academeez

Component Communication

7. @ContentChild

@ContentChild

- With the **@ContentChild** you can grab the projected content from the parent in your child class
- This allows you more power with your projected content, for example display it on certain condition or dynamically create it

Parent

```
@Component({
  selector: 'app-root',
  template: `
    <academeez-child>
      <ng-template>
        <h1>
          Hello from parent
        </h1>
      </ng-template>
    </academeez-child>
  `,
})
```

Child

```
export class ChildComponent {
  @ContentChild(TemplateRef)
  passedTemplate: TemplateRef<any>;
}
```

Selector

- The **@ContentType** will get in the decorator the selector which works similar to **@ViewChild** selector
  - Select by type - will look for the first match
  - Select by TRV

```
export class ChildComponent {
  @ContentChild(TemplateRef)
  passedTemplate: TemplateRef<any>;

  @ContentChild('someTRV')
  trvDeclaredInParent: ElementRef
}
```

What you can select

- Similar to **@ViewChild** you can select
  - Component, Directive, ElementRef, TemplateRef, ViewContainerRef

Lifecycle hooks

- The lifecycle hook for content projection works here as well
- **AfterContentInit** will jump once after the **@ContentChild** properties has bee populated
- **AfterContentChecked** will jump after every change detection which allows you to react to changes in you **@ContentChild** properties.

@ContentChildren

- Very similar to **@ViewChildren** it allows us to select a QueryList from the projected content

@ContentType main usage

academeez

- The **@ContentType** allows you to create a more generic components and directives
- You can get entire snippets of template from the parent to customise the look and behaviour of you component or directive and thus making them applicable to different apps
- We can get inspiration to different patterns we can use **@ContentChild** by looking at **@angular/material**

Summary

- With the **@ContentType** decorator we can grab the template passed in the parent between the child tags and get template items in our child component class.
- The main usage is to make the child component more generic, I can now get from the parent customised appearance and change my behaviour based on what I get.
- With the lifecycle Hooks **AfterContentInit, AfterContentChecked** I can hook to the @ContentChild properties and know when they are initialised and when they are changed.

academeez

Thank You

Congratulations! You are an expert in component communication