**Angular Pipes with Examples**

**Why we need Angular Pipes?**

As we already know every web application starts with a simple task: first get the data, then transform the data into some format, and finally, show the formatted data to the users.

Getting the data is very simple, you can create a local variable or a complex type to hold the data or even you may get the data from APIs.

Once you get the data, then you could show the raw data as it is to the end-user, but that will not make a good user experience. To get a good user experience we need to modify the raw data into some specific format and in such cases, Angular Pipes plays an important role.

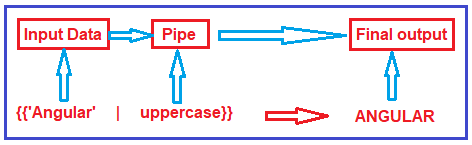
**What are Pipes in Angular Application?**

The Angular Pipe takes the raw data as input and then transforms the raw data into some desired format. So in simple words, we can say that the angular pipes transform the data into a specific format before displaying them to the end-users.

Using the Pipe (|) operator, we can apply the pipes features to any of the property in angular application. There are so many built-in pipes provides by Angular Framework such as lowercase, uppercase, titlecase, decimal, date, percent, currency etc. It is also possible to create custom pipes.

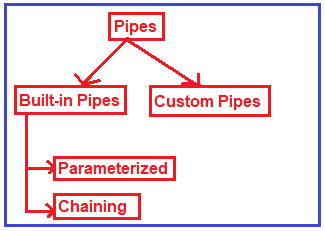
**Syntax to use Pipes in Angular Application:**

Syntaxes are written inside the HTML. The syntax to use the Angular Pipe is given below. To apply a pipe on a property, you need to use the pipe operator “|”.



**Types of Pipes in Angular:**

The Angular Framework divided the Pipes into two types i.e. Built-in Pipes and Custom Pipes. Further Built-in Pipes are divided into two types i.e. Parameterized and chaining as shown in the below image.



**Example to understand Angular Pipes:**

Let us see an example to understand pipes. First we will see the output without pipes and then we will see the output with pipes.

**Modify app.component.ts file:**

Please modify the app.component.ts file as shown below. Here, we have created one student array with some dummy data that we want to show in the web page.

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

@Component**({**

selector: 'app-root',

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

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

**})**

**export** **class** AppComponent **{**

students: **any[]** = **[**

**{**

ID: 'std101', Name: 'RaKesh ROUT',

DOB: '12/8/1988', Gender: 'Male', CourseFee: 1234.56

**}**,

**{**

ID: 'std102', Name: 'ANURAG Mohanty',

DOB: '10/14/1989', Gender: 'Male', CourseFee: 6666.00

**}**,

**{**

ID: 'std103', Name: 'Priyanka Dewangan',

DOB: '7/24/1992', Gender: 'Female', CourseFee: 6543.15

**}**,

**{**

ID: 'std104', Name: 'Hina SHARMA',

DOB: '8/19/1990', Gender: 'Female', CourseFee: 9000.50

**}**,

**{**

ID: 'std105', Name: 'SamBIt SataPATHY',

DOB: '4/12/1991', Gender: 'Male', CourseFee: 9876.54

**}**

**]**;

**}**

**Modify app.component.html file:**

Please modify **app.component.html**file as shown below. As you can see at the moment we are not using any pipes.

**<table** border="1"**>**

**<thead>**

**<tr>**

**<th>**Student ID**</th>**

**<th>**Name**</th>**

**<th>**DOB**</th>**

**<th>**Gender**</th>**

**<th>**Course Fee**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID}}**</td>**

**<td>**{{student.Name}}**</td>**

**<td>**{{student.DOB}}**</td>**

**<td>**{{student.Gender}}**</td>**

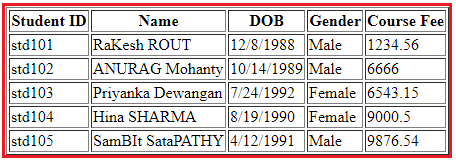
**<td>**{{student.CourseFee}}**</td>**

**</tr>**

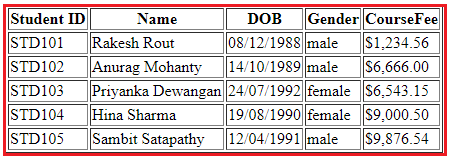
**</tbody>**

**</table>**

Now, if you browse the application, then you will get the following output in the browser.



As you can see in the above image, the data is not that user friendly. Let us discuss how we want to display the above data. We want to display the Student ID in upper case character and Name using the title case i.e. first character of every word in uppercase. Again, we want to display display the Date of Birth in **MM/DD/YYYY** format. We can achieve this by using the date pipe. Again we want to display the Gender in lower case and finally, we want to display the course Fee using the $ sign as shown in the below image.



**How can we achieve the above output?**

In order to achieve the desired output, we are going to use the following built-in pipes.

1. **lowercase**: This is used to convert the characters into lower case.
2. **uppercase**: This is used to convert the characters into upper case.
3. **titlecase**: This built-in pipe is used to convert the first character in each word to upper case.
4. **date**: This pipe is used to convert a date to some specific format.
5. **currency**: this pipe is used to convert number to currency with currency symbol.

So, modify the **app.component.html** file as shown below to use the required built-in pipes to get the desired output.

**<table** border="1"**>**

**<thead>**

**<tr>**

**<th>**Student ID**</th>**

**<th>**Name**</th>**

**<th>**DOB**</th>**

**<th>**Gender**</th>**

**<th>**CourseFee**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID | uppercase}}**</td>**

**<td>**{{student.Name | titlecase}}**</td>**

**<td>**{{student.DOB | date:'dd/MM/yyyy'}}**</td>**

**<td>**{{student.Gender | lowercase}}**</td>**

**<td>**{{student.CourseFee | currency:'USD':true}}**</td>**

**</tr>**

**</tbody>**

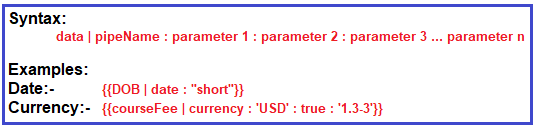
**</table>**

Now, run the application and you should get the output as expected. Here, I just show you how to use Pipes with simple examples.

## ****Angular Parameterized Pipes with Examples****

**What are Angular Parameterized Pipes?**

In Angular, we can pass any number of parameters to the pipe using a colon (:) and when we do so, it is called Angular Parameterized Pipes. The syntax to use Parameterized Pipes in Angular Application is given below.



**Date Pipe:**

Let us understand the Parameterized Date pipes with some examples. When you worked with any real-time applications, then you need to display the date time data in different formats. Here, I am going to show you some of the formats then I will provide you the link from where you will get all the available data formats.

**Modify app.component.ts file**

Open **app.component.ts** file and then copy and paste the following code in it. Here we simply create a variable i.e. today to hold the current data. As you can see you can use **Date.now()** to get the current date in typescript.

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

@Component**({**

selector: 'app-root',

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

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

**})**

**export** **class** AppComponent **{**

today: **number** = Date.now**()**;

**}**

**Modify app.component.html file**

Open **app.component.html** file and then copy and paste the following code in it. As you can see, here we are using the parameterized data pipe to show different date formats.

**<p>**Date Pipe : {{today | date}}**</p>**

**<p>**Full Date : {{today | date:'fullDate'}}**</p>**

**<p>**Mediate Date : {{today | date:'medium'}}**</p>**

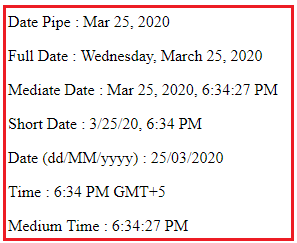
**<p>**Short Date : {{today | date:'short'}}**</p>**

**<p>**Date (dd/MM/yyyy) : {{today | date:'dd/MM/yyyy'}}**</p>**

**<p>**Time : {{today | date:'h:mm a z'}}**</p>**

**<p>**Medium Time : {{today | date:'mediumTime'}}**</p>**

With the above changes in place nor run the application and you should the date in different formats in the web page as shown in the below image.



##### **Currency Pipe:**

The Angular Currency Pipe is used to transforms a number to a currency string, formatted according to locale rules that determine group sizing and separator, decimal-point character, and other locale-specific configurations. Let us understand this with an example.

##### **Step1: Modify app.component.ts**

Open **app.component.ts** file and then copy and paste the following code in it. Here, we just created one property of type number.

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

@Component**({**

selector: 'app-root',

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

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

**})**

**export** **class** AppComponent **{**

salary: **number** = 456723.50;

**}**

##### **Step2: Modify app.component.html file**

Open **app.component.html** file and then copy and paste the following code in it.

**<p>**Currency USD in Symbol : {{salary | currency:'USD':true}}**</p>**

**<p>**Currency INR in Symbol : {{salary | currency:'INR':true}}**</p>**

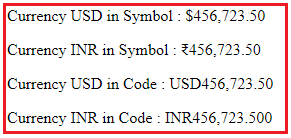
**<p>**Currency USD in Code : {{salary | currency:'USD':false:'4.2-2'}}**</p>**

**<p>**Currency INR in Code : {{salary | currency:'INR':false:'1.3-3'}}**</p>**

##### **Let us understand the above code.**

1. The first parameter is the currency Code (i.e. USD or INR)
2. The second parameter is boolean – True to display the currency symbol where as false to display the currency code.
3. The third parameter (‘1.3-3’ or ‘4.2-2’) specifies the number of integer and fractional digits.

Now save the changes and have a look at the browser and you should get the following output.



## ****Creating Angular Custom Pipe****

##### **Let us understand the need of Angular Custom Pipe with an example.**

Suppose you want to display the students detail in a web page. So, let us first create the student data in the AppComponent.

##### **Modify app.component.ts file:**

Open **app.component.ts** file and then copy and paste the following code in it. As you can here we simply created an student array.

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

@Component**({**

selector: 'app-root',

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

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

**})**

**export** **class** AppComponent **{**

students: **any[]** = **[**

**{**

ID: 'std101', Name: 'Rakesh Rout',

DOB: '12/8/1988', Gender: 'Male', CourseFee: 1234.56

**}**,

**{**

ID: 'std102', Name: 'Anurag Mohanty',

DOB: '10/14/1989', Gender: 'Male', CourseFee: 6666.00

**}**,

**{**

ID: 'std103', Name: 'Priyanka Dewangan',

DOB: '7/24/1992', Gender: 'Female', CourseFee: 6543.15

**}**,

**{**

ID: 'std104', Name: 'Hina Sharma',

DOB: '8/19/1990', Gender: 'Female', CourseFee: 9000.50

**}**,

**{**

ID: 'std105', Name: 'Sambit Satapathy',

DOB: '4/12/1991', Gender: 'Male', CourseFee: 9876.54

**}**

**]**;

**}**

Let us show these student data in the web page.

##### **Modify app.student.html file:**

Open app.student.html file and then copy and paste the following code in it. As you can here we have applied some built-in pipes to format the data.

**<table** border="1"**>**

**<thead>**

**<tr>**

**<th>**Student ID**</th>**

**<th>**Name**</th>**

**<th>**DOB**</th>**

**<th>**Gender**</th>**

**<th>**CourseFee**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID | uppercase}}**</td>**

**<td>**{{student.Name }}**</td>**

**<td>**{{student.DOB | date:'dd/MM/yyyy'}}**</td>**

**<td>**{{student.Gender | lowercase}}**</td>**

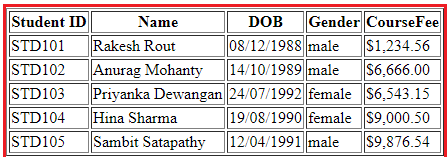
**<td>**{{student.CourseFee | currency:'USD':true}}**</td>**

**</tr>**

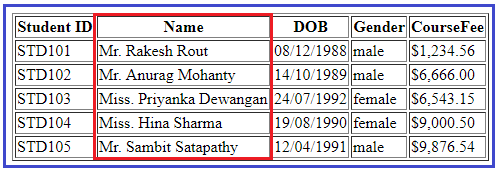
**</tbody>**

**</table>**

At this moment if you run the application, then you will get the following output in the browser.



Now, the requirement changes, now they want to show the title depending on the gender of the student I.e. we need to add Mr. or Miss. prefixed before the name of the student as shown in the below image.

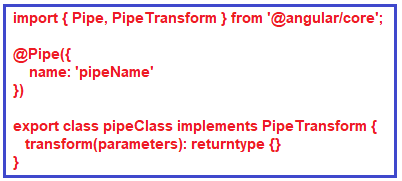


**How we can achieve this?**

We can achieve this very easily by creating an angular custom pipe.

**How to create Angular Custom Pipe?**

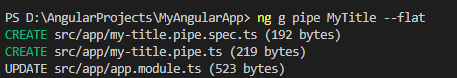
In order to create a custom pipe in angular, you have to apply the @Pipe decorator to a class which you can import from the Angular Core Library. The @Pipe decorator allows you to define the pipe name that you will use within the template expressions. The syntax to create a pipe in angular is given below.



**Note:**The transform method will decide the input types, the number of parameters and the output type.

**Creating Angular Custom Pipe using Angular CLI:**

Let say we want to create a custom pipe with the name MyTitle. In order to create a custom MyTitle pipe open a new terminal and type **ng g pipe MyTitle –flat** and press enter as shown in the below image.



Once you type **ng g pipe MyTitle –flat** and press enter, it will take some time and create two files (**my-title.pipe.ts** and **my-title.pipe.spec.ts**) within the app folder. Along the way, it also update the app.module.ts file.

**Modify my-title.pipe.ts file:**

Now, open my-title.pipe.ts file and then copy and paste the following code in it.

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

@Pipe**({**

name: 'myTitle'

**})**

**export** **class** MyTitlePipe implements PipeTransform **{**

transform**(**name: **string**, gender: **string)**: **string** **{**

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

**return** "Mr. " + name;

**else**

**return** "Miss. " + name;

**}**

**}**

**Understanding above code:**

1. First, we import the **Pipe**decorator and **PipeTransform**interface from the Angular core Library.
2. Then we decorated the “**MyTitlePipe**” class with the **Pipe**decorator so that this class will become an Angular pipe.
3. We then set the name property of the pipe decorator to **myTitle**so that we can use this name (myTitle) on any HTML page where we want this pipe functionality.
4. The **MyTitlePipe**class implements the **PipeTransform**interface and that interface has one method called **transform**() and here we implement that method.
5. As you can see in the above code, the **transform**method takes 2 parameters (name and gender). The name parameter will receive the name of the student whereas the gender parameter will receive the gender of the student. The method returns a string i.e. Mr. or Miss. prefixed to the name of the student depending on their gender.

**Registering the Custom Pipe in Angular Application:**

Before using the custom MyTitlePipe, first we need to register it in the app.module.ts file. If you are creating it using Angular CLI, then the angular framework will automatically register the pipe. To make sure, let us modify the app.module.ts file as shown below. Here, first we need to import the MyTitlePipe and then we need to include it in the “declarations” array of NgModule decorator.

**import** **{** BrowserModule **}** from '@angular/platform-browser';

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

**import** **{** FormsModule **}** from '@angular/forms';

**import** **{** AppRoutingModule **}** from './app-routing.module';

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

**import** **{** MyTitlePipe **}** from './my-title.pipe';

@NgModule**({**

declarations: **[**

AppComponent,

MyTitlePipe,

**]**,

imports: **[**

BrowserModule,

AppRoutingModule,

FormsModule

**]**,

providers: **[]**,

bootstrap: **[**AppComponent**]**

**})**

**export** **class** AppModule **{** **}**

**Using the Custom Pipe in Angular Application:**

Modify the **app.component.html** file as shown below. Notice that we are passing the student gender as an argument for the gender parameter to our custom pipe. The Student name gets passed automatically.

**<table** border="1"**>**

**<thead>**

**<tr>**

**<th>**Student ID**</th>**

**<th>**Name**</th>**

**<th>**DOB**</th>**

**<th>**Gender**</th>**

**<th>**CourseFee**</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr** \*ngFor='let student of students'**>**

**<td>**{{student.ID | uppercase}}**</td>**

**<td>**{{student.Name | myTitle:student.Gender}}**</td>**

**<td>**{{student.DOB | date:'dd/MM/yyyy'}}**</td>**

**<td>**{{student.Gender | lowercase}}**</td>**

**<td>**{{student.CourseFee | currency:'USD':true}}**</td>**

**</tr>**

**</tbody>**

**</table>**

With the above changes in place, now run the application and you should the output as expected.

# **Passing Parameters to Angular Pipes**

A pipe can also have optional parameters to change the output. To pass parameters, after a pipe name add a colon( : ) followed by the parameter value.

**Syntax**:

1. pipename : parametervalue

A pipe can also have multiple parameters as shown below

1. pipename : parametervalue1:parametervalue2

Below are the built-in pipes present in Angular, which accept optional parameters using which the pipe's output can be fine-tuned.

**currency**

This pipe displays a currency symbol before the expression. By default, it displays the currency symbol $

**Syntax**:

1. {{ expression | currency:currencyCode:symbol:digitInfo:locale }}

**currencyCode** is the code to display such as INR for the rupee, EUR for the euro, etc.

**symbol** is a Boolean value that represents whether to display currency symbol or code.

* **code**: displays code instead of a symbol such as USD, EUR, etc.
* **symbol** (default): displays symbol such as $ etc.
* **symbol-narrow**: displays the narrow symbol of currency. Some countries have two symbols for their currency, regular and narrow. For example, the Canadian Dollar CAD has the symbol as CA$ and symbol-narrow as $.

**digitInfo** is a string in the following format

{minIntegerDigits}.{minFractionDigits} - {maxFractionDigits}

* minIntegerDigits is the minimum integer digits to display. The default value is 1
* minFractionDigits is the minimum number of digits to display after the fraction. The default value is 0
* maxFractionDigits is the maximum number of digits to display after the fraction. The default value is 3

**locale**is used to set the format followed by a country/language. To use a locale,  the locale needs to be registered in the root module.

For Example,to set locale to French (fr), add the below statements in app.module.ts

1. import { registerLocaleData } from '@angular/common';
2. import localeFrench from '@angular/common/locales/fr';
3. registerLocaleData(localeFrench);

**Examples**:

1. {{ 25000 | currency }} will display $25,000.00
2. {{ 25000 | currency:'CAD' }} will display CA$25,000.00
3. {{ 25000 | currency:'CAD':'code' }} will display CAD25,000.00
4. {{ 25000 | currency:'CAD':'symbol':'6.2-3'}} will display CA$025,000.00
5. {{ 25000 | currency:'CAD': 'symbol-narrow':'1.3'}} will display $25,000.000
6. {{ 250000 | currency:'CAD':'symbol':'6.3'}} will display CA$250,000.000
7. {{ 250000 | currency:'CAD':'symbol':'6.3':'fr'}} will display 250 000,000 CA$

**date**

This pipe can be used to display the date in the required format

**Syntax**:

1. {{ expression | date:format:timezone:locale }}

An**expression**is a date or number in milliseconds

The**format**indicates in which form the date/time should be displayed. Following are the pre-defined options for it.

* 'medium' :equivalent to 'MMM d, y, h:mm:ss a' (e.g. Jan 31, 2018, 11:05:04 AM)
* 'short': equivalent to 'M/d/yy, h:mm a' (e.g. 1/31/2018, 11:05 AM)
* 'long': equivalent to 'MMMM d, y, h:mm:ss a z' (e.g. January 31, 2018 at 11:05:04 AM GMT+5)
* 'full': equivalent to 'EEEE, MMMM d, y, h:mm:ss a zzzz' (e.g. Wednesday, January 31, 2018 at 11:05:04 AM GMT+05:30)
* 'fullDate' : equivalent to 'EEEE, MMMM d, y' (e.g. Wednesday, January 31, 2018)
* 'longDate' : equivalent to 'MMMM d, y' (e.g. January 31, 2018)
* 'mediumDate' : equivalent to 'MMM d, y' (e.g. Jan 31, 2018)
* 'shortDate' : equivalent to 'M/d/yy' (e.g. 1/31/18)
* 'mediumTime' : equivalent to 'h:mm:ss a' (e.g. 11:05:04 AM)
* 'shortTime' :  equivalent to 'h:mm a' (e.g. 11:05 AM)
* 'longTime': equivalent to 'h:mm a' (e.g. 11:05:04 AM GMT+5)
* 'fullTime': equivalent to 'h:mm:ss a zzzz' (e.g. 11:05:04 AM GMT+05:30)

**Timezone** to be used for formatting. For example, ’+0430’ (4 hours, 30 minutes east of the Greenwich meridian) If not specified, the local system timezone of the end-user's browser will be used.

**locale**is used to set the format followed by a country/language. To use a locale, register the locale in the root module.

For Example, to set locale to French (fr), add the below statements in app.module.ts

1. import { registerLocaleData } from '@angular/common';
2. import localeFrench from '@angular/common/locales/fr';
3. registerLocaleData(localeFrench);

**Examples**:

1. {{ "6/2/2017" | date }} will display Jun 2, 2017
2. {{ "6/2/2017, 11:30:45 AM" | date:'medium' }} will display Jun 2, 2017, 11:30:45 AM
3. {{ "6/2/2017, 11:30:45 AM" | date:'mmss' }} will display 3045
4. {{"1/31/2018, 11:05:04 AM" | date:'fullDate':'0':'fr'}} will display mercredi 31 janvier 2018
5. {{ 90000000 | date }} will display Jan 2, 1970 – date pipe will start from Jan 1, 1970 and based on the given number of milliseconds, it displays the date

**percent**

This pipe can be used to display the number as a percentage

**Syntax**:

1. {{ expression | percent:digitInfo:locale }}

**digitInfo** is a string in the following format

{minIntegerDigits}.{minFractionDigits} - {maxFractionDigits}

* minIntegerDigits is the minimum integer digits to display. The default value is 1
* minFractionDigits is the minimum number of digits to display after the fraction. The default value is 0.
* maxFractionDigits is the maximum number of digits to display after the fraction. The default value is 3.

**locale**is used to set the format followed by a country/language. To use a locale,  register the locale in the root module.

For Example, to set locale to French (fr), add the below statements in app.module.ts

1. import { registerLocaleData } from '@angular/common';
2. import localeFrench from '@angular/common/locales/fr';
3. registerLocaleData(localeFrench);

**Examples**:

1. {{ 0.1 | percent }} will display 10%
2. {{ 0.1 | percent:'2.2-3' }} will display 10.00%
3. {{ 0.1 | percent:'2.2-3': 'fr' }} will display 10.00 %

**slice**

This pipe can be used to extract a subset of elements or characters from an array or string respectively.

**Syntax**:

1. {{ expression | slice:start:end }}

The **expression**can be an array or string

**start**represents the starting position in an array or string to extract items. It can be a

* positive integer which will extract from the given position till the end
* negative integer which will extract the given number of items from the end

**end** represents the ending position in an array or string for extracting items. It can be

* positive number that returns all items before the end index
* negative number which returns all items before the end index from the end of the array or string

**Examples**:

1. {{ ['a','b','c','d']| slice:2}} will display c,d
2. {{ ['a','b','c','d']| slice:1:3}} will display b,c
3. {{ 'Laptop Charger'| slice:3:6}} will display top
4. {{ 'Laptop Charger'| slice:-4}} will display rger
5. {{ 'Laptop Charger'| slice:-4:-2}} will display rg

**number**

This pipe can be used to format a number.

**Syntax**:

1. {{ expression | number:digitInfo }}

The **expression**should be numeric

**digitInfo** is a string in the following format

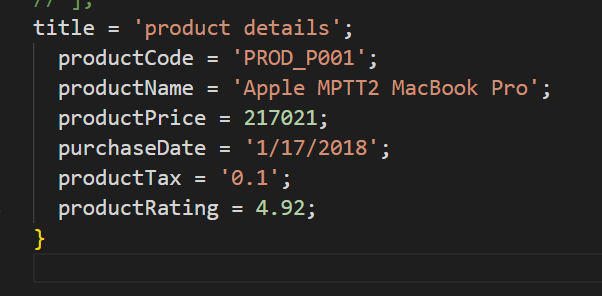
{minIntegerDigits}.{minFractionDigits} - {maxFractionDigits}

* minIntegerDigits is the minimum integer digits to display. The default value is 1.
* minFractionDigits is the minimum number of digits to display after the fraction. The default value is 0.
* maxFractionDigits is the maximum number of digits to display after fraction. The default value is 3.

**Example**:

1. {{ 25000 | number }} will display 25,000
2. {{ 25000 | number:'.3-5' }} will display 25,000.000

Exercise:



Display Product Details shown in the below image

