Angular's `<ng-template>` element defines a template that is not rendered by default.

With `<ng-template>`, you can define template content that is only being rendered by Angular when you, whether directly or indirectly, specifically instruct it to do so, allowing you to have full control over how and when the content is displayed.

@usageNotes

Structural Directives

One of the main uses for `<ng-template>` is to hold template content that will be used by Structural directives. Those directives can add and remove copies of the template content based on their own logic.

When using the structural directive shorthand, Angular creates an `<ng-template>` element behind the scenes.

TemplateRef

`<ng-template>` elements are represented as instances of the `TemplateRef` class.

To add copies of the template to the DOM, pass this object to the `ViewContainerRef` method `createEmbeddedView()`.

Template Variables

`<ng-template>` elements can be referenced in templates using standard template variables.

This is how `<ng-template>` elements are used as `ngIf` else clauses.

Such template variables can be used in conjunction with `ngTemplateOutlet` directives to render the content defined inside `<ng-template>` tags.

Querying

A Query (such as `ViewChild`) can find the `TemplateRef` associated to an `<ng-template>` element so that it can be used programmatically; for instance, to pass it to the `ViewContainerRef` method `createEmbeddedView()`.

Context

Inside the `<ng-template>` tags you can reference variables present in the surrounding outer template. Additionally, a context object can be associated with `<ng-template>` elements. Such an object contains variables that can be accessed from within the template contents via template (`let` and `as`) declarations.