**Lesson04 Custom Numeric Directive**

**Notes:-**

**1-In order to create custom Numeric Directive:-**

**1-we need to import decimal and percent pie inside custom pie**

**2-inject custom pie inside custom directive and using the custom pie to convert the 3 types of integer and decimal and percent to the syntax we want**

**Steps:-**

**1-on the model.ts we write the following class**

**export enum NumberTypes{**

**Integer = 1 ,**

**Decimal = 2,**

**Percentage = 3 ,**

**Currency = 4}**

**2-on the app.module.ts we import the**

providers: [

NgModel,

NumericPie,

DecimalPipe,

PercentPipe,

CurrencyPipe,

DatePipe]

export class AppModule { }

**3-on the custom pie we write the following code**

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

import { DecimalPipe, PercentPipe, CurrencyPipe } from '@angular/common';

import {NumberTypes} from '../Model/model';

@Pipe({name: 'numeric'})

export class NumericPie implements PipeTransform {

//we define reference variable that assign to the alias name of directive

@Input('numeric') digitInfo: string = "1.3-3";

//we inject Decimal Pipe and Percent Pipe and Currency Pipe

constructor(private decimalPipe: DecimalPipe, private percent: PercentPipe,private cur1:CurrencyPipe) { }

transform(value, numberType, digitInfo: string,currencyFormat:string = "") {

if(value == undefined || value == "") { value = "0"}

let numericValue = "0";

if (numberType == NumberTypes.Decimal) {

if (value.toString().includes(',')) {

if (value.indexOf(',') > -1) {

let arrValue = value.split(',');

let v = '';

arrValue.forEach(element => {

v += element;});

value = parseFloat(v);}}

return numericValue = this.decimalPipe.transform(value, digitInfo);}

else if (numberType == NumberTypes.Percentage) {

return numericValue = this.percent.transform((parseFloat(value) / 100), digitInfo);}

else if (numberType == NumberTypes.Integer) {

return numericValue = this.decimalPipe.transform(value, digitInfo).replace(",", "");}

//If the number type is currency format

else if (numberType == NumberTypes.Currency ){

//if the content not include $ symbol

if(!value.includes('$')){

numericValue = this.cur1.transform(value, currencyFormat,true,digitInfo);}

//If the content include $ symbol then parse the symbol $ and then assign the syntax $0.00

else{

let cur = value.split('$');

numericValue = this.cur1.transform(cur[1],currencyFormat,true,digitInfo);}

return numericValue}

else

return numericValue = value;}

//this method used to convert the Numeric format like 123,000 to 123.000

parse(value) {

if(value != ""){

if (value.toString().includes(','))

value = value.replace(",", "");

if (value.toString().includes('%'))

value = parseInt(value.substring(0, value.length - 1));

return parseFloat(value);}

else{return parseFloat("0.00");}}}

**4-on the custom Directive we set the following code**

import { Directive, ElementRef, OnInit, HostListener, Input, Self, ChangeDetectorRef } from '@angular/core';

import { NumberTypes } from '../Model/model';

import { NgModel, NgControl } from '@angular/forms';

import { DecimalPipe, PercentPipe,CurrencyPipe } from '@angular/common';

import { NumericPie } from '../pipes/numeric-pie';

@Directive({selector: '[NumFor]'})

export class NumericDirective implements OnInit {

private inputElement: HTMLInputElement;

@Input() numberType;

@Input() minDecimalPoint;

@Input() maxDecimalPoint;

@Input() allowNegative;

@Input() min;

@Input() max;

@Input() currencyFormat;

@Input() value;

//we using ngControl to bind the change to DOM Html element

constructor(private elementRef: ElementRef,private ngControl: NgControl,private ngModel: NgModel,private cdRef: ChangeDetectorRef,private numerPie:NumericPie){

this.inputElement = this.elementRef.nativeElement;}

ngOnInit() {

//the operation happen 2 way binding datatable

if (this.ngModel.name !== null) {

this.transform(this.ngModel.model);

this.ngModel.update.emit(this.inputElement.value);

this.cdRef.detectChanges();}

//the operation happen on reactive forms

else {this.transform(this.inputElement.value);}}

//when the user focus out the element

@HostListener("blur", ["$event.target.value"])

onBlur(value) {

if (value)

this.transform(value);}

//when the user focus in the element

@HostListener("focus", ["$event.target.value"])

onfocus(value) {

if (value)

this.transform(value);}

transform(value) {

if (this.allowNegative == "false" && value < 0) {

value = value \* -1;}

let digitInfo = "1.";

let numericValue = "0";

if (this.numberType == NumberTypes.Decimal) {

digitInfo += this.minDecimalPoint ? this.minDecimalPoint : '0';

digitInfo += this.maxDecimalPoint ? '-' + this.maxDecimalPoint : '-3';

numericValue = this.numerPie.transform(value, this.numberType, digitInfo);

if (!this.currencyFormat)

numericValue = numericValue.replace(",", "");}

else if (this.numberType == NumberTypes.Percentage) {

digitInfo += this.minDecimalPoint ? this.minDecimalPoint : '0';

digitInfo += this.maxDecimalPoint ? '-' + this.maxDecimalPoint : '-3';

numericValue = this.numerPie.transform(value, this.numberType, digitInfo);}

else if (this.numberType == NumberTypes.Integer) {

digitInfo += '0-0';

numericValue = this.numerPie.transform(value, this.numberType, digitInfo);}

else if (this.numberType == NumberTypes.Currency) {

digitInfo += this.minDecimalPoint ? this.minDecimalPoint : '0';

digitInfo += this.maxDecimalPoint ? '-' + this.maxDecimalPoint : '-3';

numericValue = this.numerPie.transform(value, this.numberType, digitInfo,this.currencyFormat);}

else {numericValue = value;}

if (this.min && this.numerPie.parse(numericValue) < this.min)

numericValue = this.numerPie.transform(this.min, this.numberType, digitInfo);

if (this.max && this.numerPie.parse(numericValue) > this.max)

numericValue = this.numerPie.transform(this.max, this.numberType, digitInfo);

this.ngControl.control.patchValue(numericValue);

this.inputElement.value = numericValue;}

@HostListener('keypress', ['$event'])

keyEvent(event: KeyboardEvent) {

if (this.numberType != undefined && this.numberType != '') {

let regexStr = '^[0-9\_\\.\\-]\*$';

return new RegExp(regexStr).test(event.key);}}}

**5-on the html page we set the following code**

<form class="form-horizontal" [formGroup]="empForm" (ngSubmit)="onSubmit()">

<div class="panel panel-primary">

<div class="panel-heading">

<h3 class="panel-title">Create Employee</h3>

</div>

<div class="panel-body">

<div class="row form-row" dragula="sortable">

<div class="form-group col-md-8">

<label class="control-label col-sm-5 col-xs-12">Signed Line %</label>

<ul class="input-group col-sm-7 col-xs-12">

//Integer Sample

<li><input type="text" formControlName="minSalary" class="form-control" required="required" pInputText NumFor

numberType="1" allowNegative="false" /></li>

//Decimal Sample

<li><input type="text" formControlName="Salary" class="form-control" required="required" pInputText NumFor numberType="2"

minDecimalPoint="2" maxDecimalPoint="4" allowNegative="false" /></li>

//Percentage Sample

<li><input type="text" class="form-control" min="1" max="100" formControlName="signedLine" required="required"

pInputText NumFor numberType="3" minDecimalPoint="2" maxDecimalPoint="2" allowNegative="false" />

</li>

//Currency Sample

<li><input type="text" formControlName="maxSalary" class="form-control" required="required" pInputText NumFor

numberType="4" allowNegative="false" currencyFormat="USD" minDecimalPoint="2" maxDecimalPoint="2" />

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

</div></div>

<div class="panel-footer">

<button class="btn btn-primary" type="submit">Save</button></div>

</div>

</form>