labGuide for built in Pipes

Pipes are a great feature in an Angular that transforms the data for display with a variety of formats and options. Angular provides some built-in pipes to transform the data for display. Here let us see how to use that built-in pipes and how to create our custom pipes in an angular application.

Built-in Pipes of Angular

* DatePipe
* UpperCasePipe
* LowerCasePipe
* DecimalPipe
* PercentPipe
* CurrencyPipe

Use DatePipe in Angular Application

**DatePipe**formats the date value based on the locale rules. The input values of **DatePipe** are string, number, or Date object. The syntax of DatePipe is

{{value\_expression | date [ : format [ : timezone [ : locale ] ] ] }}

**value\_expression** is an input value that is of type string, number or Date object.

The parameters of **DatePipe** are format, timezone, and locale.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Default Value** |
| Format | string | 'mediumDate' (Optional) |
| Timezone | string | undefined (Optional) |
| Locale | string | undefined (Optional) |

Example

**app.component.html**

<**div**>{{ datestring | date }}</**div**>

<**div**>{{ currentDate | date:"MM/dd/yy" }} </**div**>

<**div**>{{currenttime | date:"short"}}</**div**>

**app.component.ts**

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

@Component({

selector: 'app-root',

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

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

})

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

//Input Values of Date Pipe

//String

datestring = "21 SEP 2021";

//Date object

currentDate = new Date();

//Number

currenttime = Date.now(); //retruns current date in milliseconds

}

**Output**

Sep 21, 2021

09/07/21

9/7/21, 8:39 PM

In the above example, We used simply the DatePipe like '**date**' and the DatePipe with the formats like a **date: "MM/dd/yy"** and **date: "short"**.

By default, the format of **DatePipe**is 'mediumDate'. So if we use DatePipe without any format then it displays the date value with the default format 'mediumDate'.

Angular provides some of the predefined format options of **DatePipe**that are

|  |  |  |
| --- | --- | --- |
| **Format Options** | **Equivalent to** | **Examples (Using en-IN locale)** |
| 'short' | 'M/d/yy, h:mm a' | 9/7/21, 8:59 PM |
| 'medium' | 'MMM d, y, h:mm:ss a' | Sep 7, 2021, 9:00:05 PM |
| 'long' | 'MMMM d, y, h:mm:ss a z' | September 7, 2021 at 9:02:06 PM GMT+5 |
| 'full' | 'EEEE, MMMM d, y, h:mm:ss a zzzz' | Tuesday, September 7, 2021 at 9:02:56 PM GMT+05:30 |
| 'shortDate' | 'M/d/yy' | 9/7/21 |
| 'mediumDate' | 'MMM d, y' | Sep 7, 2021 |
| 'longDate' | 'MMMM d, y' | September 7, 2021 |
| 'fullDate' | 'EEEE, MMMM d, y' | Tuesday, September 7, 2021 |
| 'shortTime' | 'h:mm a' | 9:04 PM |
| 'mediumTime' | 'h:mm:ss a' | 9:05:11 PM |
| 'longTime' | 'h:mm:ss a z' | 9:05:26 PM GMT+5 |
| 'fullTime' | 'h:mm:ss a zzzz' | 9:05:42 PM GMT+05:30 |

Use UpperCasePipe and LowerCasePipe in Angular

**UpperCasePipe**transforms the text to all upper cases likewise the **LowerCasePipe**transforms the text to all lower cases.

Example for both

**app.component.html**

<**div**>{{website | uppercase}}</**div**>

<**div**>{{website | lowercase}}</**div**>

**app.component.ts**

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

@Component({

selector: 'app-root',

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

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

})

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

website = "TipsToCode!."

}

output

TIPSTOCODE!.

tipstocode!.

Use DecimalPipe and PercentPipe in Angular

**DecimalPipe**transforms the number into a decimal point string based on the specification of locale rules. The syntax for the DecimalPipe is

{{ value\_expression | number [ : digitsInfo [ : locale ] ] }}

**value\_expression** is an input value that is of type string or number. The parameters of DecimalPipe are digitsinfo and locale.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Usage** |
| Digitsinfo | string | It describes the digits and decimal points |
| Locale | string | It species the locale format rules. |

**digitsinfo** parameter specifies the decimal point representation to the number. The format of digitsinfo paremter is

**{minIntegerDigits}.{minFractionDigits}-{maxFractionDigits}**

|  |  |
| --- | --- |
| **minIntegerDigits** | It specifies the minimum number of integer digits before the decimal point. Default is 1. |
| **minFractionDigits** | It specifies the minimum number of integer digits after the decimal point. Default is 0. |
| **maxFractionDigits** | It specifies the maximum number of digits after the decimal point. Default is 3. |

Example

app.component.html

<!-- Input:5.5625 => Output:5.563 -->

<**div**>{{amount | number}}</**div**>

<!-- Input:5.5625 => Output:6 -->

<**div**>{{amount | number : '0.0-0'}} (or) {{amount | number : '1.0-0'}}</**div**>

<!-- Input:5.5625 => Output:006 -->

<**div**>{{amount | number : '3.0-0'}}</**div**>

<!-- Input:5.5625 => Output:5.6 -->

<**div**>{{amount | number : '1.1-1'}}</**div**>

<!-- Input:5.5625 => Output:5.56 -->

<**div**>{{amount | number : '1.1-2'}}</**div**>

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

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

amount = 5.5625;

}

output

5.563

6 (or) 6

006

5.6

5.56

**PercentPipe**

**PercentPipe**transforms the number to a percentage string based on the specification of locale rules. The syntax of PercentPipe is

{{ value\_expression | percent [ : digitsInfo [ : locale ] ] }}

The input values and the parameters of **PercentPipe**are the same as the DecimalPipe. So refer to the same.

Let us see the example below. Generally, how do we find the percentage of the number?. Right!. multiply the number by 100 produces the percentage of the number.

app.component.html

<**div**>{{amount | percent}}</**div**>

<**div**>{{amount | percent : '1.1-2'}}</**div**>

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

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

amount = 0.4245;

}

output

42%

42.45%

Use CurrrencyPipe in Angular

CurrencyPipe transforms a number into a currency string based on the locale rules. These locale rules determine group sizing and separator, decimal-point character, and other locale configurations.

The syntax of **CurrencyPipe**is

{{ value\_expression | currency [ : currencyCode [ : display [ : digitsInfo [ : locale ] ] ] ] }}

The input value of **CurrencyPipe**is of type string or number. The parameters are currencyCode, display, digitsInfo, and locale

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Usage** |
| currencyCode | string | Like USD for dollar, INR for Rupee |
| display | string / boolean | Displays the currency indicator with the specified format. **code**: Displays the code like USD for dollar. **symbol:**Displays the symbol like $. **symbol-narrow**: It uses the narrow symbol for locales. Example: The country Canada has two symbols for their currency like CA$ and $. Here CA$ is the symbol and $ is the symbol-narrow. **Note**: if the locale has no narrow symbol, uses the standard symbol for the locale. **Default Value**: 'symbol' that is $. |
| digitsInfo | string | It specifies the decimal point representation of the string. The format is {minIntegerDigits}.{minFractionDigits}-{maxFractionDigits} |
| locale | string | It specifies the locale code for the locale format rules to use. |

**Examples**.

app.component.html

<**h3**>Currency Pipe Examples</**h3**>

<**div**>{{amount | currency}} (Use default Symbol $)</**div**>

<**div**>Canadian Dollor: {{amount | currency : 'CAD'}}</**div**>

<**div**>Euro: {{amount | currency : 'EUR'}}</**div**>

<**div**>Indian Rupee: {{amount | currency : 'INR'}}</**div**>

<**div**>US Dollar: {{amount | currency : 'USD': 'symbol':'4.2-2'}}</**div**>

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

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

amount = 150;

}

output

Currency Pipe Examples

$150.00 (Use default Symbol $)

Canadian Dollor: CA$150.00

Euro: €150.00

Indian Rupee: ₹150.00

US Dollar: $0,150.00

I hope you understood how to use the built-in pipes like DatePipe, UpperCasePipe, LowerCasePipe, DecimalPipe, PercentPipe, and CurrencyPipe. Now let us see how to create our custom pipes in the Angular Application.

Custom Pipe in Angular Application

Here let me explain how to create and use custom pipe in an angular application.

1. Custom Pipe with no parameter.
2. Custom Pipe with parameter.

**Custom Pipe without parameter**

The below example uses the custom pipe without parameters. This custom pipe calculates the grade for the student based on their score and we use that pipe to display the grade.

First, let us create the custom pipe by using the below command.

**ng g pipe gradepipe**

The above command creates the custom pipe named **GradepipePipe** under the app folder. Now let us open the **gradepipe.pipe.ts** file and replace it with the below code.

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

@Pipe({

name: 'gradepipe'

})

**export** **class** GradepipePipe implements PipeTransform {

transform(score: **any**): **string** {

**if**(score>=90)

**return** "A";

**else** **if**(score>=70&&score<90)

**return** "B";

**else**

**return** "C";

}

}

The transform function in the above snippet calculates and returns the grade for the students based on their scores.

app.component.html

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

<**thead**>

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

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

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

<**th**>Score</**th**>

<**th**>Grade</**th**>

</**thead**>

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

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

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

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

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

<**td**>{{ student.score | gradepipe}}</**td**>

</**tr**>

</**table**>

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

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

students= [

{

id: 1,

name: 'Salman',

gender: 'MALE',

score: 80

},

{

id: 2,

name: 'John',

gender: 'MALE',

score: 90

},

{

id: 3,

name: 'Fazil',

gender: 'MALE',

score: 70

},

{

id: 4,

name: 'Sharmi',

gender: 'FEMALE',

score: 90

},

{

id: 5,

name: 'Kajal',

gender: 'FEMALE',

score: 50

}

]

}

output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Gender** | **Score** | **Grade** |
| 1 | Salman | MALE | 80 | B |
| 2 | John | MALE | 90 | A |
| 3 | Fazil | MALE | 70 | B |
| 4 | Sharmi | FEMALE | 90 | A |
| 5 | Kajal | FEMALE | 50 | C |

**Custom Pipe with parameter**

The below example uses the custom pipe with parameters. This custom pipe generates the honorific like Mr or Miss for the students based on their gender.

Let us create the custom pipe by using the below command.

**ng g pipe honorificPipe**

The above command creates the custom pipe named **HonorificPipePipe** under the app folder. Now let us open the **honorific-pipe.pipe.ts** file and replace it with the below code.

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

@Pipe({

name: 'honorificPipe'

})

**export** **class** HonorificPipePipe implements PipeTransform {

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

**if**(gender == "MALE"){

**return** 'Mr. '+name;

}

**return** 'Miss. '+name;

}

}

The transform function in the above snippet generates the honorific for the students based on their gender and it returns the name appends with the corresponding honorific.

app.component.html

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

<**thead**>

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

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

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

<**th**>Score</**th**>

<**th**>Grade</**th**>

</**thead**>

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

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

<**td**>{{ student.name | honorificPipe:student.gender}}</**td**>

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

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

<**td**>{{ student.score | gradepipe}}</**td**>

</**tr**>

</**table**>

output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Name** | **Gender** | **Score** | **Grade** |
| 1 | Mr. Salman | MALE | 80 | B |
| 2 | Mr. John | MALE | 90 | A |
| 3 | Mr. Fazil | MALE | 70 | B |
| 4 | Miss. Sharmi | FEMALE | 90 | A |
| 5 | Miss. Kajal | FEMALE | 50 | C |

I hope you understood the built-in and custom pipes in an Angular Application. Thanks!. Keep Reading!.

# Mask credit card and mobile number using pipe in Angular

how to mask the credit/debit card number and mobile number and display it on the web pages in an angular application.

First, let us know how to use the methods **slice**and **replace**with string values. Because slice and replace will be used in credit/debit card number masking. So let us recall first.

## Slice and Replace method in Angular

**Slice method**

The Slice method is used to extract some parts of the string from the original string and return the extracted parts in a new string. The syntax is given below.

**slice(start, end)**

The parameter start describes the position where to start the extraction and the parameter end describes the position where to end the extraction but the extraction does not include the character in that position.

The position of a first character is 0 and a second character is 1 and so on.

Let us see some examples

Perform slice with this string value (**let siteName="TipsToCode!"**)

siteName.slice(0) - Extract the entire string. The output is **TipsToCode!**.

siteName.slice(1,4)- Extract the string from the position 1 and upto the position 4. The output is **ips**.

siteName.slice(-1) - Extract the last character (or) first character from the reverse of the string. The output is **!**.

**Replace method**

Replace method search the replaceable string in the original string and replace that if it finds. It returns the replaced string in a new string.

We can do the replacement based on the string value or an expression. Let us see some examples.

Perform replace with this string value (**let siteName="Welcome to TipsToCode! and Welcome to Programming world!."**)

siteName.replace("Welcome", "Hello"); - The output is "Hello to TipsToCode! and Welcome to Programming world!.".

In the above string, the word Welcome is replaced by Hello. It replaced the first matching word, not all the matching words. If you want to replace all the matching words then you have to use the global modifier (g).

siteName.replace(/Welcome/g, "Hello"); - The output is "Hello to TipsToCode! and Hello to Programming world!.".

I hope you understood the usage of slice and replace methods in angular. Now let us see how to create the mask pipe for credit/debit card numbers and mobile numbers.

## Mask Credit/Debit card number and display it on the page by using Pipe in Angular

Let us create the pipe named '**credit-debit-mask-pipe**' by using the below command

ng g pipe **credit-debit-mask-pipe**

The above command created the two files under the app folder named

'**credit-debit-mask-pipe**.**pipe.spec.ts**' and '**credit-debit-mask-pipe**.**pipe.ts**'. It registered the pipe in the app.module.ts file like below.

app.module.ts

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

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

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

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

**import** { CreditDebitMaskPipePipe } from './credit-debit-mask-pipe.pipe';

@NgModule({

declarations: [

AppComponent,

CreditDebitMaskPipePipe

],

imports: [

BrowserModule,

AppRoutingModule

],

providers: [],

bootstrap: [AppComponent]

})

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

Now let us write the logic to show only the last given digits of credit/debit card number and mask the remaining numbers

Do the changes like below in the corresponding files.

credit-debit-mask-pipe.pipe.ts

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

@Pipe({

name: 'creditDebitMaskPipe'

})

**export** **class** CreditDebitMaskPipePipe implements PipeTransform {

transform(cardNumber: **string**, visibleDigits: **number**): **string** {

//show number of digits at last based on input

**let** maskedNumbers = cardNumber.slice(0, -visibleDigits);

**let** visibleNumbers = cardNumber.slice(-visibleDigits);

**return** maskedNumbers.replace(/./g, '\*') + visibleNumbers;

}

}

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

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

cardNumber = "1234567890123456";

}

app.component.html

<**h1** style="text-align: center">Welcome to TipsToCode Website!.</**h1**>

<**p**>Credit Card Number: {{ cardNumber | creditDebitMaskPipe:4 }}</**p**>

<**router-outlet**></**router-outlet**>

Now save all the changes and run the application. The last 4 digits of the credit/debit card number are visible and others are masked like below.

Credit Card Number: \*\*\*\*\*\*\*\*\*\*\*\*3456

If you want to show first given digits of credit/debit card number and mask the other numbers then use the below code.

credit-debit-mask-pipe.pipe.ts

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

@Pipe({

name: 'creditDebitMaskPipe'

})

**export** **class** CreditDebitMaskPipePipe implements PipeTransform {

transform(cardNumber: **string**, visibleDigits: **number**): **string** {

//show number of digits at first based on input

**let** visibleNumbers = cardNumber.slice(0, visibleDigits);

**let** maskedNumbers = cardNumber.slice(visibleDigits);

**return** visibleNumbers + maskedNumbers.replace(/./g, '\*');

}

}

The above code displays the first 4 digits of the credit/debit card number and others are masked like below.

Credit Card Number: 1234\*\*\*\*\*\*\*\*\*\*\*\*

If you want to show only the first and last given digits of credit/debit card number and mask the numbers in between then use the below code.

credit-debit-mask-pipe.pipe.ts

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

@Pipe({

name: 'creditDebitMaskPipe'

})

**export** **class** CreditDebitMaskPipePipe implements PipeTransform {

transform(cardNumber: **string**, visibleDigits: **number**): **string** {

//show first and last digits based on given input

**let** firstVisibleNumbers = cardNumber.slice(0, visibleDigits);

**let** maskedNumbers = cardNumber.slice(visibleDigits, -visibleDigits);

**let** lastVisibleNumbers = cardNumber.slice(-visibleDigits);

**return** firstVisibleNumbers + maskedNumbers.replace(/./g, '\*') + lastVisibleNumbers;

}

}

The above code displays the first and the last 4 digits of the credit/debit card number and the numbers in between are masked like below.

Credit Card Number: 1234\*\*\*\*\*\*\*\*3456

**Note**: Mostly the credit/debit card number should be 16 digits. So we have to pass the valid digits of input in the pipe in the **app.component.html** file like {{ cardNumber | creditDebitMaskPipe:**4** }}. If you pass 16 as pipe input then that does not mask the credit/debit card number.

## Mask mobile number and display it on the page by using Pipe in Angular

The pipe created above for masking the credit card number will be used for masking the phone number too. The only thing is we have to change the credit/debit card number into a 10 digit mobile number in **app.component.ts** file like below.

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

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

cardNumber = "1234567890";

}

app.component.html

<**h1** style="text-align: center">Welcome to TipsToCode Website!.</**h1**>

<**p**>Mobile Number: {{ cardNumber | creditDebitMaskPipe:2 }}</**p**>

<**router-outlet**></**router-outlet**>

Now save all the changes and run the application again. You can see the below result in the browser.

Mobile Number: 12\*\*\*\*\*\*90

Angular provides built-in pipes to transform the data for display with a variety of formats and options. If you want to know that then please take a look at [**Angular built-in Pipes**](https://www.tipstocode.com/framework/how-to-use-built-in-and-custom-pipes-in-angular-application/).

I hope you understood how to mask credit/debit card numbers and mobile numbers using pipe in Angular Applications