@Pipe

Siddharth Ajmera **@SiddAjmera**

Content:

- Pipes
- Angular's Built-In Pipes
- Custom Pipes

Pipes

- Transforms some output into template.
- Think of it as a make-up room.
- Can handle both synchronous and asynchronous data.
- Don't change the value of the actual property itself. Just transforms the way it is presented on the UI.
- Since only responsible to transform the output, the logical place to use it, is the template.
- Several built in pipes provided as a part of Angular. Custom Pipes can also be built.
- Pipes are used by using the Pipe(|) symbol after the data to be transformed..
- Pipes can be chained with other pipes.
- Pipes can also be provided with arguments by using the colon (:) sign.

Built in Pipes

UpperCasePipe

PercentPipe

AsyncPipe

DecimalPipe

DatePipe

TitleCasePipe

JsonPipe

CurrencyPipe

I18nPluralPipe

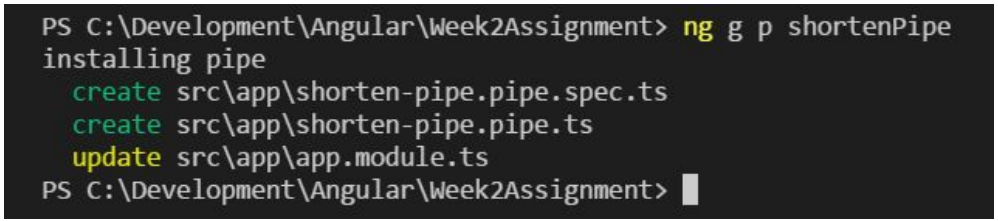
SlicePipe

I18nSelectPipe

LowerCasePipe

Custom Pipes

- Create a TypeScript Class with export keyword.
- Decorate it with the @Pipe decorator. Pass in the name property to its metadata.
- Implement the PipeTransform Interface on this class.
- Implement the transform method imposed due to the interface.
- Return the transformed data from the pipe.
- Add this pipe class to the declarations array of the module where you want to use it.
- OR simply use ng g p pipe-name. It will add the bare-bones of a pipe to your project and will also update your root module.



- You can also add arguments to your pipe by adding them to the transform function as parameters.
- By default the pipes are pure in nature. To change it and update view as the data changes, make them impure by adding the pure property to the metadata and setting it to false.
- Use AsyncPipe to handle promises or observables.