Angular Pipes

- Pipe is used to transform data.
- Angular is used in front-end to consume and present the data coming from various data sources.
- The data type of source provided and the data types supported in TypeScript will not match.
- Hence the data is not displayed in the same format how we are expecting.
- Pipe can transform the data and display in desired format.
- Angular pipe is a class that implements “PipeTransform” base.
- The functionality of pipe is defined by using “transform()” method.
- You can create custom pipes or implement built-in pipes.

Syntax:

```
import { PipeTransform } from ‘@angular/core’;
@Pipe({
   name: ‘someNameforPipe’
})
export class YourPipeName implements PipeTransform
{
   transform(value) {
         //transforms your value
```

```
        return value;
    }
}
```

- The pipes are applied to data in presentation by using "|"

Syntax:

{{ yourdata | pipeName:options }}

// options are pipe parameters.

// You can chain Pipes [Add multiple pipes]

- Angular pipes are by default "Pure" pipes.
- A "**Pure**" pipe will not change the value, it will just apply a format for value.
- If a Pipe can change the state and value then it is referred as "**Impure**" pipe.
- Angular built-in pipes are:
    - AsyncPipe
    - **CurrencyPipe**
    - **DatePipe**
    - **DecimalPipe**
    - **I18PluralPipe**
    - **I18SelectPipe**
    - **JsonPipe**
    - **KeyValuePipe**
    - **LowerCasePipe**
    - **UpperCasePipe**
    - **TitleCasePipe**

- **PercentPipe**
- **SlicePipe**

Pipe	Name	Description

UpperCase Pipe	uppercase	It changes the capitalization of text by converting all letters into Upper Case letters.
		
		Ex:
		public product = {
		Name: 'samsung tv',
		Price: 56000.50,
		Mfd: new Date('2020-03-22')
		};
		{{product.Name | uppercase}}

LowerCase Pipe	lowercase	It converts all letters into lowercase letters.
		
		Ex:
		public product = {
		Name: 'samsung tv',
		Price: 56000.50,
		Mfd: new Date('2020-03-22')
		};
		{{product.Name | lowercase}}

TitleCasePipe	titlecase	It converts the first letter of every word in a sentence into upper case letter. Ex: public product = { Name: 'samsung tv',

		Price: 56000.50, Mfd: new Date('2020-03-22') }; {{product.Name | titlecase}}
DecimalPipe	number	It is used to display numeric value with thousands separator and fractions. It comprises of digits information: **Mimimum-Integer-Digits** - It specifies the number of digits to display before fraction. - It will not slice the digits, it just defines the rule for number of digits. **Minimum-Fraction-Digits** - It specifies the minimum number of fractions to define. **Maximum-Fraction-Digits** - It specifies the maximum number of factions to define. Syntax: {{ data | number }} {{ data | number: {minIntegerDigits}.{minFractionDigits}-{maxFractionDigits} Ex: public product = {

		Name: 'samsung tv', Price: 56000.50, Mfd: new Date('2020-03-22') }; {{product.Price}} → 56000.5 {{product.Price | number}} → 56,000.5 {{product.Price | number:'5.2-4'}} → 56,000.50
CurrencyPipe	currency	It is used to display number in a currency format, with fractions and currency symbol. Syntax: {{ data | currency: 'currencyFormat':'digitsInfo'}} Ex: {{product.Price | currency:'INR':'5.4-4'}} {{product.Price | currency:'₹':'5.4-4'}}
DatePipe	date	It is used for display the date and time value in various date and time formats. You can use pre-defined formats for date or your can define custom format. **Pre-defined Formats:**

		- short - medium - long - full - shortDate - mediumDate - longDate - fullDate - shortTime - mediumTime - longTime - fullTime Syntax: {{ yourDateValue | date: 'shortDate' }} {{product.Mfd | date:'longDate'}} **Custom Format:** MM – 2 Digits Month MMM – Short Month Name MMMM – Long Month Name dd – 2 digits date d – 1 Digit date yy – 2 Digits Year yyyy – 4 Digits Year Syntax: {{product.Mfd | date:'dd-MMMM-yyyy'}}
PercentPipe	percent	Transforms a number into percentage string.

		Syntax:
		{{ value | percent: ‘digitsInfo’ }}
		
		Ex:
		public product = {
		Name: 'samsung tv',
		Price: 56000.500,
		Mfd: new Date('2020-03-22'),
		Sales: 0.259
		};
		{{product.Sales | percent:'2.2-2'}}

SlicePipe	slice	It creates a new Array or string containing a subset (sliced) of the elements. It can extract values within the specified index range and returns an array. Syntax: {{ collection | slice: startIndex: endIndex }} Ex: public products = ['TV', 'Mobile', 'Shoe', 'Watch']; <li *ngFor="let item of products | slice:1:3"> {{item}}

JsonPine	json	It converts the data into JSON format.
		JSON formatted data is used to transport via API.
		
		Syntax:
		{{ data | json }}
		
		Ex:
		public product = {
		Name: 'samsung tv',
		Price: 56000.500,
		Mfd: new Date('2020-03-22'),
		Sales: 0.259
		};
		
		<pre>
		{{product | json}}
		</pre>
		
		O/P:
		{
		"Name": "samsung tv",
		"Price": 56000.5,
		"Mfd": "2020-03-22T00:00:00.000Z",
		"Sales": 0.259
		}

KeyValuePipe	keyvalue	It is used to transform an object or map into an array of key and value

		pairs.
		It provides the properties:
		- key: used to read all keys from collection
		- value: used to read all values from collection
		Syntax:
		{{ *ngFor=”item of data/collection | keyvalue” }}
		
		Ex:
		public products = ['TV', 'Mobile', 'Shoe', 'Watch'];
		public data: {[key:number]:string} =
		{
		1: 'Samsung TV', 2: 'Nike Casuals'
		};
		
		<ul class="list-unstyled">
		<li *ngFor="let item of products | keyvalue">
		{{item.key}} : {{item.value}}
		
		<ul class="list-unstyled">
		<li *ngFor="let item of data | keyvalue">
		{{item.key}} - {{item.value}}

I18nSelect Pipe	i18Select	- I18 is a community of Angular.
		- It designed a Pipe for angular.
		- It is a Generic selector that can make decision dynamically according to the state and situation, and define the result when the relative condition is matching.
		- In Early version we have to depend on lot of iterations and conditions.
		Syntax:
		{{value_expression | i18Select: mapping }}
		
		Ex:
		export class PipedemoComponent{
		public products = [
		{Name: 'Samsung TV', City: 'Delhi'},
		{Name: 'Nike Casuals', City: 'Hyderabad'},
		{Name: 'Mobile', City: 'Mumbai'},
		{Name: 'Watch', City: 'Goa'}
		];
		public statusMessge = {
		'Hyderabad': 'Delivery in 2 Days',
		'Delhi': 'Delivery in 5 Days',
		'Mumbai': 'Not Deliverable',

		'other': 'Unkown - We Will Update'
		};
		}
		
		 Products Status
		<table class="table table-hover">
		<thead>
		<tr>
		<th>Name</th>
		<th>City</th>
		<th>Delivery Status</th>
		</tr>
		</thead>
		<tbody>
		<tr *ngFor="let item of products">
		<td>{{item.Name}}</td>
		<td>{{item.City}}</td>
		<td>{{item.City|i18nSelect:statusMessge}}</td>
		</tr>
		</tbody>
		</table>

I18PluralPipe	i18nPlural	- As per coding standards we use singular name for any reference that is storing single value and plural name for reference storing multiple

		values.
		Syntax:
		product = { };
		products = [];
		
		- Plural pipe can identify whether the object comprises of single or multiple values and defines a plural name dynamically.
		- It can get collection count and display messages according to the count.
		- It uses a map or object to verify the values.
		
		Syntax:
		{{ collection.length | i18Plural:keyValueCollection}}
		
		Ex:
		export class PipedemoComponent{
		public notifications = ['John Called', 'Sam Called', 'Raj Called'];
		public notificationsMap: {[key:string]:string} ={
		'=0': 'No Missed Calls', '=1': 'One Missed Call', 'other': '# Missed Calls'
		}
		}

		  Plural Demo ^{{{notifications.length | i18nPlural: notificationsMap}}}
AsyncPipe	async	- It handles asynchronous requests. - It can access content by using unblocking technique.

Which pipe we can name as “Impure Pipe”?

- I18nSelectPipe
- I18nPluralPipe
- SlicePipe

Custom Pipe

- Angular allows to configure and create your own pipe that can server any specific situation in your application.
- Pipe is a class that implements “PipeTransform” base class, which is defined in “@angular/core”.
- Pipes related meta data is defined by using “@Pipe()” marker [directive]
- Pipes are defined with functionality by implementing “transform()” method of “PipeTransform” base.
- Every custom pipe you define must be registered in “app.module.ts”.

Syntax:

```
import { PipeTransform } from ‘@angular/core’;
@Pipe(
 { name: ‘selectorForPipe’ }
)
export class YourPipeName implements PipeTransform
{
    transform() {
       return transformation;
    }
}
```

Ex:

1. Add a new folder by name “CustomPipes”
2. Add a new file into folder “sentencecase.pipe.ts”
   [>ng generate pipe pipeName]
3. “sentencecase.pipe.ts

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

    @Pipe({
       name: 'sentencecase'
    })
    export class SentenceCasePipe implements
    PipeTransform
```

```
{
   transform(data){
      let firstChar = data.charAt(0);
      let restChars = data.substring(1);
      let sentence  = firstChar.toUpperCase() +
restChars.toLowerCase();
      return sentence;
   }
}
```

4. Register Pipe in “app.module.ts”

```
import { SentenceCasePipe } from
'./CustomPipes/SentenceCase.pipe';

declarations: [
SentenceCasePipe,
]
```

5. Apply to your data

```
public msg  = ‘weCOME to AnGular’;
{{ msg | sentencecase }}
```

Try:

- Create a pipe for sorting a list and printing the list of values from array.

```
someCollection = [ ];
{{ *ngFor=”” | yourPipe }}
```

- Create a pipe for filtering the value and display only the value that match your word.
  {{ collection | pipe: ‘string’ }}