Computer Code

What Is Ng-Content?

One Man’s Epic Journey Across The Angular Landscape.

Josh Hicks Follow
Jul 6, 2018 · 4 min read ★

There are an enormous amount of things available to developers in the Angular framework. Some of them are used daily, and some never. As part of my ongoing journey to discover all that Angular has to offer, I occasionally like to explore some of the more seldom seen features. Today, we dig into **<ng-content></ng-content>**.

One of the main goals of Angular is to help the developer create reusable and composable components. I think ng-content is one of the simplest ways to do that once you understand how it works.

Dynamic content. That is the simplest way to explain what ng-content provides. You use the **<ng-content></ng-content>** tag as a placeholder for that dynamic content, then when the template is parsed Angular will replace that placeholder tag with your content. Think of it like curly brace interpolation, but on a bigger scale. The technical term for this is “content projection" because you are *projecting* content from the parent component into the designated child component.

If you understand **{{myValue}}**, then you understand the basics of what ng-content does. The difference is where that value comes from. With normal curly brace interpolation the value comes from the component. With ng-content the value comes from the component **in its execution context**.

Okay, enough talking. Let's see some code!

Let's say you want to create a reusable button in your app.

/src/app/add-button.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'add-btn',
  template: `
    <button class="add-btn" (click)="add()">
      Add New Item
    </button>
  `
})
export class AddButtonComponent {
  add() {
    // do stuff
  }
}
```

Here we can see a generic add button which triggers an event when clicked. Nothing crazy here. The main thing I want to point out is the button's text. "Add New Item" is hard-coded in the template. But, what if we wanted to get more specific with our button text? For example, "Add Coffee". We *could* put that value in the component like this:

/src/app/add-button.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'add-btn',
  template: `
    <button class="add-btn" (click)="add()">
      {{buttonText}}
    </button>
  `
})
export class AddButtonComponent {
  buttonText: string = 'Add Coffee';

  add() {
    // do stuff
  }
}
```

Or even as an Input from the parent component like this:

/src/app/add-button.component.ts

```
import { Component, Input } from '@angular/core';
```

```
@Component({
  selector: 'add-btn',
  template: `
    <button class="add-btn" (click)="add()">
      {{buttonText}}
    </button>
  `
})
export class AddButtonComponent {
  @Input() buttonText: string;

  add() {
    // do stuff
  }
}
```

/src/app/app.component.html

```
<h1>ng-content demo</h1>

<div class="my-content">
  <add-btn [buttonText]="'Add Latte'"></add-btn>
</div>
```

These ways work but this is where **ng-content** shines. Take a look at this:

/src/app/add-button.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'add-btn',
```

```
  template: `
    <button class="add-btn" (click)="add()">
      <ng-content></ng-content>
    </button>
  `
})
export class AddButtonComponent {
  add() {
    // do stuff
  }
}
```

/src/app/app.component.html

```
<h1>ng-content demo</h1>

<div class="my-content">
  <add-btn>
    Add New Item
  </add-btn>
</div>
```

See what's happening here?

In the template for the reusable add button component we use the **<ng-content></ng-content>** tag as our placeholder for the button text. This is telling Angular, "*Hey, I don't know what this is supposed to be right now but, I promise to tell you later*".

Then when the button component is actually being used… BAM! We put whatever text we want *inside* the component and then Angular will be like "*Oh Snap! Now I know what value to use for the button's text!*".

Pretty cool. It strips out so much of the extra setup we used in the earlier examples. No inputs to keep track of, no hard-coded values in the component. Just set up the original template and worry about it later when you actually need it.

Dynamic text is just the tip of the metaphorical iceberg though. You can put lots of different things inside the component. Including HTML tags and even other components. Hopefully your mind is spinning with the possibilities.

See something you like in this article? Let me know.

See something wrong in this article? Definitely let me know!

JavaScript Angular Software Development Coding Programming