Cheat Sheet – Pipes

Using Pipes

Built-in pipes can be used like this

```
<p>{{ myString | uppercase}}</p>
```

Custom pipes (see below) have to be added to the pipes metadata of a component.

```
@Component({
      // …
      pipes: [MyCustomPipe]
})
```

Built-in Pipes

Which Pipes are built in? The best way to find out, as well as to find documentation on those pipes, is to head over to the Angular 2 API documentation (https://angular.io/docs/ts/latest/api/#!?apiFilter=pipe) and filter for "pipe" (the link above should already yield filtered results).

Custom Pipes

Of course you may create your own pipes. This is easily done by adding the **@Pipe** decorator to a class.

```
@Pipe({
      name: 'myCustomPipe' // To be used in your template code
      pure: false // Default is 'true', use 'false' to create impure pipe
})
class MyCustomPipe  {
      // ...
}
```

Pure and Impure Pipes

By default, all pipes you create are **pure pipes**. This means, that Angular 2 won't re-run them on the value they are applied to upon each change detection cycle. This behavior makes sense, as it saves performance.

If you need to re-run the pipe on each change detection cycle, you may mark your pipe as impure by setting **'pure' to 'false'**.

The async Pipe

The **async pipe** (a built-in pipe) is an impure pipe. Its job is to fetch asynchronously returned data from Promises or Observables.
Therefore, the **async pipe** is a great helper if you want to print some data to the screen which isn't available upon component initialization.