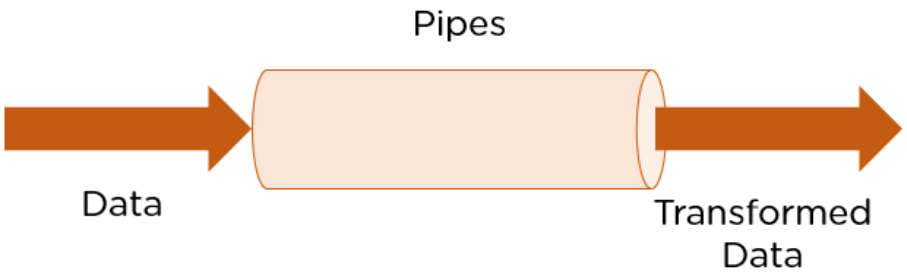
What are Angular Pipes?

Angular Pipes transform the output. You can think of them as makeup rooms where they beautify the data into a more desirable format. They do not alter the data but change how they appear to the user.



Technically, pipes are simple functions designed to accept an input value, process, and return a transformed value as the output. Angular supports many built-in pipes. However, you can also create custom pipes that suit your requirements. Some salient features include:

1. Pipes are defined using the pipe "|" symbol.
2. Pipes can be chained with other pipes.
3. Pipes can be provided with arguments by using the colon (:) sign.

Some commonly used predefined Angular pipes are:

1. DatePipe: Formats a date value.
2. UpperCasePipe: Transforms text to uppercase.
3. LowerCasePipe: Transforms text to lowercase.
4. CurrencyPipe: Transforms a number to the currency string.
5. PercentPipe: Transforms a number to the percentage string.
6. DecimalPipe: Transforms a number into a decimal point string.

Using Built-in Angular Pipes

As mentioned above, Angular provides several built-in pipes to beautify the data being shown on the user interface.

In the following section, we have added the code and screenshots of a few built-in pipes like lowercase, uppercase, and slice.

In the pipes.component.ts file, we've created properties for date and name.

```
import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-pipes',

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

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

})

export class PipesComponent implements OnInit {

  dateToday: string;

  name: string;

  constructor() { }

  ngOnInit(): void {

    this.dateToday = new Date().toDateString();

    this.name = "Simplilearn"

  }

}
```

In Pipes.component.html, we've interpolated the properties and used pipes to format them.

```
<h1>

    Date: {{ dateToday  }} <br>
```

```
    Date Pipe: {{ dateToday | date | uppercase}}<br>

    Date Pipe: {{ dateToday | date: 'short' | lowercase}} <br>

    Name: {{ name | uppercase}} <br>

    Name: {{ name | slice:6}}

</h1>
```

Creating Custom Pipes

Angular makes provision to create custom pipes that convert the data in the format that you desire. Angular Pipes are TypeScript classes with the @Pipe decorator. The decorator has a name property in its metadata that specifies the Pipe and how and where it is used.

Attached below we've added the screenshot of the code that Angular has for the uppercase pipe.

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


@Pipe({
  name: 'custompipe',
  standalone: true
})
export class CustompipePipe implements PipeTransform {


  transform(num:number): unknown {
    return num*num;
  }


}
```

Pipe implements the PipeTransform interface. As the name suggests, it receives the value and transforms it into the desired format with the help of a transform() method.

Here are the general steps to create a custom pipe:

1. Create a TypeScript Class with an export keyword.

2. Decorate it with the @Pipe decorator and pass the name property to it.
3. Implement the pipe transform interface in the class.
4. Implement the transform method imposed due to the interface.
5. Return the transformed data with the pipe.
6. Add this pipe class to the declarations array of the module where you want to use it.

Alternatively, you can use the following command,

ng g pipe <nameofthepipe>

Once run, two files are created.