Pipes in Angular Lux Academy .

When we want to develop any application, we always start the application with a simple task: retrieve data, transform data, and then display the data in front of the user through the user interface. Retrieval of data from any type of data source totally depends on data service providers like web services, Web API, etc. So, once data arrives, we can push those raw data values directly to our user interface for viewing by the user. But sometimes, this is not exactly what happens. For example, in most use cases, users prefer to see a date in a simple format like 20/10/1996 rather than the raw string format Wed Oct 20 1996 03:00:00 GMT-0700 (Pacific Daylight Time).

So, it is clear from the above example that some values require editing before being viewed in the user interface. Also, that same type of transformation might be required by us in many different user interfaces. So, in this scenario, we think about some style type properties that we can create centrally and apply whenever we require it. So, for this purpose, the Angular framework introduced Angular pipes, a definite way to write display – value transformations that we can declare in our HTML..

In Angular, pipes are typescript classes that implement a single function interface, accept an input value with an optional parameter array, and return a transformed value. To perform the value transformation, we can implement any type of business logic as per our requirement. That pipe can be used in any UI to transform that particular type of data as per the desired result.

Basic Concept of Pipes.

Basically, pipes provide a sophisticated and handsome way to perform the tasks within the templates. Pipes make our code clean and structured. In Angular, Pipes accept values from the DOM elements and then return a value according to the business logic implemented within the pipes. So, pipes are one of the great features through which can transform our data into the UI and display. But we need to understand one thing very clearly, that pipes do not automatically update our model value. It basically performs the transformation of data and returns to the component. If we need to update the model data after transformation through pipes, then we need to update our model data manually.

Expect these, we can use pipes for any of the below reason –

- If we want to retrieve the position of the elements
- If we want to track the user inputs in input type elements
- If we want to restrict the user to input some value in the input controls

Why Pipe Required?

In any application, Pipe can be needed to use as per the following reason –

- We can display only some filtered elements from an array.
- We can modify or format the value.
- We can use them as a function.
- We can do all of the above combined.

Types of Pipe

In Angular, we can categories the pipes in two types i.e. Pure Pipes and Impure Pipes.

Pure Pipes : – Pure pipes in angular are those pipes which always accepts some arguments as input value and return some value as the output according to the input values. Some examples of the pure pipes are – decimal pipes, date pipes, etc. When we use these types of pipes in Angular, we provide input value with related configuration value to the pipes and pipes return us the formatted value as an output.

Impure Pipes : – Impure pipes in angular are those pipes which also accepts the input values, but return the different types of the value set according to the state of the input value. An example of the impure pipes is async pipes. These pipes always store the internal state and return different types of value as the output according to the internal state and logic.

Filter vs Pipe

The concept of the filter is mainly used in the Angular 1.x version. From Angular 2 onwards, Google deprecated the Filter concept and introduced the new concept called Pipe. Now, as per the functionality, filter, and pipes, both are working like the same. But still, there are some differences as below −

1. Filters are acting just like helpers similar to function where we can pass input and other parameters and it will return us a formatted output value. In the case of a pipe, it works as an operator. It also accepts input value and modifies that value to return the desired output.
2. Filters can not handle directly any async type operations. We need to set those values manually. But Pipe can handle async operations on its own. For these types of operations, we need to use async type pipes.

Basic Pipes

In Angular, we can use logic in the template. We can define any function within the pipe class to implement any special type data conversion or business login and then execute that particular function from the HTML template to obtain the desired result. The syntax of the Pipe in the HTML template begins with the input value and then followed the pipe symbol (|) and then need to provide the pipe name. The parameters of that pipe can be sent separately by a colon (:). The order of execution of a pipe is right to left. In General, Pipe is working within the HTML only. \

The most commonly used built-in pipes are:

- Currency
- Date
- Uppercase
- Lowercase
- JSON
- Decimal
- Percent
- Async

Syntax of Pipes

```
{{myValue | myPipe:param1:param2 | mySecondPipe:param1}}
```

Custom Pipes

Now we can define custom pipes in Angular as per our custom business logic. To configure custom pipes, we need to use a pipes object. For this, we need to define a custom pipe with the @Pipe decorator and use it by adding a pipes property to the @View decorator with the pipe class name. We use the transform method to do any logic necessary to convert the value that is being passed in as an input value. We can get a hold of the arguments array as the second parameter and pass in as many as we like from the template.

The @Pipe decorator contains the below two properties –

1. name: It contains the name of the pipe which needs to be used in the DOM elements.
2. pure: It accepts the Boolean value. It identifies the pipe is a pure or impure pipe. The default value of the pure property is true i.e. it always considers the custom pipe is a pure type pipe. It means Angular Framework will execute a pure pipe only when it detects a pure change in the input value. Pure change data can be a primitive (means data contains only single values) or non–primitive (means data contains such data type which accepts a group of values). If we need to make the pipe as impure then we need to pass false as the value against this property. In the case of the Impure pipe, the Angular framework will execute the pipes on every component change detection cycle. In this case, the pipe is often called as often as every keystroke or mouse–move.

When we define any pipe class using @Pipe decorator, we need to implement the PipeTransforms interface which mainly used to accept the input values (optional values) and return the transform values to the DOM.

What is Viewchild?

Basically, Viewchild is one of the new features introduced in the Angular framework. Angular is basically depended on component–based architecture. So when we try to develop any web page or UI, it is most obvious that that page or UI must be a component that basically contains multiple numbers of different functional components as a child component. So in simple words, it is basically a parent component – child component–based architecture. In this scenario, there are some situations occurred when a parent component needs to interact with the child component. We can establish connections between parent and child components in many ways. One of the ways is ViewChild decorator. ViewChild decorator can be used if we need to access the instance of a child component or a directive from the parent component class. So when the need to invoke any method of the child component from the parent component, it can inject the child component as a Viewchild within the parent component. In cases where you'd want to access multiple children, you'd use ViewChildren instead.

To implement ViewChild, we need to use @ViewChild decorator in the parent component. The @ViewChild decorator provides access to the class of child components from the parent component. The @ViewChild decorator is basically a function or method which accepts the name of a component class as its input and finds its selector in the template of the related component to bind to.

The basic syntax of the ViewChild decorator in Angular is as below –

```
@ViewChild(ChildComponent, { static:true}) _calculator: ChildComponent;
```

So, as per the above example, ViewChild decorator contains the following metadata:-

1. selector: It contains the directive or component name which need to use as view child
2. static: It accepts the Boolean value. We need to pass true as a value if we want to resolve the query result before change detection occurs. So, when we pass the value as true, then the Angular framework tries to locate that element at the time of component initialization i.e. in the ngOnInit method. If we pass the value as false, then Angular will try to find the element after the initializing of the view i.e. in the ngAfterViewInit method.

Best Wishes Lux Tech Academy.