Angular 8 : Online Class

01-March-2020

By: Sahosoft Solutions

Presented by : Chandan Kumar

Parameterized pipes

Passing parameter for a pipe

We can pass any number of parameters to the pipe using a colon (:)

A pipe can accept any number of optional parameters to achieve output. The parameter value can be any valid template expressions. To add optional parameters, follow the pipe name with a colon (:)

data | pipeName: parameter 1 : parameter 2 : parameter 3 …. : parameter n

Multiple pipes (chaining pipes)

We can use multiple pipes with the same data at the same. This also referred to as chaining pipes

data | pipe 1 | pipe 2 | pipe 3 | …… pipe n
OR
data | pipe 1 : parameter 1 : parameter 2 : parameter 3 …. : parameter n | pipe 2 : parameter 1 : parameter 2 : parameter 3 …. : parameter n | pipe 3 : parameter 1 : parameter 2 : parameter 3 …. : parameter n | …… pipe n : parameter 1 : parameter 2 : parameter 3 …. : parameter n

The chaining Pipe is used to perform the multiple operations within the single expression. This chaining operation will be chained using the pipe (|).

We can chain multiple pipes together. This particularly helps in scenarios where we need to associate more than one pipe that needs to be applied, and the final output will be transformed with all the pipes applied.

The workflow or chains will be triggered and apply the pipes one after another.

Async Pipe

AsyncPipe is a convenience function which makes rendering data from observables and promises much easier.

If we have Observable or Promise instance then we use it directly with AsyncPipe using directive such as NgFor, NgIfand NgSwitch. AsyncPipe belongs to angular common module. AsyncPipe subscribes to Observable or Promise and returns the latest data. It plays the role that marks the component to check for data changes. AsyncPipe is used as follows.

```
<div>{{ observablevar | async }} </div>
```

Custom Pipe

We can write our own custom pipe and that will be used in the same way as angular built-in pipes.
To create custom pipe, angular provides Pipe and **PipeTransform** interfaces. Every pipe is decorated with @Pipe where we define the name of our custom pipe.

Every pipe will implement **PipeTransform** interface. This interface provides **transform()** method and we have to override it in our custom pipe class. transform() method will decide the input types, number of arguments and its types and output type of our custom pipe.

Examples:
Welcome Custom Pipe
Bold Custom Pipe
Reverse String Custom Pipes
Limit word Custom Pipe