Using Pipes to Transform Output

Introduction and Why Pipes are Useful

- **Pipes** allow us to **transform output** inside of the **template**
- **Pipes** exist for both **synchronous** and **asynchronous** data
- Example
  - **<p>{{ "max" | uppercase }}</p>**
  - Transforms **"max"** to **"MAX"**

Using Pipes

- Pipes are applied in the **template** (duh)
- In our **string interpolation delimiters**, add the pipe operator (**|**) and the desired pipe's name

Parameterizing Pipes

- To handle pipe parameters, apply a colon (**:**) and argument after the pipe's name
- Example → **<p>{{ server.date | date: "fullDate" }}</p>**

Where to Learn More About Pipes

- We can learn about pipe's in the **API Reference** page of the **Angular Docs**
- Search for **pipes**

Chaining Multiple Pipes

- We can **chain** pipes by adding another **pipe operator** and **pipe name** after existing **pipes**
- All later pipes are applied to previous pipes, enforcing a **piping order**
  - From **left to right**
- Example → **<p>{{ server.date | date: "fullDate" | uppercase }}</p>**

Creating a Custom Pipe

- Generate a new **pipe** file
  - CLI → **ne generate pipe <pipe-name>**

- Pipes must implement **PipeTransform**
    - This makes us apply the **transform** method → Makes us return a new string
- Pipes must include the **@Pipe decorator**
    - Contains a **name** field → Characterizes how the pipe is referenced
- Pipes must be added to **AppModule**'s **declarations** array
- Like built-in pipes, we can add them within the **string interpolation delimiters**

Parameterizing a Custom Pipe

- To add parameters to a pipe, merely add another parameter to the pipe's **transform** method
- We can pass values in the **template** by adding a **colon** (**:**) and value **after the pipe**
- We merely add more parameters and apply them with more colons and values for however many parameters we want

Pure and Impure Pipes

- Angular **doesn't naturally re-run pipes** when the **data changes**
    - Not automatically triggered by changing data
- Can force the pipe to update by adding the **pure** field to the **@Pipe decorator** and setting it to **false**

Understanding the *Async* Pipe

- Angular will naturally output an **async string** (from a **promise** or **observable**) as an **object**
- To output the actual value, we must include the **async pipe**