

Dealing with Numbers

https://en.wikipedia.org/wiki/ISO_4217

In this lecture, we're going to learn about **two pipes for formatting numbers**.

The **decimal and currency pipes**.

Both can format numeric values. Formatting numbers is a common problem you'll encounter in app development.

Let's start with the **currency pipe**.

The currency pipe will format a number by adding a currency symbol before the value. On top of adding a currency symbol, the currency pipe can format a number with a fixed decimal size. If our value requires decimals, they will be added automatically.

First, we will need to create a property for storing a number.

Open the app component class file:

Inside this class, we will add a property called **cost**, its value will be 2000. This property will hold the cost of some imaginary product.

```
TS app.ts 1, M X
basics > src > app > TS app.ts > ...
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { RouterOutlet } from '@angular/router';
4  import { Post } from '../post/post';
5
6  @Component({
7    selector: 'app-root',
8    imports: [RouterOutlet, Post, CommonModule],
9    templateUrl: './app.html',
10   styleUrls: ['./app.css'],
11 })
12 export class App {
13   protected title = 'basics';
14
15   protected name = 'daniel kandalaft';
16   protected imgURL = 'https://picsum.photos/id/237/500/500';
17   protected currentDate = new Date();
18   protected cost = 2000;
19
20   getName() {
21     return this.name;
22   }
23
24   changeImage(e: KeyboardEvent) {
25     this.imgURL = (e.target as HTMLInputElement).value;
26   }
27
28   logImg(event: string) {
29     console.log(event);
30   }
31 }
32
```

Let's update the app component template to output the price with the correct currency symbol:

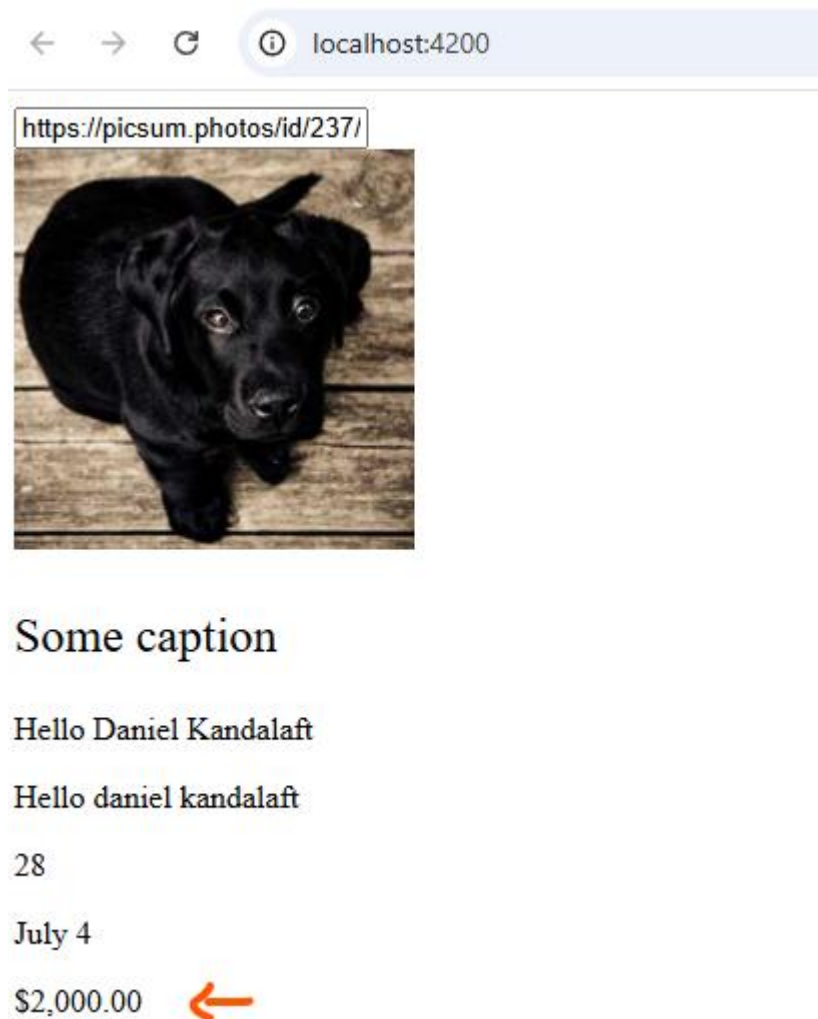
Below the other paragraph tags, we will add a paragraph with an expression, the expression will be the **cost property** with the **currency pipe**.

```

app.html M X
basics > src > app > app.html > p
Go to component
1 <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3 <app-post [img]="imgURL" (imgSelected)="logImg($event)">
4   <p>Some caption</p>
5 </app-post>
6
7 <p>Hello {{ name | titlecase }}</p>
8 <p>Hello {{ getName() }}</p>
9 <p>{{ 15 + 13 }}</p>
10 <p>{{ currentDate | date : "MMMM d" }}</p>
11 <p>{{ cost | currency }}</p>
12

```

Next, let's refresh the page:



The **currency pipe** has transformed the output. It's pre-fixing the number with the **dollar sign symbol, two decimals' values have been appended to the output.** This is the standard formatting for the U.S. currency.

Neither of these symbol nor decimal values were a part of the original value. Angular was able to apply the correct formatting to our number.

We may want to change the currency. The **currency pipe** has a parameter for modifying the **currency.** **We need to provide this pipe with the currency code.**

In the resource section of this lecture, I provide a link to a list of standard currency codes called **ISO 4217**: https://en.wikipedia.org/wiki/ISO_4217

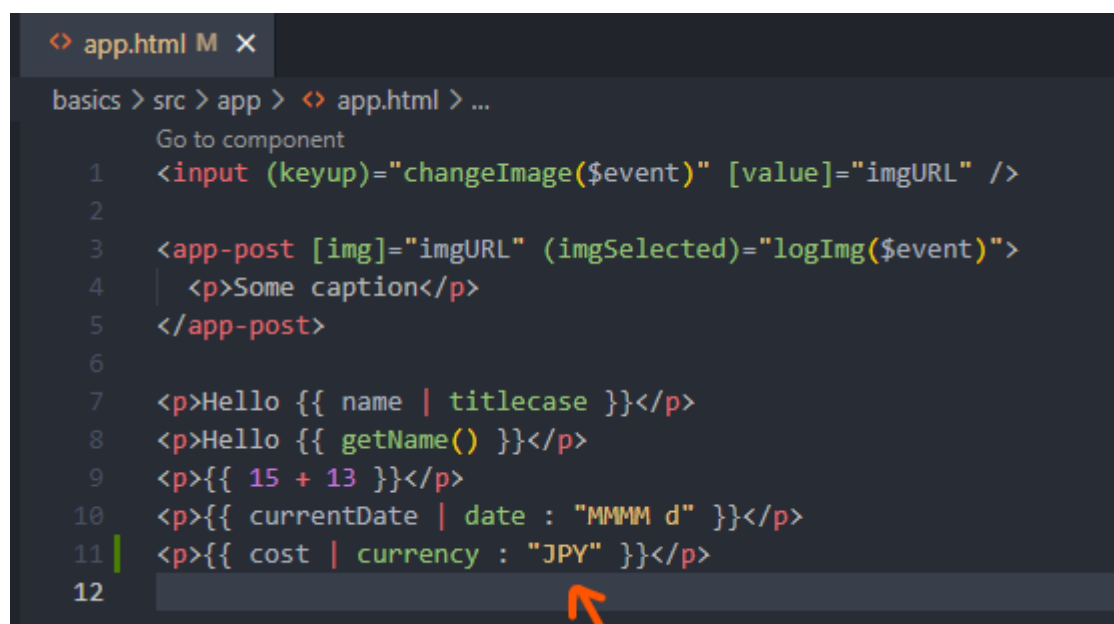
This page contains a comprehensive list of currency codes. Angular supports various currencies by their codes.

For example, let's say we want to change the currency to the Japanese Yen. We can pass in the code to the currency you pay to modify the current currency:

JPY	392	0	Japanese yen	Japan
-----	-----	---	--------------	-------

Back to the editor, we will pass in a string with the currency value code. The currency code for the Japanese Yen is **JPY**, therefore:

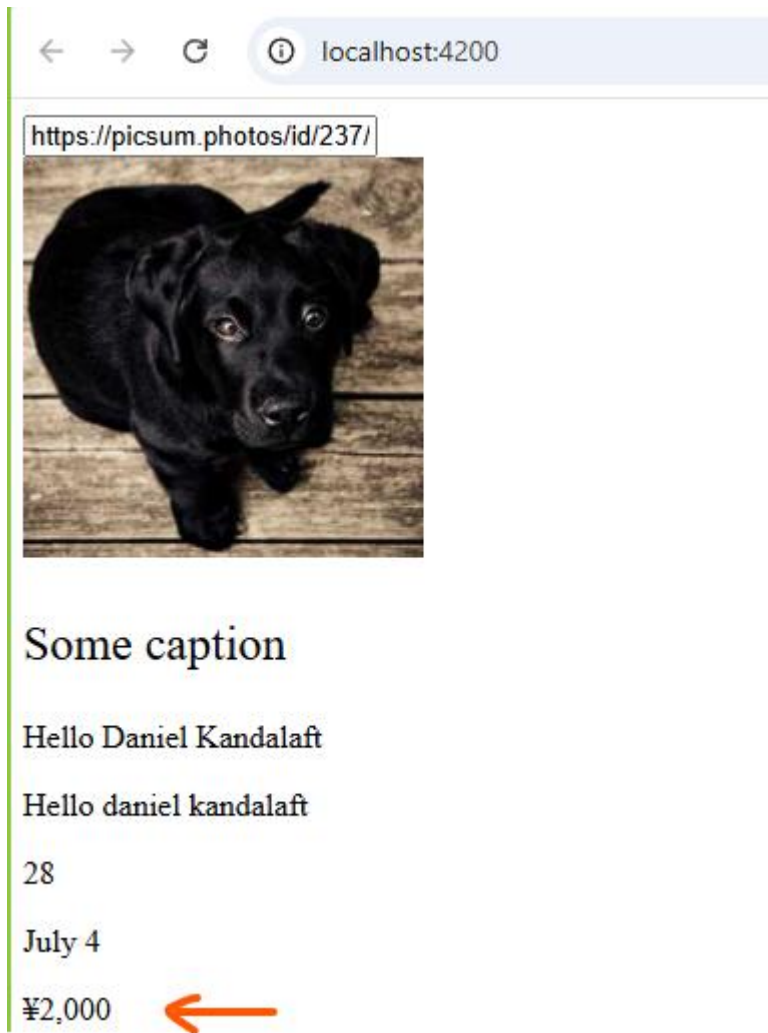
```
<p>{{ cost | currency: 'JPY' }}</p>
```



```

app.html M X
basics > src > app > app.html > ...
Go to component
1 <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3 <app-post [img]="imgURL" (imgSelected)="logImg($event)">
4   <p>Some caption</p>
5 </app-post>
6
7 <p>Hello {{ name | titlecase }}</p>
8 <p>Hello {{ getName() }}</p>
9 <p>{{ 15 + 13 }}</p>
10 <p>{{ currentDate | date : "MMM d" }}</p>
11 <p>{{ cost | currency : "JPY" }}</p>
12
  
```

Switch back to the page in the browser:



The pipe will format our value correctly. There is one thing to keep in mind about this pipe. It will format our value, but it will not convert the value. There is difference between formatting and conversion. Currencies constantly change value. You will need to make sure the currency is properly converted before formatting it.

For example, 2000 USD is not equal to 2000 Yen.

Let's move on to the next pipe.

The other **pipe** is called **decimal**. It functions similarly to the currency pipe. It deals with the **numeric values**. We can use it to format numbers by adding or removing decimal values.

For example, we may want to output the temperature. Calculating the temperature may result in a decimal value. Typically, it's not common to output the temperature with a decimal value. A lot of apps stick to outputting whole numbers. We should strip the decimal values from the temperature if their present.

Switch over to the app component class:

Inside this class, we will add a property called **temperature**, its value will be 25.3.

```

TS app.ts 1, M X
basics > src > app > TS app.ts > ...
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { RouterOutlet } from '@angular/router';
4  import { Post } from './post/post';
5
6  @Component({
7    selector: 'app-root',
8    imports: [RouterOutlet, Post, CommonModule],
9    templateUrl: './app.html',
10   styleUrls: ['./app.css'],
11 })
12 export class App {
13   protected title = 'basics';
14
15   protected name = 'daniel kandalaft';
16   protected imgURL = 'https://picsum.photos/id/237/500/500';
17   protected currentDate = new Date();
18   protected cost = 2000;
19   protected temperature = 25.3;
20
21   getName() {
22     return this.name;
23   }
24
25   changeImage(e: KeyboardEvent) {
26     this.imgURL = (e.target as HTMLInputElement).value;
27   }
28
29   logImg(event: string) {
30     console.log(event);
31   }
32 }
33

```

The .3 portion of the value should be removed from the property during the rendering of the template. This is a good opportunity to apply the decimal type.

Switch over to the template file:

We will add another set of paragraph tags. Inside these tags, we will create an expression for the **temperature** property with the **number pipe**. The name of the **pipe** is completely different from the class name. If you ever look up the documentation for the **pipe**, the official name is called **decimal**. Typically, the class name is the same as the **pipe** name. In this case, the names are not equal. Keep that in mind. Whenever you're working with **pipes**. The class names are not used as the **pipe names**.

By default, applying the **number pipe** to a property will output the number without formatting. It's almost as if you didn't apply the pipe. In most cases, you will want to configure the formatting. The **number pipe** can be configured by passing in a string. The format for the string is a bit funky.

Let's think about the format of a number. Numbers are formatted with the whole numbers and decimals. These portions of a number are separated by a dot. Angular allows us to format both portions separately. The beginning of the format will apply to the whole number portion of a numeric value. It's the minimum number of digits that should appear on the left side of the number. Angular will append zeros to compensate the desired length if the number falls short of the specified length. For example, if we input 5, Angular will prefix our number with three zeros to reach the desired length.

For this demonstration, we are going to set the length to one. If the length exceeds the specified length, Angular will not do anything.

Next, we can add a dot to start configuring the decimal portion of the number to the dot's right. We can set the minimum and maximum number of digits for the decimal value, respectively.

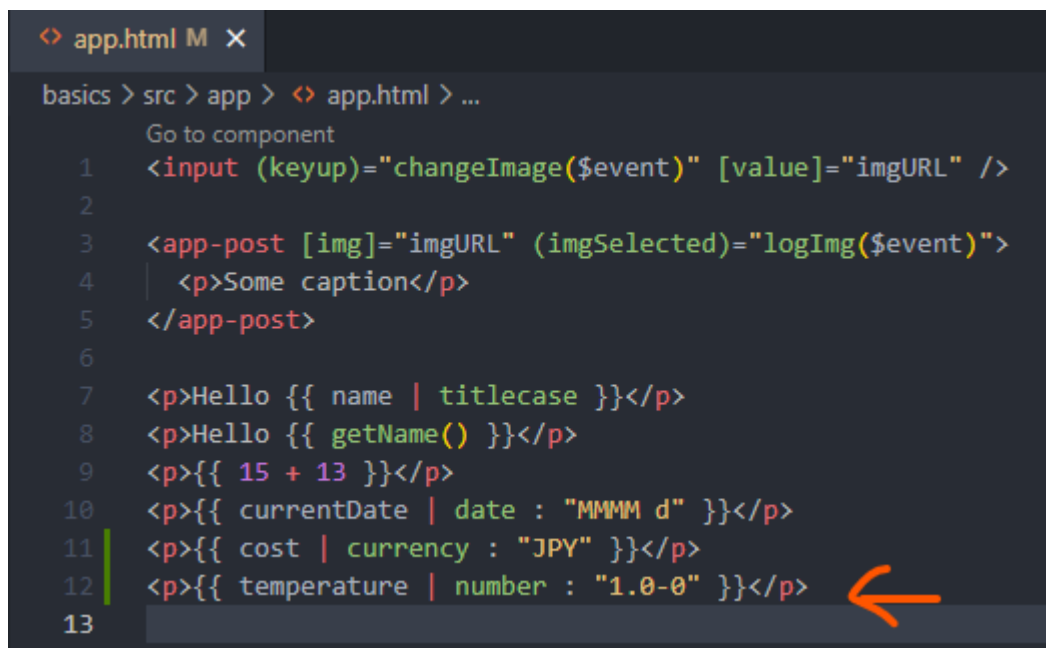
Let's pass in zero zero.

```
<p>{{ temperature | number: '1.0-0' }}</p>
```

The first number [zero] will set the minimum. The second number [zero] will set the maximum.

By setting both numbers to zero, Angular will not allow the value to have any decimal values.

The number pipe will strip them away from the value.



```

app.html M X
basics > src > app > app.html > ...
  Go to component
1  <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3  <app-post [img]="imgURL" (imgSelected)="logImg($event)">
4    <p>Some caption</p>
5  </app-post>
6
7  <p>Hello {{ name | titlecase }}</p>
8  <p>Hello {{ getName() }}</p>
9  <p>{{ 15 + 13 }}</p>
10 <p>{{ currentDate | date : "MMM d" }}</p>
11 <p>{{ cost | currency : "JPY" }}</p>
12 <p>{{ temperature | number : "1.0-0" }}</p>
13

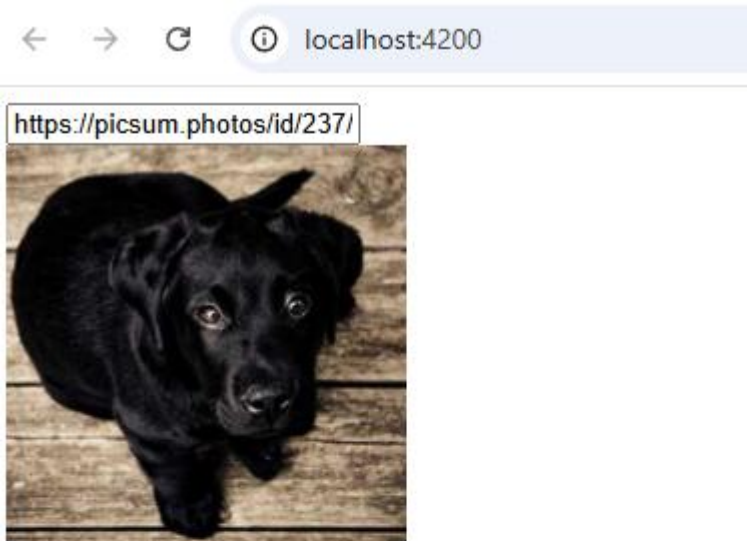
```

Let's say we change the minimum and maximum to two and five:

```
app.html M X
basics > src > app > app.html > ...
Go to component
1 <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3 <app-post [img]="imgURL" (imgSelected)="logImg($event)">
4   <p>Some caption</p>
5 </app-post>
6
7 <p>Hello {{ name | titlecase }}</p>
8 <p>Hello {{ getName() }}</p>
9 <p>{{ 15 + 13 }}</p>
10 <p>{{ currentDate | date : "MMM d" }}</p>
11 <p>{{ cost | currency : "JPY" }}</p>
12 <p>{{ temperature | number : "1.2-5" }}</p>
13
```

This value will tell the number of pipe to output a number with at least two decimal values, if a value has more than five digits, the additional decimal values will be stripped out.

Time to refresh the page:



Some caption

Hello Daniel Kandalaft

Hello daniel kandalaft

28

July 4

¥2,000

25.30

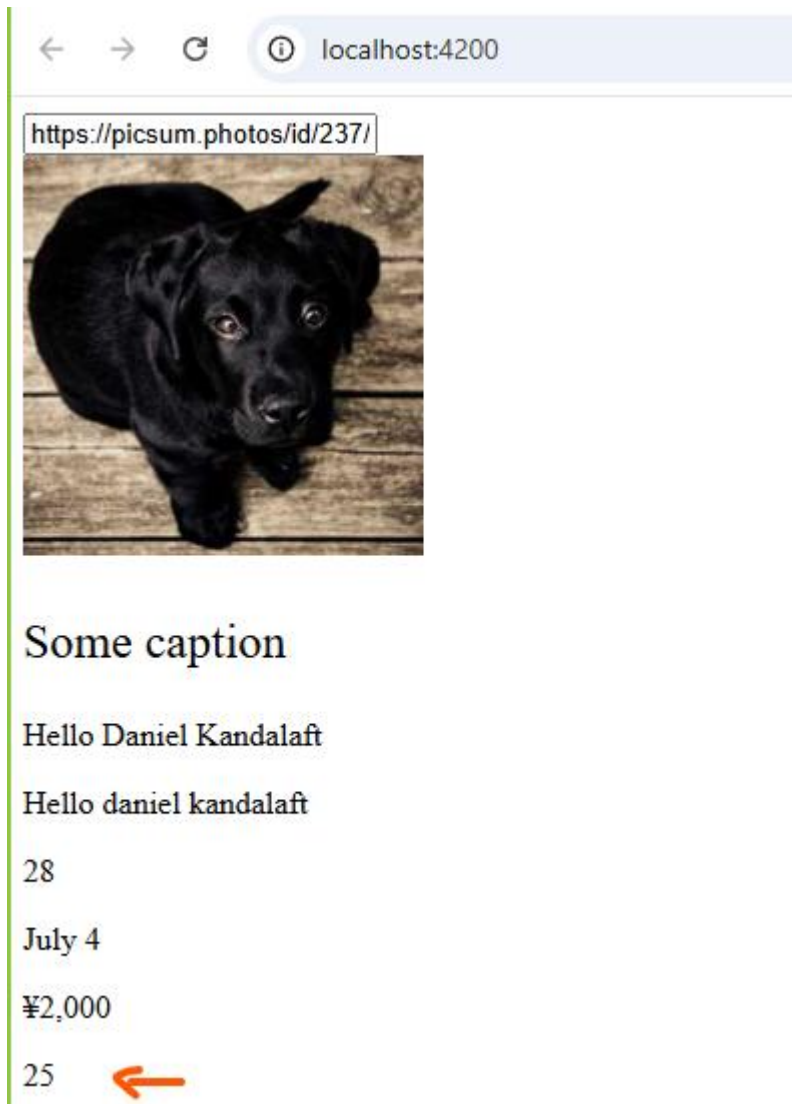


And if we modify it back to :

```

<> app.html M X
basics > src > app > <> app.html > ...
  Go to component
1  <input (keyup)="changeImage($event)" [value]="imgURL" />
2
3  <app-post [img]="imgURL" (imgSelected)="logImg($event)">
4    <p>Some caption</p>
5  </app-post>
6
7  <p>Hello {{ name | titlecase }}</p>
8  <p>Hello {{ getName() }}</p>
9  <p>{{ 15 + 13 }}</p>
10 <p>{{ currentDate | date : "MMM d" }}</p>
11 <p>{{ cost | currency : "JPY" }}</p>
12 <p>{{ temperature | number : "1.0-0" }}</p>
13
  
```





As we wanted, the temperature will display a whole number. The decimal values have been taken away from the value.