECMAScript 5.
* The browser support for ECMAScript 6 is still incomplete
* Transpilation compiles ECMAScript 6 to ECMAScript 5.
* ECMAScript 6 is officially known as ECMAScript 2015.
* New features in ECMAScript 2015 – Classes, Modules, Arrow functions etc.

**What is TypeScript**

* Free and open source programming language developed by Microsoft
* Superset of JavaScript
* Transpilation Compiles TypeScript to JavaScript

**TypeScript Benefits**

* Intellisense
* Autocompletion
* Code navigation
* Advance refactoring
* Strong Typing
* Supports ES 2015 (ES 6) features like classes, interfaces & inheritance

TypeScript is supported by several code editors

* Visual Studio
* Visual Studio code
* Eclispe
* WebStorm
* Atom
* Sublime Text etc.

**Setting up Angular 2**

Step1: Install Node.js and npm

* Node version 4.6.x or greater
* Npm 3.x.x or greater
* Node -v
* Npm -v

**Angular 2 Components**

What is Component in Angular2

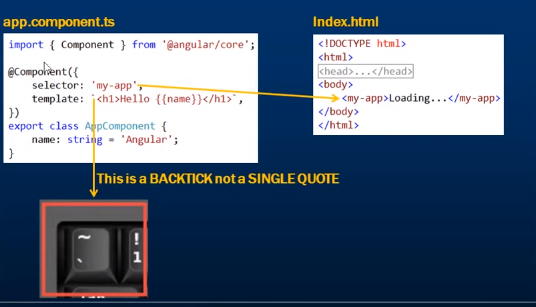
A component in Angular is a class with a template and a decorator

* Template
* Class
* Decorator

Template defines the user interface. Contain the HTML, Directive and data binding.

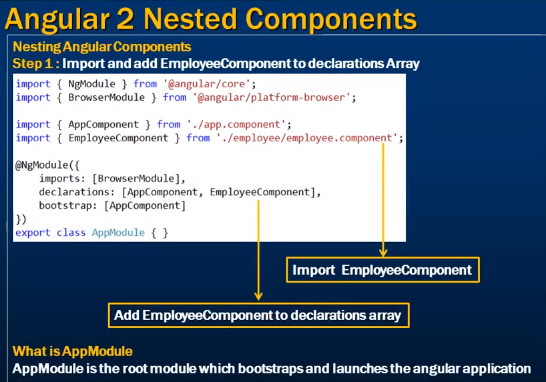
Class contains the code required for the template

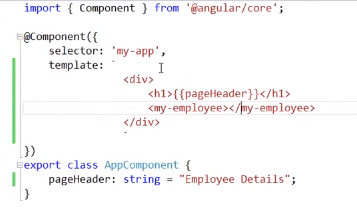
Decorator adds meta data to the class making it an Angular Component



**Component decorator template and templateUrl properties**

**Angular2 Nested Components**





**Different options available to apply styles to Angular Components**

* Styles in external stylesheets: styles.css
* Styles inline in the component HTML file
* Styles in the component html file using <style> tag
* Specify the styles using the @component decorator styles property
* Specify the styles using the @Component decorator styleUrls property

**Interpolation in Angular**

|  |  |
| --- | --- |
| **Data-Binding** | **Description** |
| One-way data-binding | From Component to View Template |
| One-way data-binding | From View template to Component |
| Two-way data-binding | From component to view template &from view template to component |

{{}} -Interpolation / Template Expression

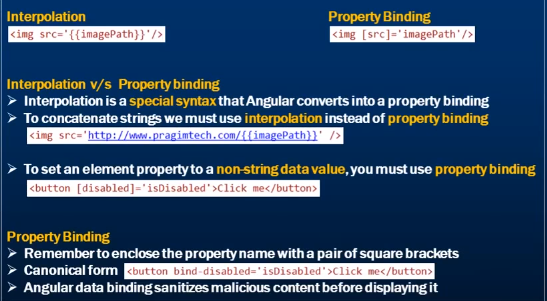
<h1>{{title}} </h1>

Template Expression with ternary operator

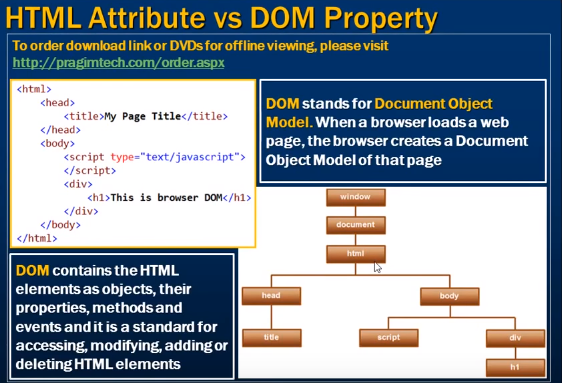
<h1> {{title? title: ’No title’}} </h1>

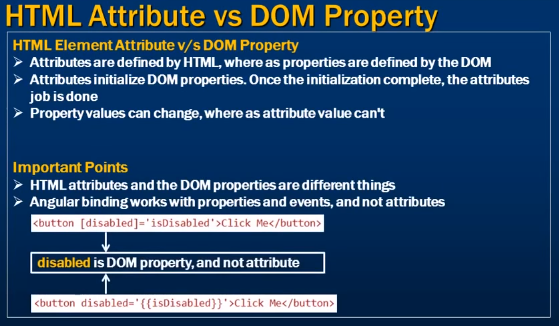
Interpolation is a one-way data movement from component to view.

**Property Binding in Angular [] / bind -**



**HTML Attribute vs DOM Property**

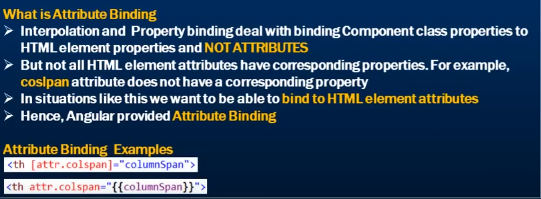




**Angular Attribute Binding**

Attr.colspan=”{{abc}}”

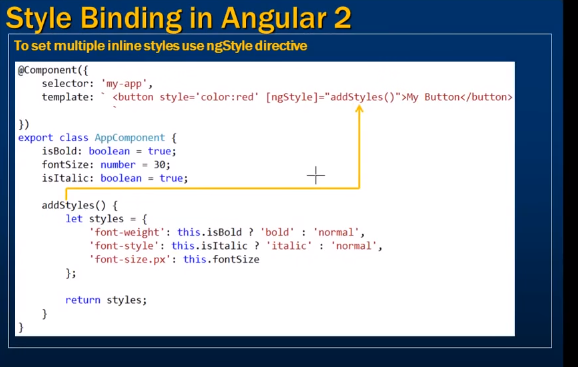
[Attr.colspan]=”abc”



**CSS class Binding**



**Style Binding in Angular**



**Angular 2 Events Binding**

These binding flow data in one direction i.e. from a component class property to an HTML element property

* Interpolation
* Property binding
* Attribute binding
* Class binding
* Style Binding

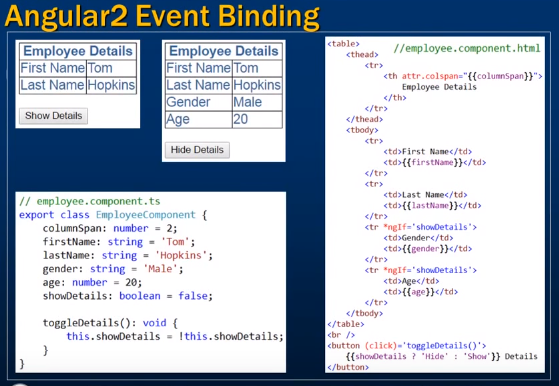
Event binding flows data in the opposite direction i.e. from an HTML element to a component.

Syntax for event binding:

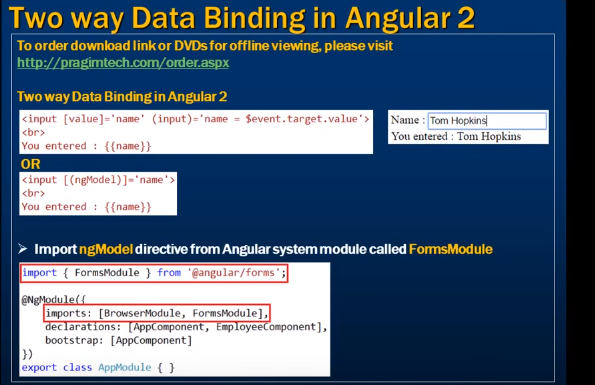
<button (click)=” onClick ()”>Click me</button>

*Canonical form*

<button on-click=” onClick ()”>Click me</button>



**Two-way Data Binding in Angular 2**



**Angular ngFor Directive**

Syntax:

<tr \*ngFor=”let employee of employees”>

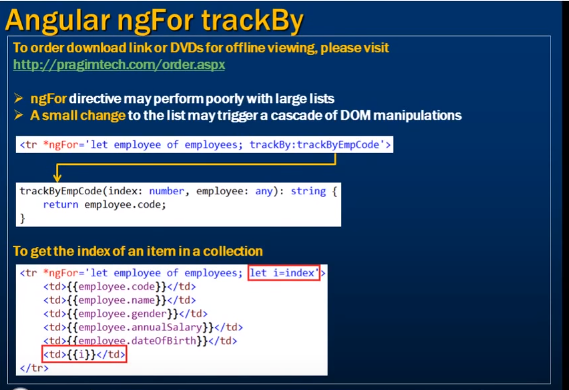
// employee is template input varaible

<td>employee.xxxxx</td>

<td>employee.xxxxx</td>

</tr>

**Angular ngFor trackBy**



**Bootstrap in Angular2**

* npm install [bootstrap@3.3.7](mailto:bootstrap@3.3.7) jquery –save

Configure. angular-cli.json

"styles": [

"styles.css",

"../node\_modules/bootstrap/dist/css/bootstrap.min.css"

],

"scripts": [

"../node\_modules/jquery/dist/jquery.min.js",

"../node\_modules/bootstrap/dist/js/bootstrap.min.js"

],

# **Angular pipes {{ | }}**

Pipes transform data before display. Pipe character “|”

Built in pipes include Uppercase, lowercase, date, currency,decimal , percent etc.

{{emp.name | uppercase}}

We can also chain pipes

<td> {{emp.DOB | date | uppercase}}

Pass parameter to pipe using colon “:”



**Angular Custom Pipes**

* Create type script file for custom file



<td>{{emp.name | employeeTitle:employee.gender}}<td>

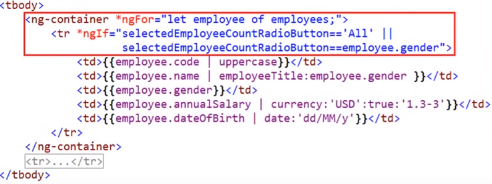
**Angular Container and Nested Component**

**Angular component input properties @input**

**Angular component output properties @output**

**Create custom event -EventEmitter**

**Ng-container**

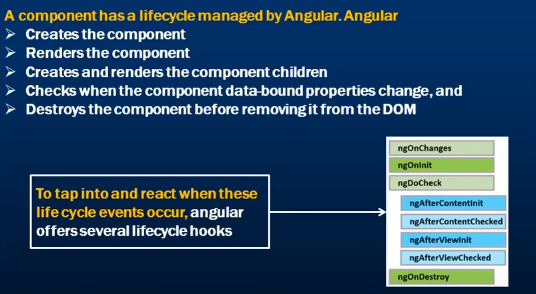


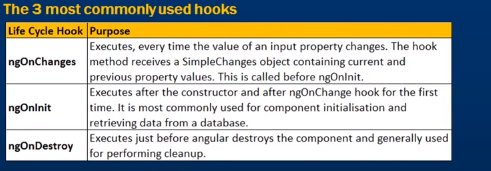
**Interface in angular 2**

Interface are the abstract type. its only content declaration of property, methods and events. Class implements the interface. Interface define contract that is shape or structure of API yet to implemented.

TypeScript is strongly typed language.

**Angular Component Lifecycle Hooks**





**Angular Services**

A service in angular is generally used when you need to reuse data or logic across multiple component.

Constructor is used to perform dependency injection for services.

ngOnInit is best place to call service and component initialization.

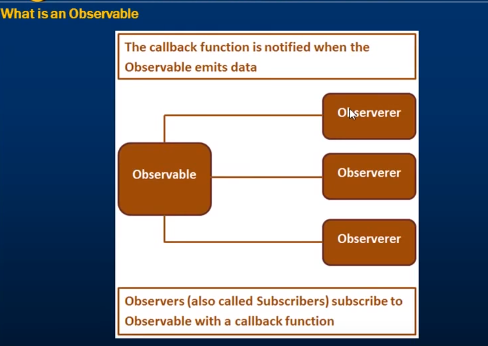
**Angular 2 http service**

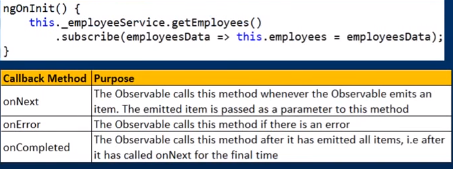
import { HttpClientModule } from '@angular/common/http';

imports: [

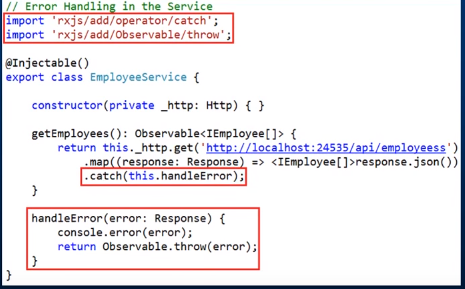
BrowserModule, HttpClientModule,FormsModule

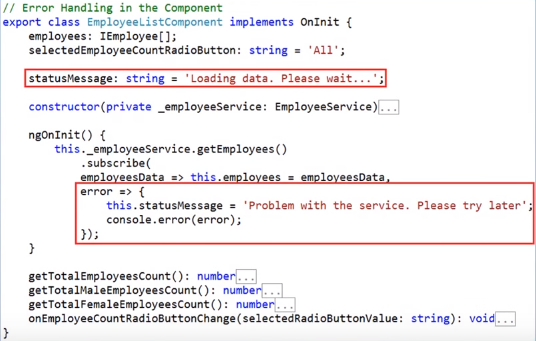
],





**Angular 2 http error handling**

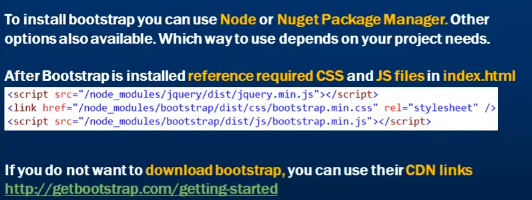




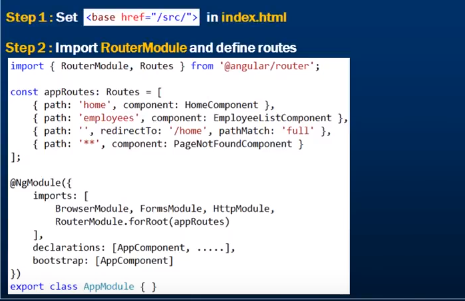
**Using Bootstrap with Angular 2**

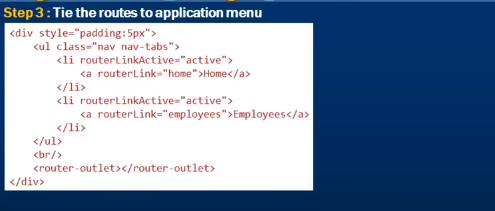
Using node package manager.

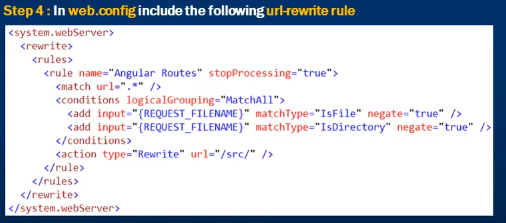
* + Open node.js cmd
  + npm install bootstrap@3 jquery –save
  + package.json file should be updated with dependency
  + add required file into index.html



**Angular 2 routing**



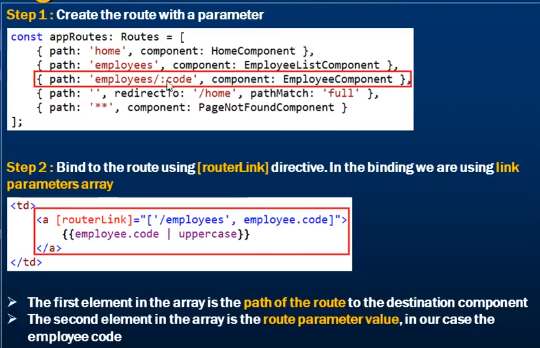




**Hash Style Routing**

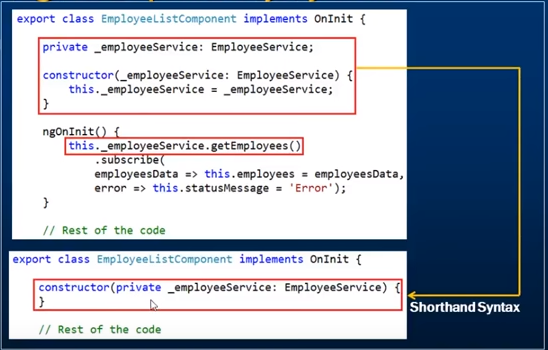


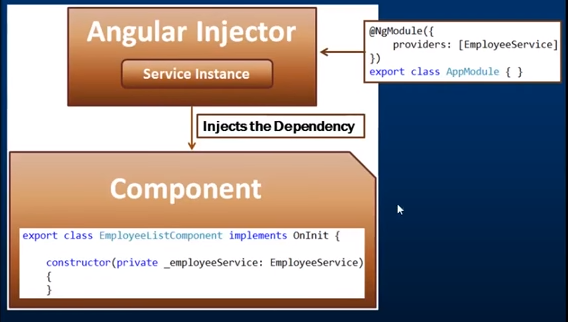
**Angular 2 route parameters**



**Angular dependency injection**

Angular Injector creates instance of service and provide its access to class/component.



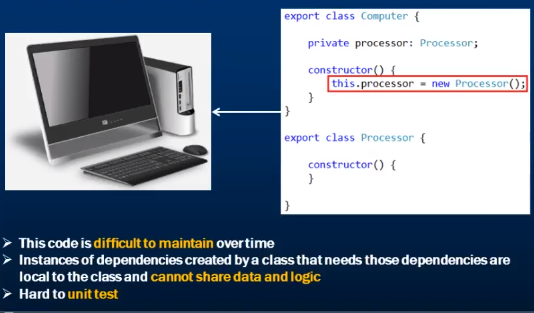


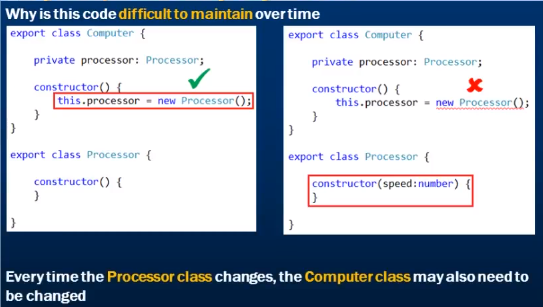
Constructor specify dependency on any service so angular injector creates instance of service. before that we should register this service to angular injector otherwise it will throw run time exception i.e. No providers for service

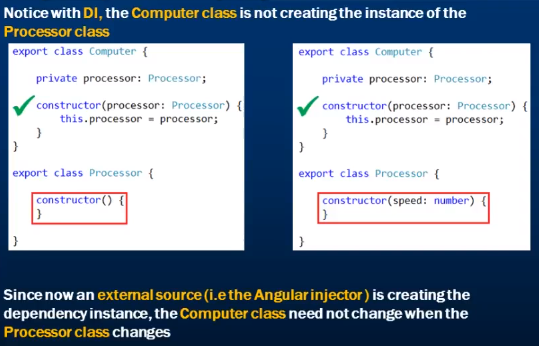
It’s a coding pattern in which a class receives its dependencies from an external source rather than creating them itself.

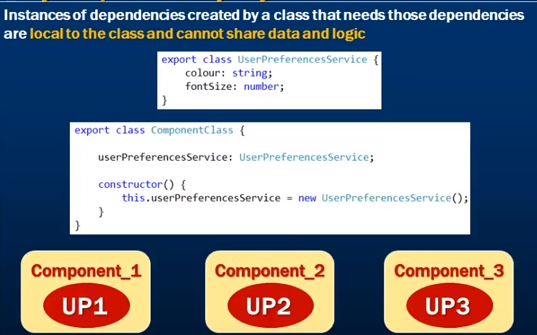
**Why dependency injection**

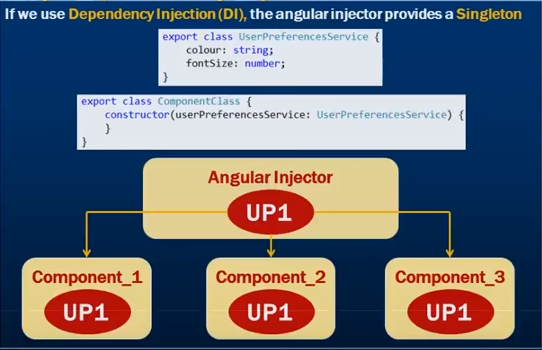
Without dependency injection

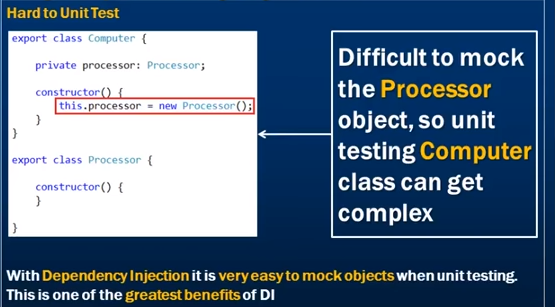


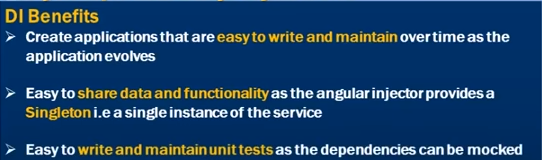










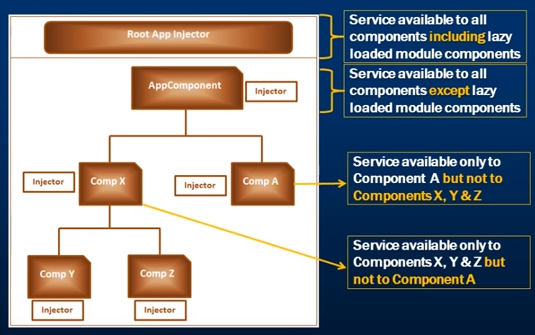


**Angular Singleton Service**

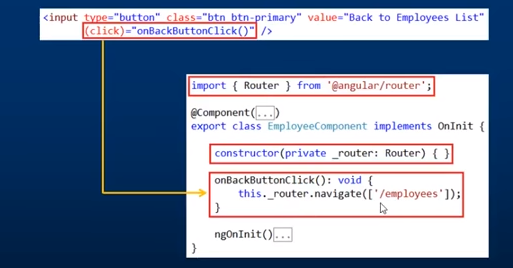
Angular injector creates single instance of any service and its share among the all component.

**Angular Injector**

**Angular root injector**



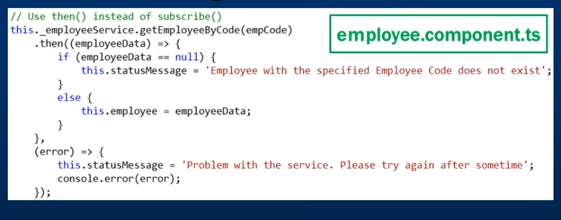
**Angular router navigate method**



**Promises in angular 2**

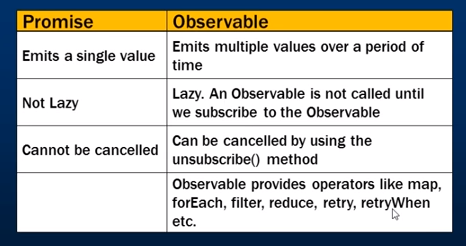
Promise execution is asynchronous, which means that it's executed, but the program won't wait until it's finished to continue with the rest of the code.





To deal with asynchronous data we use observable or promises.

**Angular promises vs observables**



Promises are most commonly used to handle HTTP requests. In this model, you make a request and then wait for a single response. You can be sure that there won’t be multiple responses to the same request.

#### Single value vs multiple values

const numberObservable = new Observable((observer) => {

observer.next(5);

observer.next(10);

});

numberObservable.subscribe(value => console.log(value));

// prints 5 and 10

const numberPromise = new Promise((resolve) => {

resolve(5);

resolve(10);

});

numberPromise.then(value => console.log(value));

// still prints only 5

const numberPromise = new Promise((resolve) => {

resolve(5);

});

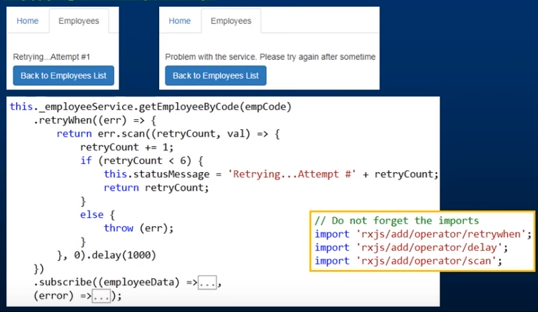
numberPromise.then(value => console.log(value));

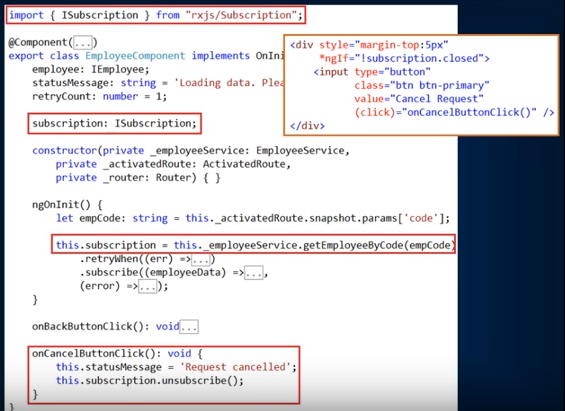
// will simply print 5

**Observable retry on error**

How to re-subscribe and retry an Observable if there is an error.

When connection issue with service occurs, we can re-subscribe using retry,



**Angular observable unsubscribe** 

**Difference between AngularJS, Angular 2 and Angular 4**

|  |  |
| --- | --- |
| **Version** | **Year** |
| AngularJS | 2010 |
| Angular 2 | 2016 |
| Angular 4 | 2017 |

