```
/**
 * test.component
 */
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-test',
  // templateUrl: './test.component.html',
  template:`
  <h2>welcome {{name}}</h2>
  <input [id]="myId" type="text" value="notChangingAttribute">
  <input [disabled]="isDisabled" id="{{myId}}" type="text"
value="notChangingAttribute">
  <h2 class="text-success">{{name}}</h2>
  <h2 class="text-special" [class]="successClass">{{name}}</h2>
  <h2 class="text-success">{{name}}</h2>
  <h2 [class.text-success]="hasError">{{name}}</h2>
  <h2 [ngClass] = "messagesClasses">{{name}}</h2>

  <h2 [style.color] = "hasError? 'red': green">StyleBinding1</h2>
  <h2 [style.color] = "highlightColor">StyleBinding2</h2>
  <h2 [ngStyle] = "titleStyles"> StyleBinding3 </h2>
  <h2></h2>

  <input #myInput type="text" value="input value">
  <button (click) = clicked($event)>click event</button>
  <button (click) = logMsg(myInput.value)>show input value</button>
  <button (click) = logMsg(myInput)>show input stub</button>

  <h2 *ngIf = "displayName; else elseBlock">
  codeEvolution
  </h2>
  <ng-template #elseBlock>
  <h2>Name is hidden</h2>
  </ng-template>

  <div *ngIf = "displayName; then thenBlock; else elseBlock"></div>

  </h2>
  <ng-template #thenBlock>
  <h2>codeEvolution</h2>
  </ng-template>

  <ng-template #elseBlock>
  <h2>Name is hidden</h2>
  </ng-template>

  <div [ngSwitch]="color">
    <div *ngSwitchCase ="'red'">You picked red color</div>
```

```
    <div *ngSwitchCase ="'blue'">You picked blue color</div>
    <div *ngSwitchCase ="'black'">You picked black color</div>
    <div *ngSwitchDefault>picke up a color again</div>
  </div>

  <div *ngFor="let color of colors; index as i">
    <h2>{{i}} {{color}}</h2>
  </div>

  <!-- f is true or false -->
  <div *ngFor="let color of colors; first as f">
  <h2>{{f}} {{color}}</h2>
  </div>

  <!-- l is true or false : we could use odd/even -->
  <div *ngFor="let color of colors; last as l">
  <h2>{{f}} {{color}}</h2>
  </div>

  `,
  // styleUrls: ['./test.component.css']
  styles: [`
      .text-success {
        color:green;
      }

      .text-danger{
        color:red;
      }

      .text-special{
        font-style:italic
      }
  `]
})
export class TestComponent implements OnInit {

  //[id]="myId": syntax not limited only to string only (includes
true/false,...), while 'id="{{myId}}"' is limited to string only...
  // <input [id]="myId" type="text" value="notChangingAttribute">

  //<h2 class="text-special" [class]="successClass">{{name}}</h2>
  // [class]="successClass" overrides the other because it's a property (not
an attribute)

  public name="codeevolution";
  public myId="testId";
  public isDisabled = false;
  public successClass="text-success";
```

```
  public hasError = false;
  public isSpecial=true;
  public messagesClasses = {
    "text-success": !this.hasError,
    'text-danger' :this.hasError,
    'text-special':this.isSpecial
  }

  public highlightColor="orange";
  public titleStyles = {
    color:"blue",
    fontStyle:"italic"
  };
  public displayName = false;
  public colors=["red","blue","yellow","greeb"];

  clicked(event){
    console.log(event);
  }
  logMsg(valObj:any){
    console.log(valObj);
  }

  constructor() { }

  ngOnInit() {
  }

}
```

```
/*
  Sending Data between Parent and child component
*/

<!--app.component.html-->
<div style="text-align:center">
  <h1>{{message}}</h1>
  <app-test (childEventProperty)="message=$event" [parentData]="name"></app-
test>
</div>

/*
  app.component.ts file
 */
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  // template:`<h1>hello<h1>
  // <div>I'm here</div>
  // <div>I'm here</div>`,
 templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
  message='';
  public name="Kejeiri";
}
```

```
/*
* test.component.ts File
*/
import { Component, OnInit,Input,EventEmitter,Output } from '@angular/core';
// import { EventEmitter } from 'events';

@Component({
  selector: 'app-test',
  // templateUrl: './test.component.html',
  template:`
    <h2>Hello {{parentData}} </h2>
    <h2>Hello {{name}}</h2>
    <button (click)=fireChildEvent($event)>fire Child Event</button>
  `,
  styleUrls: ['./test.component.css']
})
export class TestComponent {

  // the 1st statement:
  @Input() public parentData:string;

  //Alias name overrides the 1st statement (parentData): parentData becomes
empty
  @Input('parentData') public name;


  //Child sending data to the parent through event: creates an evt emmitter
(import  evt emitter class)
  @Output() public childEventProperty= new EventEmitter();

  fireChildEvent(event){
    this.parentData="parentData Empty";
    this.childEventProperty.emit('Hey child message from test, value of alias
(name)='+ this.name);
  }
}
```

```
/*
    * Pipe  changes the value only in the view not in the
class
    */

/**
 * test.component.ts
 */

import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-test',
  // styleUrls: ['./test.component.css']
  styles: [`
      .text-success {
        color:green;
      }

      .text-danger{
        color:red;
      }

      .text-special{
        font-style:italic
      }
  `] ,
  // templateUrl: './test.component.html',
  //Pipe  changes the value only in the view not in the class
  template:`
    <h2>{{name}}</h2>
    <h2>name|lowercase: {{name|lowercase}}</h2>
    <h2>name|uppercase: {{name|uppercase}}</h2>
    <h2>name|titlecase: {{name|titlecase}}</h2>
    <h2>start at 3rd pos and stop at 10th pos, name|slice:3:10:
{{name|slice:3:10}}</h2>
    <h2>person|json: {{person|json}}</h2>
    <h2>5.123456789|number:'1.2-3' (number:'min.min-max'): {{5.123456789
|number:'1.2-3'}}</h2>
    <h2>5.123456789 |number:'2.4-5' (number:'min.min-max'): {{5.123456789
|number:'2.4-5'}}</h2>
    <h2>5.123456789 |number:'3.1-2' (number:'min.min-max'): {{5.123456789
|number:'3.1-2'}}</h2>
    <h2>5.123456789 |number:'4.5-9' (number:'min.min-max'): {{5.123456789
|number:'4.5-9'}}</h2>
    <h2>0.258| percent {{0.258| percent}}</h2>
    <h2>0.258| currency (usd by default)-> {{0.258| currency}}</h2>
    <h2>0.258| currency :'GBP'->    {{0.258| currency :'GBP'}}</h2>
```

```
    <h2>0.258| currency :'GBP':'code'->    {{0.258| currency :'GBP'
:'code'}}</h2>
    <h2>0.258| currency :'EUR'->    {{0.258| currency :'EUR'}}</h2>
    <h2>0.258| currency :'EUR':'code'->    {{0.258| currency :'EUR'
:'code'}}</h2>
    <h2>dateProperty -> {{dateProperty}}</h2>
    <h2>dateProperty|date:short -> {{dateProperty|date:short}}</h2>
    <h2>dateProperty|date:shortDate -> {{dateProperty|date:shortDate}}</h2>
    <h2>dateProperty|date:shortTime -> {{dateProperty|date:shortTime}}</h2>
  `
})
export class TestComponent {
  /*
   * Pipe  changes the value only in the view not in the class
   */
  public name ="KEJEIRI Mohamed";
  public person = {
    "firstName":"John",
    "lastName":"Doe"
  };
  public dateProperty = new Date();
}
```

Dependency Injection (used for services, @)

Principle in programming

DRY: ***Do*** not ***Repeat*** your ***Self***

Single Responsibility Principle: one component one responsibility

Hence come the ***Services***:

1- ***Sharing data*** across multiple components.

2- ***Implements application Logic***: Logic should be independent from any component view.

3- ***Use services for external interactions***: such as connecting to a databases.

So far we used hardcode data in the service; we need to get the data from a server instead

Hence ***HTTP and Observables***

Observable is an 'http response' getting fetched data from the server and sending back to the service.

Observables is sequence of items that arrive async over time, each single item is single http response arriving after an http call is made to the server.

```
@Injectable() //Decorator inside the service indicates that this service might also have dependencies injected in the future, the component includes @Component decorator which implicitly allow injection (includes decorator @Injectable).
```

Steps to implement observable:

http request -> receive observable cast into the *variable* with the needed type (e.g employee array) ->subscribe to observable from components (e.g employee and employeeDetail) ->assign the *variable* (e.g employee array) to a local variable.

RxJs: Reactive extensions for javascript library to work with observable (nothing todo with React lib from facebook).

```
/** * employees.json (_url="/assets/data/employees.json";) or an API */

[
  {"id":1, "name":"Andrew", "age":30},
  {"id":2, "name":"John", "age":41},
  {"id":3, "name":"Steve", "age":56},
  {"id":4, "name":"Bob", "age":28},
  {"id":5, "name":"Sylvia", "age":25}
]
```

```
/**
* app.module.ts
*/

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

import { AppComponent } from './app.component';
import { TestComponent } from './test/test.component';
import { EmployeeListComponent } from './employee-list/employee-
list.component';
import { EmployeeDetailComponent } from './employee-detail/employee-
detail.component';
import { EmployeeService } from './employee.service';
import {HttpClientModule} from '@angular/common/http';


@NgModule({
declarations: [
  AppComponent,
  TestComponent,
  EmployeeListComponent,
  EmployeeDetailComponent
],
imports: [
  BrowserModule,HttpClientModule //we also registering http service with the
injector we do not neet to add it to  providers...
],
providers: [EmployeeService],
bootstrap: [AppComponent]
})
export class AppModule { }
```

```
/*
* employee.service.ts
*/
import { Injectable } from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';
import { IEmployee } from './IEmployee';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/throw';

@Injectable() //@Injectable() //Decorator inside the service indicates that
this service might also have dependencies
            //injected in the future, the component includes @Component
decorator which implicitly
            //allow injection (includes decorator @Injectable).
export class EmployeeService {
private _url:string = "/assets/data/employees.json";
constructor(private http:HttpClient) { }

// getEmployees(){
//   return [
//     {id:1, name:"Andrew", age:30},
//     {id:2, name:"John", age:30},
//     {id:3, name:"Steve", age:30},
//     {id:4, name:"Bob", age:30},
//   ];
// }

getEmployees():Observable<IEmployee[]>{
  // return this.http.get<any>('https://api.github.com/users');//use any
instead of interface

  return this.http.get<IEmployee[]>(this._url)//Get request
  .catch(this.errorhandler);

}
errorhandler(error:HttpErrorResponse){
   return Observable.throw(error.message || "Server Error");
}

}
```

```ts
/*
* IEmployee.ts
*/

export interface IEmployee{
  id:number,
  name:string,
  age:number
}
```

```
/*
* employee-detail.component.ts
*/
import { Component, OnInit } from '@angular/core';
import {EmployeeService} from '../employee.service';
import { error } from 'selenium-webdriver';
@Component({  //the component includes @Component decorator which implicitly
            //allow injection (includes decorator @Injectable)
selector: 'app-employee-detail',
templateUrl: './employee-detail.component.html',
styleUrls: ['./employee-detail.component.css']
})
export class EmployeeDetailComponent implements OnInit {
public employees =[];
public errorMessage:string;
constructor(private _employeeService:EmployeeService) {}

ngOnInit() {
return this._employeeService.getEmployees()
.subscribe(data => this.employees = data,
            error => this.errorMessage = error);
}

}
```

```ts
/*
* employee-list.component.ts
*/
import { Component, OnInit } from '@angular/core';
import {EmployeeService} from '../employee.service';
import { IEmployee } from '../IEmployee';

@Component({  //the component includes @Component decorator which implicitly
            //allow injection (includes decorator @Injectable)
selector: 'app-employee-list',
templateUrl: './employee-list.component.html',
styleUrls: ['./employee-list.component.css']
})
export class EmployeeListComponent implements OnInit {

public employees = [];
public errorMessage:string;

constructor(private _employeeService:EmployeeService) { }

//get called once the component is initialized...
ngOnInit() {
 return this._employeeService.getEmployees()
 .subscribe(data => this.employees = data,
  error => this.errorMessage = error);
}

}


/*
* employee-list.component.html
*/

<h1>Employee list</h1>
<ul>
  <li *ngFor="let employee of employees">{{employee.name}}</li>
</ul>
<div>{{errorMessage}}</div>

<div>
  <h1>From Employee details</h1>
  <ul>
    <li *ngFor="let employee of employees">
      {{employee.id}} {{employee.name}} {{employee.age}}
    </li>
  </ul>
  <div>{{errorMessage}}</div>
</div>
```

```
<!—app.html-->
<div style="text-align:center">
  <h1>
  </h1>
  <!-- <app-test></app-test>   -->
  <app-employee-list></app-employee-list>
  <app-employee-detail></app-employee-detail>
</div>
```

```
/**
 * department-detail.component.ts
 */
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute,Router, ParamMap} from '@angular/router';
import { parse } from 'querystring';
import { ADDRGETNETWORKPARAMS } from 'dns';

@Component({
  selector: 'app-department-detail',
  template: `
    <h3>
      department with id = {{departmentId}}
    </h3>
    <br><br><br>
    <a (click)="previousElement(departmentId)">Previous</a>
    <a (click)="nextElement(departmentId)">Next</a>


  `,
  styles: []
})
export class DepartmentDetailComponent implements OnInit {
  public departmentId:number;
  constructor(private route:ActivatedRoute, private routerNavigate:Router) { }

  ngOnInit() {
    //get the parameter id from url

    /*
    * snapshot.paramMap get called only once through ngOnInit and didn't
refresh the template we next to subscribe to
    * an observable
    */

    //this.departmentId = parseInt(this.route.snapshot.paramMap.get('id'));
```

```
    //We subscribe to an observable ParamMap to watch the value id.
    this.route.paramMap
    .subscribe((params:ParamMap) => this.departmentId =
parseInt(params.get('id')));

  }

  previousElement(departmentId){
    let id = this.departmentId-1;
    this.routerNavigate.navigate(['/departments/',id]);
  }

  nextElement(departmentId){
    let id = this.departmentId+1;
    console.log(id);
    this.routerNavigate.navigate(['/departments/',id]);
  }
}
```

```
/**
 *  department-list.component.ts
 */
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-department-list',
  template: `
    <p>
      Department list
    </p>
    <ul>
    <li *ngFor="let department of departments"
(click)="OnSelect(department)"><span>{{department.id}}</span>
{{department.name}}</li>
    </ul>
  `,
  styles: []
})
export class DepartmentListComponent implements OnInit {
constructor(private _router:Router) { }

  OnSelect(department){
    this._router.navigate(['/departments', department.id]);
  }
  ngOnInit() {
  }

  //Dummy data.
  public departments = [
    {id:1, name:"Angular"},
    {id:2, name:"Node"},
    {id:3, name:"Mongo"},
    {id:4, name:"bootstrap"}
] ;

}
```

```ts
/**
 * employee-list.component.ts
 */
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-employee-list',
  template: `
    <p>
      employee-list works!
    </p>
  `,
  styles: []
})
export class EmployeeListComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

```
/**
 * app-routing.module.ts
 */
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { DepartmentListComponent } from './department-list/department-
list.component';
import { DepartmentDetailComponent } from './department-detail/department-
detail.component';
import { EmployeeListComponent } from './employee-list/employee-
list.component';
import { PageNotFoundComponent } from './page-not-found/page-not-
found.component';

const routes: Routes = [
  // {path:"", component: DepartmentListComponent},//not prefered instead use
the next one
  {path:'', redirectTo :'/departments', pathMatch: 'full' }, //pathMatch:
'prefix' : routes everything to departments since an empty string ('') can't
be a prefix
  {path:'departments', component: DepartmentListComponent},
  {path:'departments/:id', component: DepartmentDetailComponent},
  {path:'employees', component: EmployeeListComponent},
  {path:'**', component: PageNotFoundComponent} //wildcard route should be the
last one in the configuration
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
export const routingComponents =
[DepartmentListComponent,EmployeeListComponent, DepartmentDetailComponent]
```

```
/*
* app.module.ts
*/

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

import { AppRoutingModule,routingComponents } from './app-routing.module';

import { AppComponent } from './app.component';
import { PageNotFoundComponent } from './page-not-found/page-not-
found.component';
import { DepartmentDetailComponent } from './department-detail/department-
detail.component';
//DRY PB
// import { DepartmentListComponent } from './department-list/department-
list.component';
// import { EmployeeListComponent } from './employee-list/employee-
list.component';

/**
 * app.module.ts
 */
@NgModule({
  declarations: [
    AppComponent,
    // DepartmentListComponent,
    // EmployeeListComponent
    routingComponents, //allow us to avoid DRY PB!
    PageNotFoundComponent,
    DepartmentDetailComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```ts
/*
*  app.component.ts
*/
import { Component } from '@angular/core';

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

```html
<!-- app.component.html -->

 <div class = "container">
  <h1>
   Routing and navigation...
  </h1>
</div>
<ul>
  <li><a routerLink="/departments"
routerLinkActive="active">Departments</a></li>
  <li><a routerLink="/employees" routerLinkActive="active">Employees</a></li>
</ul>

<div class="jumbotron">
    <router-outlet></router-outlet>
</div>
<!-- router view goes here -->
```

```
<!-- Index.html -->
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Routing Demo</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```
