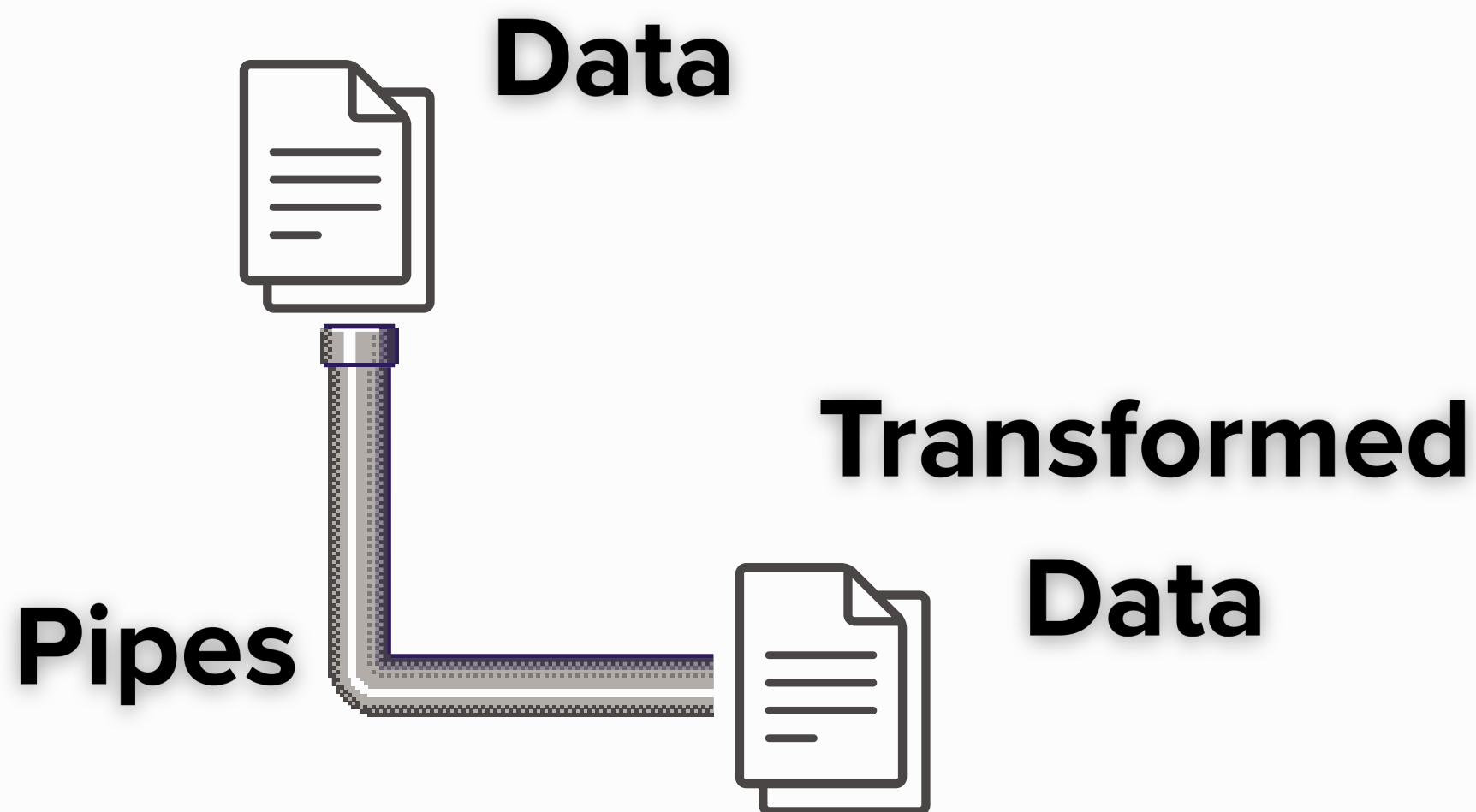


Pipes in Angular



Turn raw data into polished views seamlessly.

Niyati Kaneriya



What Are Pipes in Angular?

- Pipes let you transform and format data directly in Angular templates without touching your component logic.
- For example, you can use pipes to:
- Format dates (date pipe)
- Convert text case (uppercase, lowercase)
- Format numbers or currencies (number, currency)

```
{{ value | pipeName:optionalParameters }}
```

Why Use Pipes?

- Pipes help you write **cleaner**, more **expressive templates** by transforming data right where it's displayed.
- Here's why developers love them:
- Cleaner Templates
- Reusability
- Declarative Logic
- Chaining Power
- Improved Readability
- Built-in & Custom Options

Pipes aren't just syntax sugar — they're productivity tools disguised as one-liners.

Built-in Pipes

Examples

- **DATEPIPE**
`{{ TODAY | DATE: 'SHORTDATE' }}`
- **CURRENCYPIPE**
`{{ PRICE | CURRENCY: 'INR' }}`
- **DECIMALPIPE**
`{{ VALUE | NUMBER: '1.2-2' }}`
- **UPPERCASEPIPE**
`{{ NAME | UPPERCASE }}`

No need to reinvent the wheel — built-in pipes handle most of your formatting needs.

Chaining Pipes

Angular lets you chain multiple pipes together to apply several transformations in sequence — all inline, all elegant.

```
{{ dateValue | date:'fullDate' | uppercase }}
```

- First, dateValue is converted into a full date string
- Then, the result is transformed to UPPERCASE

With chaining, your templates stay clean and your data stays polished

Custom Pipe Example

Create your own pipe in 3 steps:

Generate the Pipe

```
ng generate pipe reverse
```

Add Your Logic

```
transform(value: string): string {  
  return value.split('').reverse().join('');  
}
```

Use it in your template:

```
{{ 'Angular' | reverse }}
```

»» ralugnA

Pure vs Impure Pipes

Pure (default): Called only when inputs change

Impure: Called on every change detection cycle

Use impure only when needed — performance matters!

Common Pitfalls

- Don't overuse pipes in performance-critical sections
- Avoid heavy logic in templates
- Know when to go custom

TL;DR – Pipes Are Powerful!

Simpler templates



reusable logic



Pipes

Next time you write a helper method, ask yourself — could this be a *pipe*?