Lesson 8 - Pipes and Data Transformation

What We'll Cover:

1. What Are Pipes? (Transforming Data in Templates)
2. Built-in Pipes (date, uppercase, currency, etc.)
3. Chaining Pipes
4. Creating a Custom Pipe (ng generate pipe)
5. Practical Examples (Formatting Dates, Filtering Lists)

Slide 3: What Are Pipes?

- **Concept**: Pipes transform data in templates without changing the source
- **Syntax**: {{ value | pipeName }}
- **Purpose**:
  - Format data (e.g., dates, numbers)
  - Make templates cleaner and more readable
- **Analogy**: A filter on a water pipe—changes output, not the source
- **Example**:

```
{{ "hello" | uppercase }} <!-- Outputs: HELLO →
```

Key Point: Purely presentational, no logic in components

Built-in Pipes

- **Concept**: Angular provides ready-to-use pipes
- **Examples**:
  - date: {{ today | date:'medium' }} → "Feb 23, 2025, 12:00:00 PM"
  - uppercase: {{ "angular" | uppercase }} → "ANGULAR"
  - currency: {{ 42 | currency:'USD' }} → "$42.00"
  - percent: {{ 0.75 | percent }} → "75%"
- **Component**:

```
@Component({
  template: `<p>{{ today | date:'short' }}</p>`
})
export class DemoComponent {
  today = new Date();
}
```

Key Point: No extra code—just use them!

Chaining Pipes

- **Concept**: Combine multiple pipes for complex transformations
- **Syntax**: {{ value | pipe1 | pipe2 }}
- **Example**:

```
@Component({
  template: `<p>{{ today | date:'MMM d, y' | uppercase }}</p>`
})
export class DemoComponent {
  today = new Date(); // Feb 23, 2025
}
```

Output: "FEB 23, 2025"

Order Matters: Pipes execute left to right

Use Case: Format and style in one go

Creating a Custom Pipe

- **How**: Use Angular CLI to generate a pipe
- **Command**:

```
ng generate pipe reverse
```

Example:

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'reverse', standalone: true })

export class ReversePipe implements PipeTransform {

  transform(value: string): string {

      return value.split('').reverse().join('');

  }

}
```

Usage:

```
@Component({
  standalone: true,
  imports: [ReversePipe],
  template: `<p>{{ "Angular" | reverse }}</p>`
})
export class DemoComponent {}
```

Output: "ralugnA"

Practical Examples

- **Formatting Dates**:

```
template: `<p>Due: {{ dueDate | date:'fullDate' }}</p>`
dueDate = new Date('2025-03-01'); // "Saturday, March 1, 2025"
```

Filtering Lists (Custom Pipe):

```
@Pipe({ name: 'filterTasks', standalone: true })
export class FilterTasksPipe implements PipeTransform {
  transform(tasks: string[], query: string): string[] {
      return tasks.filter(task => task.includes(query));
  }
}
```

```
@Component({
  standalone: true,
  imports: [CommonModule, FilterTasksPipe],
  template: `
        <input [(ngModel)]="query">
        <ul>@for (task of tasks | filterTasks:query; track task) { <li>{{ task }}</li> }</ul>
  `
})
export class TaskListComponent {
  tasks = ['Learn Angular', 'Build App', 'Test Pipe'];
  query = '';
}
```

Demo: Type "Angular" → Only "Learn Angular" shows

Putting It Together

- **Mini-Project**: Task list with pipes
- **Code**:

```
@Component({
  standalone: true,
  imports: [CommonModule, FormsModule, FilterTasksPipe],
  template: `
        <input [(ngModel)]="query" placeholder="Filter tasks">
        <ul>
        @for (task of tasks | filterTasks:query; track task) {
        <li>{{ task | uppercase }} - {{ dueDate | date:'shortDate' }}</li>
        }
        </ul>
  `
})
```

```
export class TaskListComponent {
  tasks = ['Learn Angular', 'Build App'];
  query = '';
  dueDate = new Date();
}
```

Output: "LEARN ANGULAR - 2/23/25", etc.

Key Takeaways

- Pipes transform data in templates
- Built-in pipes handle common formats (date, currency)
- Chain pipes for multi-step transformations
- Custom pipes solve specific needs
- Practical for dates, lists, and more