If you have a string that contains HTML tags and you want to **extract only the inner text** (without the tags) in Angular, you can do it safely using the browser's built-in DOM parsing.

Here's a **clean and safe** approach:

1. Using `DOMParser` (Recommended)

```
// Example function to strip HTML tags and get only text
getInnerTextFromHtml(htmlString: string): string {
  if (!htmlString) return '';

  try {
    const parser = new DOMParser();
    const doc = parser.parseFromString(htmlString, 'text/html');
    return doc.body.textContent || '';
  } catch (error) {
    console.error('Error parsing HTML:', error);
    return '';
  }
}
```

Usage:

```
const html = '<p>Hello <strong>World</strong>!</p>';
const text = this.getInnerTextFromHtml(html);
console.log(text); // Output: "Hello World!"
```

2. Using a Temporary `<div>` Element

```
getInnerTextFromHtml(htmlString: string): string {
  if (!htmlString) return '';

  const tempDiv = document.createElement('div');
  tempDiv.innerHTML = htmlString;
  return tempDiv.textContent || '';
}
```

3. As an Angular Pipe (Reusable)

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

@Pipe({ name: 'stripHtml' })
export class StripHtmlPipe implements PipeTransform {
  transform(value: string): string {
    if (!value) return '';
    const tempDiv = document.createElement('div');
    tempDiv.innerHTML = value;
    return tempDiv.textContent || '';
  }
}
```

Usage in Template:

```
<p>{{ htmlString | stripHtml }}</p>
```

✅ Why this works:

- Both methods rely on the browser's HTML parser, which is safer and more accurate than regex for stripping tags.
- Works in Angular without needing `innerHTML` binding or `DomSanitizer` since we're not rendering HTML, just extracting text.

If you want, I can also show you a **version that removes HTML tags using a regex** for quick one-liners, but it's less safe for complex HTML.

Do you want me to include that regex-based quick method too?