1. Introduction to ng-template

The <ng-template> is an Angular element used to define template content that will not be rendered by default. It's used for conditional or dynamic rendering.

2. Basic Syntax

```
<ng-template>
  <p>This is inside an ng-template</p>
</ng-template>
```

This block is not rendered unless Angular explicitly tells it to.

3. Using ngIf with ng-template

```
<div *ngIf="show; else elseBlock">Shown when 'show' is true</div>
<ng-template #elseBlock>
  <div>This is the else block</div>
</ng-template>
```

4. Structural Directives and ng-template

When you use a structural directive like *ngIf or *ngFor, Angular actually transforms it into an <ng-template> behind the scenes.

5. Template Reference Variable

You can reference an ng-template using a template reference variable (e.g., #myTemplate) and use it with directives like ngIf or ViewContainerRef.

6. ngTemplateOutlet

Used to render an <ng-template> dynamically.

Example:

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

<ng-template #myTemplate>

  <p>Dynamic content here</p>

</ng-template>
```

7. Passing Data to ng-template

You can pass context data to a template using ngTemplateOutletContext.

Example:

```
<ng-container *ngTemplateOutlet="tmpl; context: { $implicit: user }"></ng-container>

<ng-template #tmpl let-user>

  <p>{{ user.name }}</p>

</ng-template>
```

8. Use Cases

- Conditional Rendering (ngIf else)

- Dynamic Component Rendering

- Custom Structural Directives

9. Conclusion

ng-template is a powerful Angular feature for managing dynamic and conditional content rendering, especially useful in combination with ngIf, ngFor, and ngTemplateOutlet.

10. Passing Templates from Parent to Child using Content Projection

Angular allows passing templates from a parent component to a child using <ng-content>.

Parent Component:

```
<app-child>
```

```
  <ng-template #projectedTemplate let-name="name">
    <p>Hello, {{ name }}!</p>
  </ng-template>
</app-child>
```

Child Component:

```
<ng-content></ng-content>
```

With @ContentChild, the child can get the template and instantiate it using ViewContainerRef:

```
@ContentChild('projectedTemplate') template: TemplateRef<any>;
this.viewContainer.createEmbeddedView(this.template, { name: 'Ajaykumar' });
```

11. Instantiating Template with ngTemplateOutlet using Different Data

ngTemplateOutlet lets you render the same template with different context data.

```
<ng-container *ngFor="let u of users">
  <ng-container *ngTemplateOutlet="templateRef; context: { $implicit: u }"></ng-container>
</ng-container>

<ng-template #templateRef let-user>
  <p>User Name: {{ user.name }}</p>
</ng-template>
```