In Angular, ng-template is a powerful structural directive that allows you to define and reuse blocks of HTML within your templates. It provides a way to create reusable templates that can be dynamically rendered based on certain conditions or passed as input to other components.

**Key Uses of ng-template:**

1. **Conditional Rendering:**
   * Combine ng-template with \*ngIf to conditionally render blocks of HTML.

HTML

<ng-template #myTemplate>

This content is displayed conditionally.

</ng-template>

<div \*ngIf="condition">

<ng-container \*ngTemplateOutlet="myTemplate"></ng-container>

</div>

1. **Reusability:**
   * Define reusable template fragments that can be used in multiple places within your application.

HTML

<ng-template #myList>

<li \*ngFor="let item of items">{{ item }}</li>

</ng-template>

<ul>

<ng-container \*ngTemplateOutlet="myList"></ng-container>

</ul>

1. **Context Passing:**
   * Pass data and other context information to the template using the let keyword.

HTML

<ng-template #myItem let-item>

<div>{{ item.name }}</div>

</ng-template>

<ng-container \*ngTemplateOutlet="myItem; context: { $implicit: myData }"></ng-container>

1. **Content Projection:**
   * Use ng-template in conjunction with ng-content for more complex content projection scenarios.
2. **Custom Directives:**
   * Create custom directives that utilize ng-template to define and render custom content.

**Key Considerations:**

* ng-template itself does not render any content. It's a placeholder for a block of HTML that can be rendered later using other directives like \*ngIf, \*ngFor, or ngTemplateOutlet.
* ng-container is often used in conjunction with ng-template to avoid creating unnecessary DOM elements.

By effectively using ng-template, you can create more modular, reusable, and maintainable Angular templates.