

# Format data using pipes

## Pipes

- Small lightweight components for formatting data
- Used in templates
- Input: any object (view unfriendly) + parameters
- Output: mostly string values (view friendly), but also used e.g. to slice arrays.

## Using pipes

- Use a | after the name of the variable
- Pass parameters separated by :

```
<p>The date of today is: {{today}}</p>
<!-- The date of today is: Thu Jun 16 2016 14:45:53 GMT+0200 (Romance Daylight
<p>The formatted date of today is: {{today | date}}</p>
<!-- The formatted date of today is: Jun 16, 2016 -->
<p>The formatted date of today is: {{today | date : 'short'}}</p>
<!-- The formatted date of today is: Jun 16, 2016 -->
```

## Default pipes

- DatePipe
- UpperCasePipe
- LowerCasePipe
- CurrencyPipe
- PercentPipe
- JsonPipe
- DecimalPipe
- SlicePipe
- AsyncPipe

## Internationalization

You can configure this API in your module

```
import { NgModule, LOCALE_ID } from '@angular/core';
import { registerLocaleData } from '@angular/common';
import localeNL from '@angular/common/locales/nl';

registerLocaleData(localeNL);

@NgModule({
  providers: [
    { provide: LOCALE_ID, useValue: 'nl-NL' },
    ...
  ],
  ...
})
export class AppModule { }
```

## Some more pipe examples

DecimalPipe

```
<p>decimal: {{2.3454 | number}}</p>
<p>decimal formatted: {{2.3454 | number : '1.0-2'}}</p>
```

CurrencyPipe

```
<p>currency: {{2.3454 | currency}}</p>
<p>currency in 'EUR': {{2.3454 | currency : 'EUR'}}</p>
<p>currency in '€': {{2.3454 | currency : 'EUR' : true}}</p>
<p>currency in '€', formatted: {{2.3 | currency : 'EUR' : true : '1.2-2'}}</p>
```

The format for the last parameter:

```
{minIntegerDigits}.{minFractionDigits}-{maxFractionDigits}
```

## Creating a pipe

Simply implement `PipeTransform`

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

@Pipe({
  name: 'greet'
})
export class GreetPipe implements PipeTransform {
  transform(value: any, args: any[]): any {
    return `Hey ${value}!`
  }
}
```

Register the pipe with the module for use

```
import { GreetPipe } from './greet/index';

@NgModule({
  imports: [...],
  declarations: [..., GreetPipe],
  exports: [..., GreetPipe]
})
export class MyPipesModule { }
```

## Creating a pipe

Import the module where the pipe resides in

```
import { MyPipesModule } from '../my-pipes/my-pipes.module';

@NgModule({
  imports: [..., MyPipesModule],
  declarations: [...],
  exports: [...],
  providers: [...]
})
export class SomeModule { }
```

And use it in the template of a component inside that `SomeModule`

```
<p>{{myName | greet}}</p>
```

## Pure vs impure pipes

@Pipe has a `pure` property:

```
@Pipe({ name: 'my', pure: false })  
export class MyPipe implements PipeTransform {
```

Pure pipes:

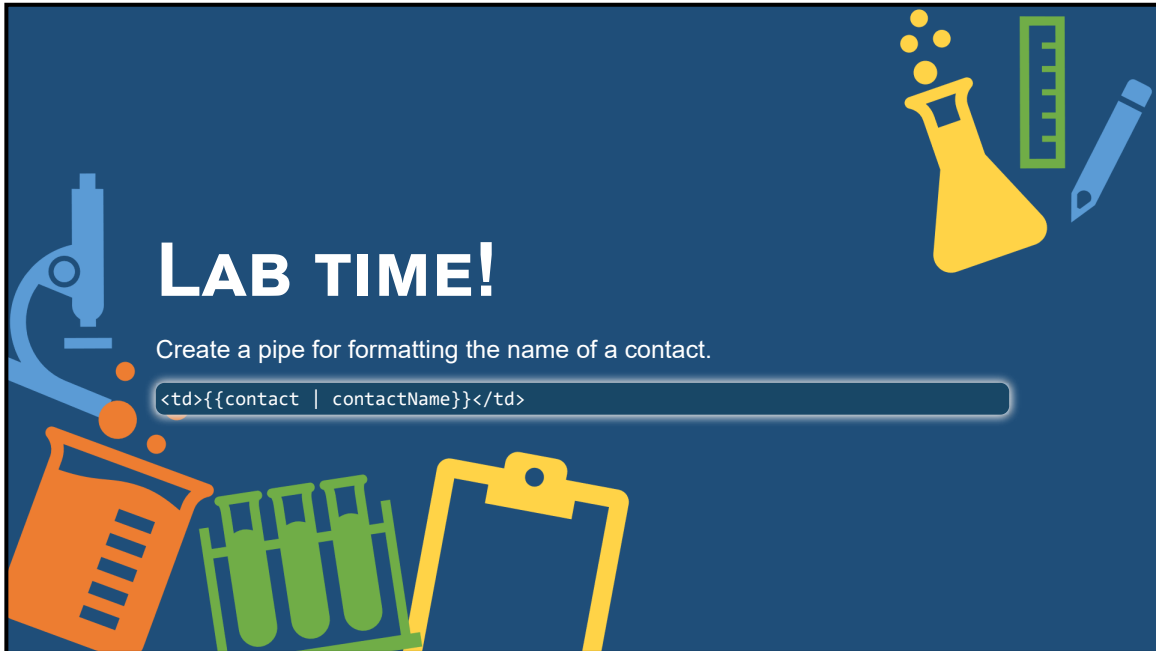
- Are the default, most (custom) pipes are pure
- Get called when the input value has changed

Impure pipes:

- Get called for every time change detection gets triggered
- Can drain performance quickly
- `async` pipe is impure

## Recap

- Creating and using pipes for formatting pipe
- Some pipes use the Intl API for internationalization
- Exporting and importing with modules
- Pure vs impure pipes

A graphic with a dark blue background. In the top right corner, there is a yellow Erlenmeyer flask with three yellow bubbles rising from it, a green graduated cylinder, and a blue pen. In the bottom left corner, there is a blue microscope, an orange beaker with orange liquid, and a rack of three green test tubes. The text "LAB TIME!" is written in large, white, bold, sans-serif capital letters. Below it, in smaller white text, is the instruction "Create a pipe for formatting the name of a contact." At the bottom of the graphic, there is a white rounded rectangular box containing the code snippet "<td>{{contact | contactName}}</td>".

# LAB TIME!

Create a pipe for formatting the name of a contact.

```
<td>{{contact | contactName}}</td>
```