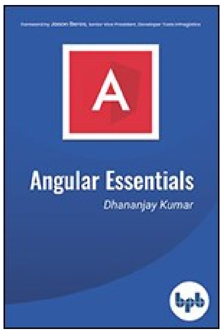

Angular Essentials: The Essential Guide to Learn Angular

by Dhananjay Kumar
BPB Publications. (c) 2019. Copying Prohibited.

Chapter 6: Pipes

In tis chapter, we will learn about Pipes in Angular. Following topics will be covered:

- Pipes
- Built-in pipes
- Custom pipes
- Types of pipes

Pipes

Angular pipes take data as input and transform it to your desired output. Angular pipes are functions, which takes input and transform it to the desired output. Keep in mind that, *it* does not change the underlying data structure of input parameter. You can visualize, Angular pipes as shown in *figure 6.1*.

Input
Pipes
Output
Figure 6.1

You may use pipes in various scenarios such as:

- Displaying data in lowercase or uppercase
- Displaying currency in a particular format such as USD, INR
- Filtering input data array to display desired rows, and so on.

Built-in Pipes

Angular provides many built-in pipes for various transformations such as:

- UpperCasePipe
- LowerCasePipe
- CurrencyPipe
- PercentPipe
- DatePipe

Let us see how you can use built-in pipes. For example, using interpolation you are displaying name of the product, however, you want to transform the product name output in uppercase. You can do this using Angular pipe uppercase as shown *code listing 6.1.*

Code Listing 6.1

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

@Component({
   selector:   'app-root',
   template:   `{{productName  |  uppercase}} `
})
export class AppComponent {
   productName =  'Cricket Bat';
}
```

As an output, **`productName`** will be displayed in uppercase. However, keep in mind that the underlying data is not changed.

Essentially pipes take input and transform it to the desired output. So, pipe works as shown in *figure 6.2*.

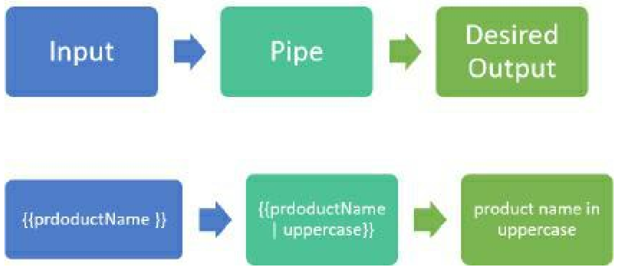

Figure 6.2

Let us see how we could use the built-in currency pipe. Currency pipe can be used as shown in the *code listing 6.2.*

Code Listing 6.2

```
import  {  Component,  Onlnit  }  from '@angular/core';
@Component({
      selector:   'app-root',
   template:  `{{productName | uppercase}} = {{productPrice
|  currency}} `
})
export class AppComponent {
   productName =  'Cricket Bat';
productPrice = 990;
}
```

You can also pass parameters to a pipe using a colon. You can pass input to currency pipe as shown in *code listing 6.3.*

Code Listing 6.3

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

@Component({
      selector:   'app-root',
   template: "{{productName | uppercase}} = {{productPrice
|  currency:'CAD':'symbol-narrow':'4.2-2'}}"
})
export class AppComponent {
      productName =  'Cricket Bat';
      productPrice = 990;
}
```

Custom Pipes

Even though Angular provides many default pipes, there could be requirements where you would need custom pipes. Creating a custom pipe is as simple as creating a function. Let us say that we want to create a pipe, which will capitalize first letter of each words in a string.

Consider component as shown *code listing 6.4.*

Code Listing 6.4

```
import  {  Component,  Onlnit  }  from '@angular/core';
@Component({
      selector:   'app-root',
      template:  '
   <ul *ngFor='let n of names'>
      <li>{{n.name}}</li>
   </ul>
        `
})
export class AppComponent {
      names =   [];
      constructor()   {
         this.names = this.getNames();
}
getNames()   {
      return   [
         {   'name':   'dhananjay Kumar'   },
         {   'name':   'jason beres'   },
         {   'name':   'adam jafe'   }
      ];
   }
}
```

As output, you will get names printed. Now we have requirement that in output, capitalize first character of each names. For that, we must create a custom pipe.

To create a custom pipe, you need to follow these steps:

1. Create a class.
2. Implement **`PipeTransform`** in the class.
3. Implement transform function.

Therefore, you can create a pipe to capitalize first character as shown in the *code listing 6.5.*

Code Listing 6.5

```
import  {  Pipe,  PipeTransform }  from '@angular/core';
@Pipe({ name:   'firstcharcateruppercase'   })
export      class      FirstCharacterUpperCase      implements
PipeTransform {
   transform(value:  string,  args:  string[]):  any {
      if  (!value)   {
         return value;
      }
      return value.replace(/\w\S*/g,  function  (str)   {
         return str.charAt(0).toUpperCase()   + str.
substr(1).toLowerCase();
      });
   }
}
```

As you see, custom pipes are nothing but functions which take input parameters, and return some value. You need to write all logic of the pipe inside transform method. To use **`firstcharacteruppercase`** pipe, first you need to declare it in the module, after that you can use that in the component as shown in *code listing 6.6.*

Code Listing 6.6

```
import  {  Component,  Onlnit  }  from '@angular/core';
@Component({
      selector:   'app-root',
      template:  '
   <ul *ngFor='let n of names'>
     <li>{{n.name  | firstcharcateruppercase}} </li>
   </ul>
   '
})
export class AppComponent {
      names =   [];
      constructor()   {
         this.names = this.getNames();
      }
      getNames()   {
         return   [
            {   'name':   'dhananjay Kumar'   },
            {   'name':   'jason beres'   },
            {   'name':   'adam jafe'   }
         ];
      }
}
```

Now you will get in output, the first character of each name in the uppercase. You can create custom pipe for other purposes also like filtering data table etc.

To summarize:

- Custom pipes are class, which is decorated with @Pipe
- Name property of @Pipe decorator defines name of the pipe
- Pipe class should implement PipeTransform interface
- Implement pipe business logic inside transform method

Types of Pipe

There are two types of pipes:

- Stateless pipes
- Stateful pipes

What we used and created above are stateless pipes. They are pure functions, which take an input and return transformed values.

Stateful pipes are complex to implement and they remember state of the data they transform. Usually they create an HTTP request, store the response, and display the output. Angular inbuilt async pipe is example of a stateful pipe.

Summary

In this chapter, we learned about pipes in Angular. Pipes transform an input data to the desired output. Angular provides many built-in pipes; however, there could be requirements to write custom pipes. There are two types of pipes: stateless pipes and stateful pipes. We covered following topics:

- Pipes
- Built-in pipes
- Custom pipes
- Types of pipes