**Section10 Pipes in Angular**

**Notes: -**

**1-pipes are feature built in Angular which allow transform some output in the template**

**Such as built-in pipes {{username | uppercase}**

**2-some pipes can pass single or multiple parameters by adding : symbol as below**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<ul class="list-group">**

**<li class="list-group-item" \*ngFor="let server of servers" [ngClass]="getStatusClasses(server)">**

**<span class="badge">{{ server.status }}</span>**

**//we can pass parameter on the built in pipes**

**<strong>{{ server.name }}</strong> | {{ server.instanceType | uppercase }} |**

**{{ server.started | date:'fullDate' }}</li></ul></div></div></div>**

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

**@Component({**

**selector: 'app-root',**

**templateUrl: './app.component.html',**

**styleUrls: ['./app.component.css']})**

**export class AppComponent {**

**servers = [{**

**instanceType: 'medium',**

**name: 'Production Server',**

**status: 'stable',**

**started: new Date(15, 1, 2017)},**

**{instanceType: 'large',**

**name: 'User Database',**

**status: 'stable',**

**started: new Date(15, 1, 2017)},**

**{instanceType: 'small',**

**name: 'Development Server',**

**status: 'offline',**

**started: new Date(15, 1, 2017)},**

**{instanceType: 'small',**

**name: 'Testing Environment Server',**

**status: 'stable',**

**started: new Date(15, 1, 2017)}];**

**getStatusClasses(server: {instanceType: string, name: string, status: string, started: Date}) {**

**return {**

**'list-group-item-success': server.status === 'stable',**

**'list-group-item-warning': server.status === 'offline',**

**'list-group-item-danger': server.status === 'critical'};}}**

**3-by go to the Angular documentation search for the pipes and you will see the parameters can be passed https://angular.io/api/common/DatePipe**

**4-when chaining multiple pipes, the order is important because its execute from left to right**

**<strong>{{ server.name }}</strong> | {{ server.instanceType | uppercase }} |**

**{{ server.started | date:'fullDate' | uppercase }}**

**Create Custom Pipe**

**Steps: -**

**1-to create custom pipe we apply command Ng g pipe shorten**

**2-on the app.module.ts we import pipe just like directive and component**

**3-we can pass multiple parameters and seperate them with using : as below**

**//we can pass multiple parameter with using pipes as below**

**<strong>{{ server.name | shorten:10 }}</strong> | {{ server.instanceType | uppercase }} | {{ server.started | date:'fullDate' | uppercase | shorten:10:'xxx' }}**

**declarations: [AppComponent,ShortenPipe],**

**import { Pipe, PipeTransform } from '@angular/core';**

**@Pipe({name: 'shorten',})**

**export class ShortenPipe implements PipeTransform {**

**//we can pass multiple parameter as below**

**transform(value: any,limit:number,endingPart:string = '...') {**

**if (value.length > limit) {**

**return value.substr(0, limit) + endingPart;}return value;}}**

**Create Custom Filter in Pipe**

**Notes: -**

**1-angular does not rerun pipes whenever the data changes , only when reload the page it will triggered (otherwise it will make bad performance)**

**2-you can enforce the pipe to recalcoulate on each data entry but be aware that this cause bad performance**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<input type="text" [(ngModel)]="filteredStaus"/>**

**<br/>**

**<button class="btn btn-primary" (click)="onAddServer()">Add</button>**

**<ul class="list-group">**

**<li class="list-group-item" \*ngFor="let server of servers | filter:filteredStaus:'status'"**

**[ngClass]="getStatusClasses(server)">**

**<span class="badge"> {{ server.status }}</span>**

**<strong>{{ server.name | shorten:10 }}</strong> | {{ server.instanceType | uppercase }} | {{ server.started | date:'fullDate' | uppercase | shorten:10:'xxx' }}**

**</li></ul></div></div></div>**

**import { Pipe, PipeTransform } from '@angular/core';**

**//in order to make pipe recalcoulate every change on angular set pure: false**

**@Pipe({**

**name: 'filter',**

**pure:false})**

**export class FilterPipe implements PipeTransform {**

**//in transform we pass value which represent the array of server and the filter String which is the //status value and the property name which is used to filter value**

**transform(value: any,filterString:string,propName:string): any {**

**if(value.length === 0){return value;}**

**const resultArray = [];**

**for(const item of value){**

**if(item[propName].includes(filterString)){**

**resultArray.push(item);}}**

**return resultArray;}}**

**Understanding async**

**Notes: -**

**1-we can create custom pipe with async that working with the server side**

**(especially when working with observable that return after period of tiems and you want of preview on the screen , we can use async)**

**<div class="container">**

**<div class="row">**

**<div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">**

**<input type="text" [(ngModel)]="filteredStaus" />**

**<br /><hr>**

**//we can working with observable response by using async pipe as below**

**obserable response: {{appStatus | async}}**

**<hr>**

**<button class="btn btn-primary" (click)="onAddServer()">Add</button>**

**<ul class="list-group">**

**<li class="list-group-item" \*ngFor="let server of servers | filter:filteredStaus:'status'"**

**[ngClass]="getStatusClasses(server)">**

**<span class="badge">**

**{{ server.status }}**

**</span>**

**<strong>{{ server.name | shorten:10 }}</strong> | {{ server.instanceType | uppercase }} | {{ server.started | date:'fullDate' | uppercase | shorten:10:'xxx' }}**

**</li></ul></div></div></div>**

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

**@Component({**

**selector: 'app-root',**

**templateUrl: './app.component.html',**

**styleUrls: ['./app.component.css'],})**

**export class AppComponent {**

**appStatus = new Promise((resolve, reject) => {**

**setTimeout(() => {resolve('stable');}, 2000);});**

**servers = [{**

**instanceType: 'medium',**

**name: 'Production Server',**

**status: 'stable',**

**started: new Date(15, 1, 2017),},**

**{instanceType: 'large',**

**name: 'User Database',**

**status: 'stable',**

**started: new Date(15, 1, 2017),},**

**{instanceType: 'small',**

**name: 'Development Server',**

**status: 'offline',**

**started: new Date(15, 1, 2017),},**

**{instanceType: 'small',**

**name: 'Testing Environment Server',**

**status: 'stable',**

**started: new Date(15, 1, 2017),}];**

**filteredStaus: string = '';**

**getStatusClasses(server: {**

**instanceType: string;**

**name: string;**

**status: string;**

**started: Date;}) {**

**return {**

**'list-group-item-success': server.status === 'stable',**

**'list-group-item-warning': server.status === 'offline',**

**'list-group-item-danger': server.status === 'critical',};}**

**onAddServer() {**

**this.servers.push({**

**instanceType: 'small',**

**name: 'New Server',**

**status: 'stable',**

**started: new Date(),});}}**