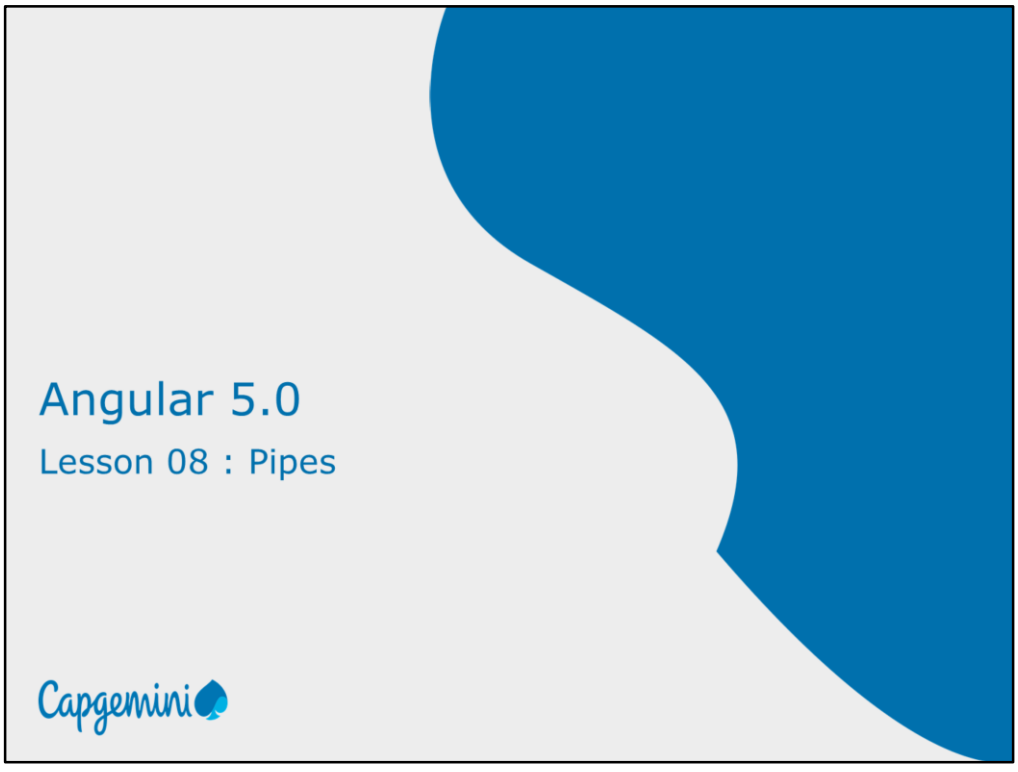
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

Lesson Objectives

- What is a pipe?
- Passing parameters to a pipe
- Chaining pipes
- Developing a custom pipe

Instructor Notes:

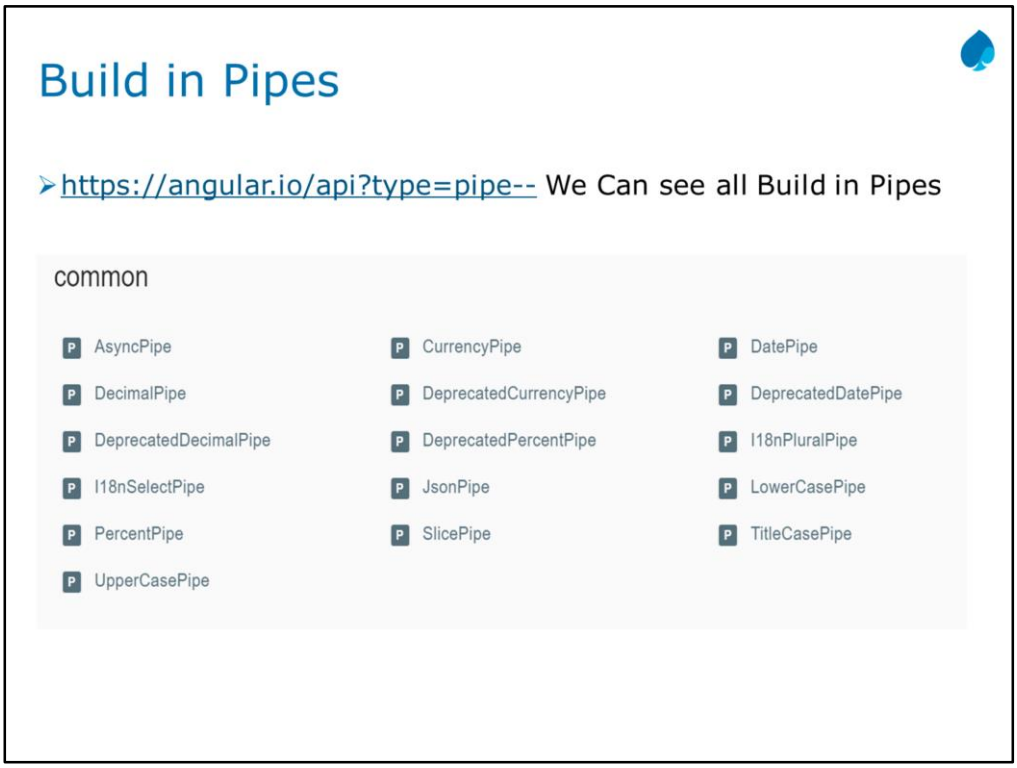
Add instructor notes here.

Pipe

- A pipe takes in data as input and transforms it to a desired output.
- Pipes transform bound properties before they are displayed
- Angular pipes, a way to write display-value transformations that we can declare in your HTML.
- To pass an argument to a pipe in the HTML form, pass it with a colon after the pipe (for multiple arguments, simply append a colon after each argument)
- Angular gives us several built-in pipe like lowercase, date, number, decimal, percent, currency, json, slice etc
- Angular provides a way to create custom pipes as well.

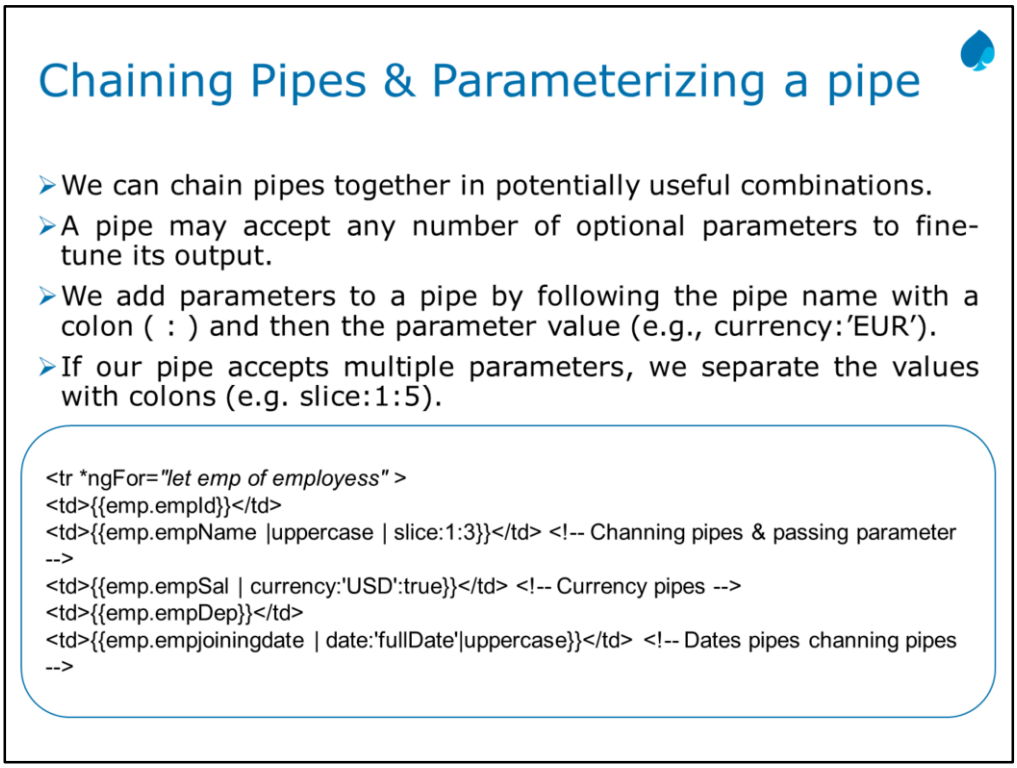
Pipes are used when the data is not in a format appropriate for display

Instructor Notes:



Angular comes with a stock of pipes such
as DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe, and PercentPipe.
They are all available for use in any template.
Angular doesn't have a FilterPipe or an OrderByPipe

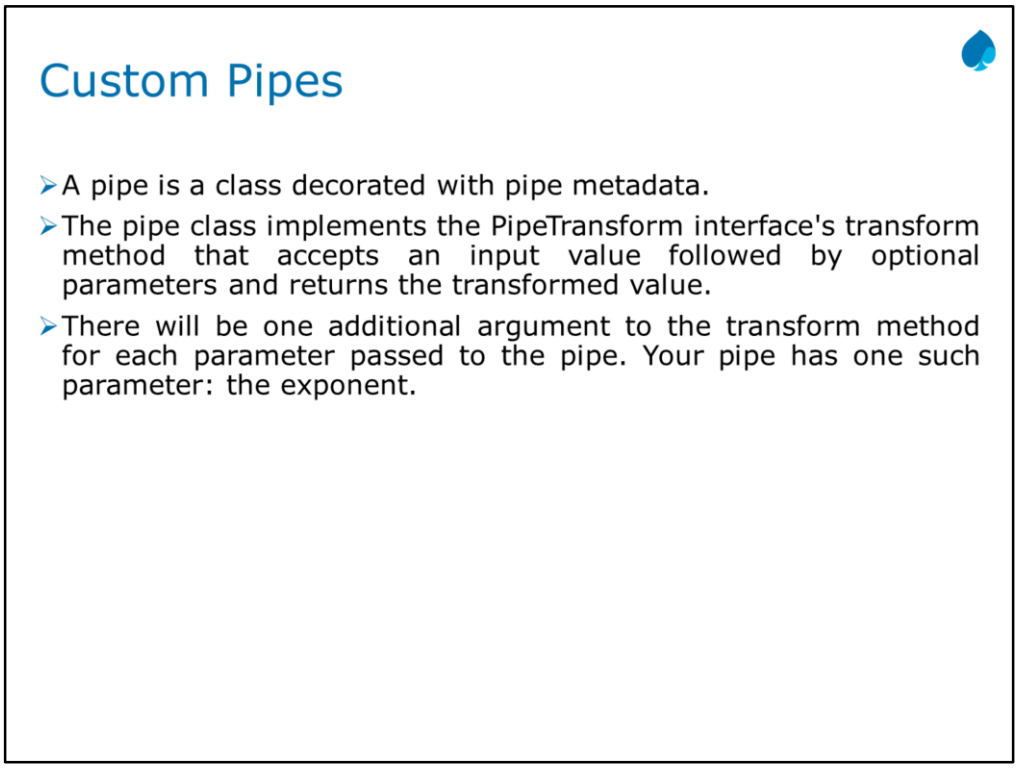
Instructor Notes:



A pipe can accept any number of optional parameters to fine-tune its output. To add parameters to a pipe, follow the pipe name with a colon (:) and then the parameter value (such as currency:'EUR'). If the pipe accepts multiple parameters, separate the values with colons (such as slice:1:5)
You can chain pipes together in potentially useful combinations. In the following example, to display the birthday in uppercase, the birthday is chained to the DatePipe and on to the UpperCasePipe. The birthday displays as **APR 15, 1988**.
The chained hero's birthday is {{ birthday | date | uppercase}}

Instructor Notes:

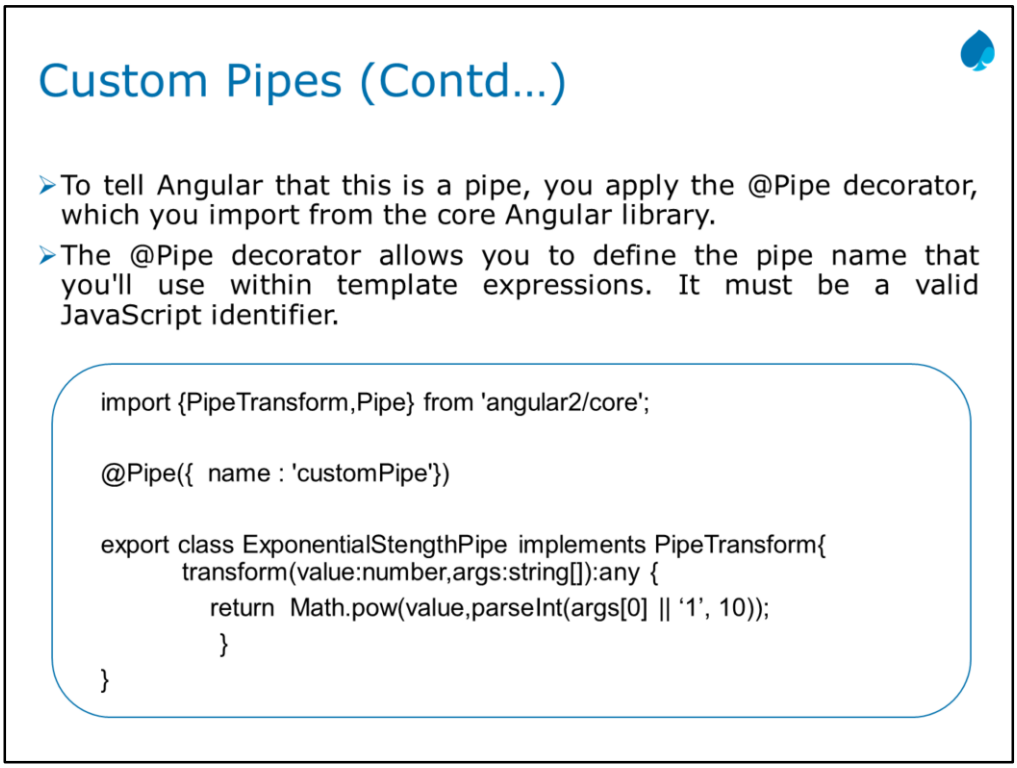
Add instructor notes here.



This pipe definition reveals the following key points:
A pipe is a class decorated with pipe metadata.
The pipe class implements the PipeTransform interface's transform method that accepts an input value followed by optional parameters and returns the transformed value.
There will be one additional argument to the transform method for each parameter passed to the pipe. Your pipe has one such parameter: the exponent.
To tell Angular that this is a pipe, you apply the @Pipe decorator, which you import from the core Angular library.
The @Pipe decorator allows you to define the pipe name that you'll use within template expressions. It must be a valid JavaScript identifier. Your pipe's name is exponentialStrength.

We use the transform method to do any logic necessary to convert the valuethat is being passed in. We can get ahold of the arguments array as the second parameter and pass in as many as we like from the template.
The *PipeTransform* interface
The transform method is essential to a pipe. The PipeTransform *interface* defines that method and guides both tooling and the compiler. Technically, it's optional; Angular looks for and executes the transform method regardless.

Instructor Notes:



In share/pipes folder, create a new .ts file named **trim.pipe.ts**
Import from 'angular/core' the module **Pipe** and **PipeTransform**. We tell Angular that this is a pipe by applying the @Pipe decorator which we import from the core Angular library.1
import {Pipe, PipeTransform} from '@angular/core';
Creating The pipe class implements the **PipeTransform** interface's transform method that accepts an input value followed by optional parameters and returns the transformed value.
The **@Pipe** decorator allows us to define the pipe name that we'll use within template expressions. It must be a valid JavaScript identifier. We should be always use the **"PipeTransform"** interface, which forces us to have a **transform()** method.
*The **transform** method is essential to a pipe. The **PipeTransform**interface defines that method and guides both tooling and the compiler. It is technically optional; Angular looks for and executes the transform method regardless.*
How to use a Custom Pipe
1. We use our custom pipe the same way we use the built-in pipes.
2. We must include our pipe in the pipes array of the @Component decorator.

Instructor Notes:

Add instructor notes here.

Demo

- Demo Build In Pipes
- Demo Custom Pipe

Add the notes here.

Instructor Notes:

Add instructor notes here.

Lab

- Lab 4.2

Add the notes here.

Instructor Notes:

Add instructor notes here.

Summary

- A pipe takes in data as input and transforms it to a desired output.
- Pipes transform bound properties before they are displayed
- Angular pipes, a way to write display-value transformations that we can declare in your HTML.
- We can chain pipes together in potentially useful combinations.
- A pipe is a class decorated with pipe metadata.
- The pipe class implements the PipeTransform interface's transform method that accepts an input value followed by optional parameters and returns the transformed value.