Marvellous Infosystems: Angular Assignment 11

Name: Shrirang Jagdish Nikam

Enrollment No: 396AM_Shrirang

1. Create angular application which creates one custom pipe named as MyRev. This custom pipe accept string as a value and display the string reverse format.

Example:
Name = "Marvellous': {{Name | MyRev}} Output of the above statement should be suollevraM
To solve this question please follow all the steps to create Custom pipe. Inside transform method of pipe write logic to reverse the string.
Consider below code snippet:

```
Import { Pipe, PipeTransform } from '@angular/core';
@Pipe({name: 'myRev'))
export class MyRev Implements PipeTransform
{
transform(value: string): string
{
// Create new string to hold the reverse string let temp: string="";
// Traverse the string from end and copy the contents in temp.
return temp;
}
}
```

Answer:

App.component.html:

```
<h1>{{ 'Marvellous' | myRev }}</h1>
```

App.module.ts:

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

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

```
import { MyRev } from './my-rev.pipe';

@NgModule({
  declarations: [
    AppComponent,
    MyRev
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

my-rev.pipe.ts:

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

@Pipe({name: 'myRev'})
export class MyRev implements PipeTransform {
  transform(value: string): string {
    let temp: string = "";
    for (let i = value.length - 1; i >= 0; i--) {
      temp += value.charAt(i);
    }
    return temp;
  }
}
```

Output:

suollevraM

2. **Create angular application which creates two custom pipe named as MyAdd & MyMult. MyAdd pipe is used to add two numbers and MyMult is used to multiply two numbers.**

Example:

((8| MyAdd: 3)) Output should be 11 ((7| MyMult: 3)) Output should be 21

For this purpose we have to create two separate pipes and Inside transform method of each pipe we have to write logic of addition and multiplication respectively.

Consider below code snippet for reference

import { Pipe, PipeTransform } from '@angular/core';
@Pipe ({ MyAdd'
name: 'MyAdd'
})
export class MyAdd Implements PipeTransform
{
transform(value: number, Param: string): number
{
// return addition of value and Param
}} Similarly we can write logic for MyMult pipe also.

Answer:

App.component.html:

```
<p>8 + 3 = {{ 8 | myAdd: 3 }}</p>
<p>7 * 3 = {{ 7 | myMult: 3 }}</p>
```

App.module.ts:

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

import { AppComponent } from './app.component';
import { MyAddPipe } from './my-add.pipe';
import { MyMultPipe } from './my-mult.pipe';

@NgModule({
  declarations: [
```

```
    AppComponent,
    MyAddPipe,
    MyMultPipe
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

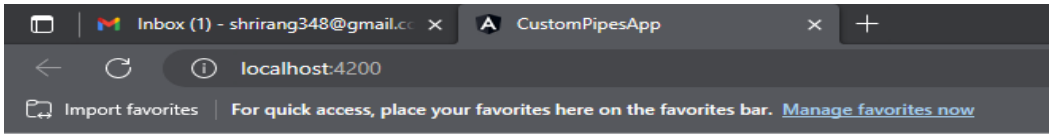
my-add.pipe.ts:

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

@Pipe({
  name: 'myAdd'
})
export class MyAddPipe implements PipeTransform {
  transform(value: number, param: number): number {
    return value + param;
  }
}
```

My-mult.pipe.ts:

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

@Pipe({
  name: 'myMult'
})
export class MyMultPipe implements PipeTransform {
  transform(value: number, param: number): number {
    return value * param;
  }
}
```

Output:



8 + 3 = 11

7 * 3 = 21

3. **Create angular application which creates one custom pipe named as MarvellousChk. This custom pipe accept one Integer as a value and parameter can be Prime, Perfect, Even, Odd. Depends on the parameter we have to check status of the number and return the result accordingly.**

Example: No = 11; {{No | Even}}
Output of the above statement should be "It is not Even" {{No | Prime))
Output of the above statement should be "It is Prime number"
{{No | Prime}
Output of the above statement should be "It is not Perfect number"
To solve this question please follow all the steps to create Custom pipe. Inside transform method depends on the parameter we have to write separate logic to check whether number is even or odd or prime or perfect.
Consider below code snippet:

```
import { Pipe, PipeTransform) from '@angular/core';
@Pipe({name: 'MarvellousChk'})
export class MarvellousChk implements PipeTransform transform(value: Number, Param: string): string
{
if(Param "Prime") ==
{
// Logic to check whether value is prime or not.
} if(Param == = "Perfect")
{
// Logic to check whether value is Perfect or not.
if(Param == "Even")
// Logic to check whether value is Even or not.
if(Param == "Odd")
{
// Logic to check whether value is Odd or not.
}
}
```

}
Return the string which displays the result.

Answer:

Marvellous-chk.pipe.ts:

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

@Pipe({name: 'MarvellousChk'})
export class MarvellousChk implements PipeTransform {
  transform(value: number, param: string): string {
    if (param === "Prime") {
      if (value <= 1) {
        return "It is not Prime number";
      }
      for (let i = 2; i <= Math.sqrt(value); i++) {
        if (value % i === 0) {
          return "It is not Prime number";
        }
      }
      return "It is Prime number";
    } else if (param === "Perfect") {
      let sum = 0;
      for (let i = 1; i < value; i++) {
        if (value % i === 0) {
          sum += i;
        }
      }
      if (sum === value) {
        return "It is Perfect number";
      } else {
        return "It is not Perfect number";
      }
    } else if (param === "Even") {
      if (value % 2 === 0) {
        return "It is Even number";
      } else {
        return "It is not Even number";
      }
    } else if (param === "Odd") {
      if (value % 2 !== 0) {
```

```
      return "It is Odd number";
     } else {
      return "It is not Odd number";
     }
   } else {
    return "Invalid parameter";
   }
  }
}
```

App.module.ts:

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

import { AppComponent } from './app.component';
import { MarvellousChk } from './marvellous-chk.pipe';

@NgModule({
  declarations: [
    AppComponent,
    MarvellousChk
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html:

```
<h1>MarvellousChk Pipe Example</h1>

<p>Number: 11</p>
<p>{{ 11 | MarvellousChk:'Even' }}</p>
<p>{{ 11 | MarvellousChk:'Prime' }}</p>
<p>{{ 11 | MarvellousChk:'Perfect' }}</p>
<p>{{ 11 | MarvellousChk:'Odd' }}</p>
```

```
<p>Number: 12</p>
<p>{{ 12 | MarvellousChk:'Even' }}</p>
<p>{{ 12 | MarvellousChk:'Prime' }}</p>
<p>{{ 12 | MarvellousChk:'Perfect' }}</p>
<p>{{ 12 | MarvellousChk:'Odd' }}</p>
```

Output: