# Angular 4 - Pipes

Pipes were earlier called filters in Angular1 and called pipes in Angular 2 and 4.

The | character is used to transform data. Following is the syntax for the same

{{ Welcome to Angular 4 | lowercase}}

It takes integers, strings, arrays, and date as input separated with **|** to be converted in the format as required and display the same in the browser.

Let us consider a few examples using pipes.

Here, we want to display the text given to uppercase. This can be done using pipes as follows −

In the **app.component.ts** file, we have defined the title variable −

### app.component.ts

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular 4 Project!';

}

The following line of code goes into the **app.component.html** file.

<b>{{title | uppercase}}</b><br/>

<b>{{title | lowercase}}</b>

The browser appears as shown in the following screenshot −



Angular 4 provides some built-in pipes. The pipes are listed below −

* Lowercasepipe
* Uppercasepipe
* Datepipe
* Currencypipe
* Jsonpipe
* Percentpipe
* Decimalpipe
* Slicepipe

We have already seen the lowercase and uppercase pipes. Let us now see how the other pipes work.

The following line of code will help us define the required variables in **app.component.ts** file −

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

@Component({

selector: 'app-root',

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

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular 4 Project!';

todaydate = new Date();

jsonval = {name:'Rox', age:'25', address:{a1:'Mumbai', a2:'Karnataka'}};

months = ["Jan", "Feb", "Mar", "April", "May", "Jun",

"July", "Aug", "Sept", "Oct", "Nov", "Dec"];

}

We will use the pipes in the **app.component.html** file.

<!--The content below is only a placeholder and can be replaced.-->

<div style = "width:100%;">

<div style = "width:40%;float:left;border:solid 1px black;">

<h1>Uppercase Pipe</h1>

<b>{{title | uppercase}}</b><br/>

<h1>Lowercase Pipe</h1>

<b>{{title | lowercase}}</b>

<h1>Currency Pipe</h1>

<b>{{6589.23 | currency:"USD"}}</b><br/>

<b>{{6589.23 | currency:"USD":true}}</b> //Boolean true is used to get the sign of the currency.

<h1>Date pipe</h1>

<b>{{todaydate | date:'d/M/y'}}</b><br/>

<b>{{todaydate | date:'shortTime'}}</b>

<h1>Decimal Pipe</h1>

<b>{{ 454.78787814 | number: '3.4-4' }}</b> // 3 is for main integer, 4 -4 are for integers to be displayed.

</div>

<div style = "width:40%;float:left;border:solid 1px black;">

<h1>Json Pipe</h1>

<b>{{ jsonval | json }}</b>

<h1>Percent Pipe</h1>

<b>{{00.54565 | percent}}</b>

<h1>Slice Pipe</h1>

<b>{{months | slice:2:6}}</b>

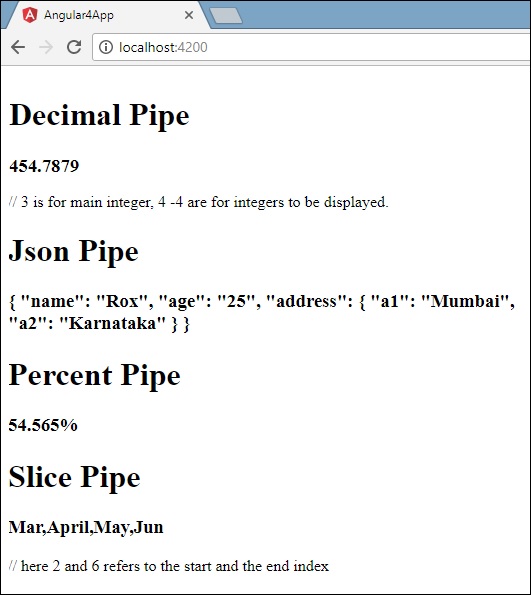
// here 2 and 6 refers to the start and the end index

</div>

</div>

The following screenshots show the output for each pipe −





Angular 2 - Custom Pipes

Angular 2 also has the facility to create custom pipes. The general way to define a custom pipe is as follows.

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

@Pipe({name: 'Pipename'})

export class Pipeclass implements PipeTransform {

transform(parameters): returntype { }

}

Where,

* **'Pipename'** − This is the name of the pipe.
* **Pipeclass** − This is name of the class assigned to the custom pipe.
* **Transform** − This is the function to work with the pipe.
* **Parameters** − This are the parameters which are passed to the pipe.
* **Returntype** − This is the return type of the pipe.

import {

Pipe,

PipeTransform

} from '@angular/core';

@Pipe ({

name: 'Multiplier'

})

export class MultiplierPipe implements PipeTransform {

transform(value: number, multiply: string): number {

let mul = parseFloat(multiply);

return mul \* value

}

}

Following points need to be noted about the above code.

* We are first importing the Pipe and PipeTransform modules.
* Then, we are creating a Pipe with the name 'Multiplier'.
* Creating a class called MultiplierPipe that implements the PipeTransform module.
* The transform function will then take in the value and multiple parameter and output the multiplication of both numbers.

**Step 3** − In the app.component.ts file, place the following code.

import {

Component

} from '@angular/core';

@Component ({

selector: 'my-app',

template: '<p>Multiplier: {{2 | Multiplier: 10}}</p>'

})

export class AppComponent { }

**Note** − In our template, we use our new custom pipe.

**Step 4** − Ensure the following code is placed in the app.module.ts file.

import {

NgModule

} from '@angular/core';

import {

BrowserModule

} from '@angular/platform-browser';

import {

AppComponent

} from './app.component';

import {

MultiplierPipe

} from './multiplier.pipe'

@NgModule ({

imports: [BrowserModule],

declarations: [AppComponent, MultiplierPipe],

bootstrap: [AppComponent]

})

export class AppModule {}

Following things need to be noted about the above code.

* We need to ensure to include our MultiplierPipe module.
* We also need to ensure it is included in the declarations section.

<table>

<thead>

<tr>

<th> Code </th>

<th>Name </th>

<th> Gender </th>

<th> Annual Salary</th>

<th> Date of Birth </th>

</tr>

</thead>

<tbody>

<tr \*ngFor = "let x of employees">

<td> {{x.code | uppercase}} </td>

<td>{{x.name | | gender : x.gender }}</td>

<td>{{x.gender }} </td>

<td> {{x.annualSalary | currency :'INR':true:'1.3-3'}} </td>

<td>{{x.dateOfBith | date :'dd/MM/yyyy'}} <br>

Full Date Format:- {{x.dateOfBith | date:"fullDate"|uppercase}}<br/>

Medium:-{{x.dateOfBith | date:"medium"}}<br/>

Short:-{{x.dateOfBith

| date:"short"}}<br/>

Specific Date Format:-{{x.dateOfBith | date:"dd/MM/yyyy"}}<br/>

Short Time format:-{{x.dateOfBith | date:"shortTime"}}<br/>

Medium Time:-{{x.dateOfBith | date:"mediumTime"}}<br/>

Medium Date format:- {{x.dateOfBith | date:"mediumDate"}}<br/>

Full Date Format:- {{x.dateOfBith | date:"fullDate"}}<br/> </td>

</tr>

</tbody>

</table>

Gender

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

@Pipe({

name: 'gender'

})

export class GenderPipe implements PipeTransform {

// transform(value: any, args?: any): any {

// return null;

// }

transform(value: any, gender: string): string {

if (gender.toLowerCase() == "male")

return "Mr." + value;

else

return "Miss." + value; }

}