

To access the view component of parent in child class we use @contentChild

Step1: we will generate one child component demo.

Step2:Call this component in App(parent)

<h1>Welcome to the ChildContent</h1>

<app-demo>

  <p #access>We will see the child component</p>

</app-demo>

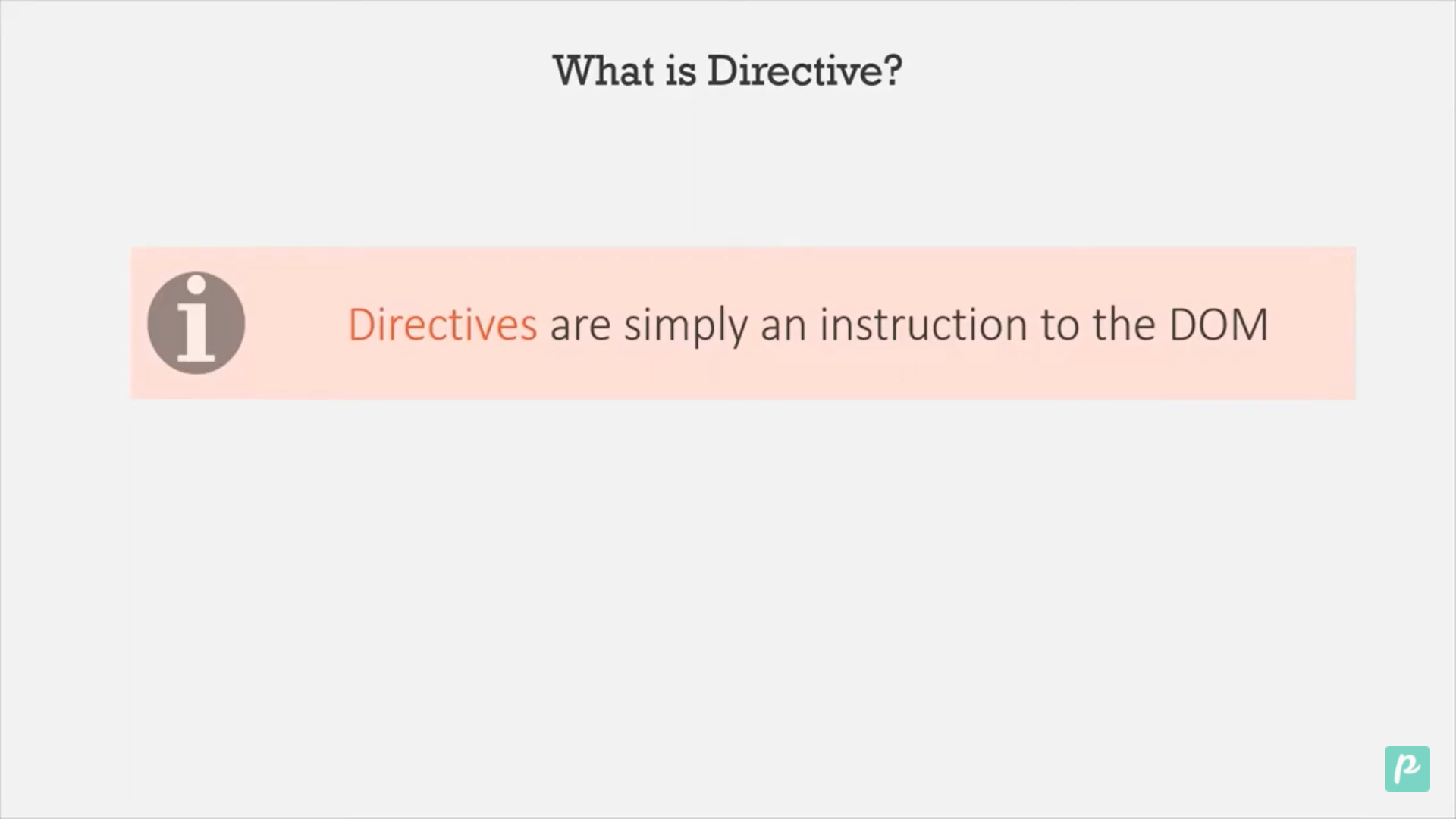
Now to access the <p> in child class .

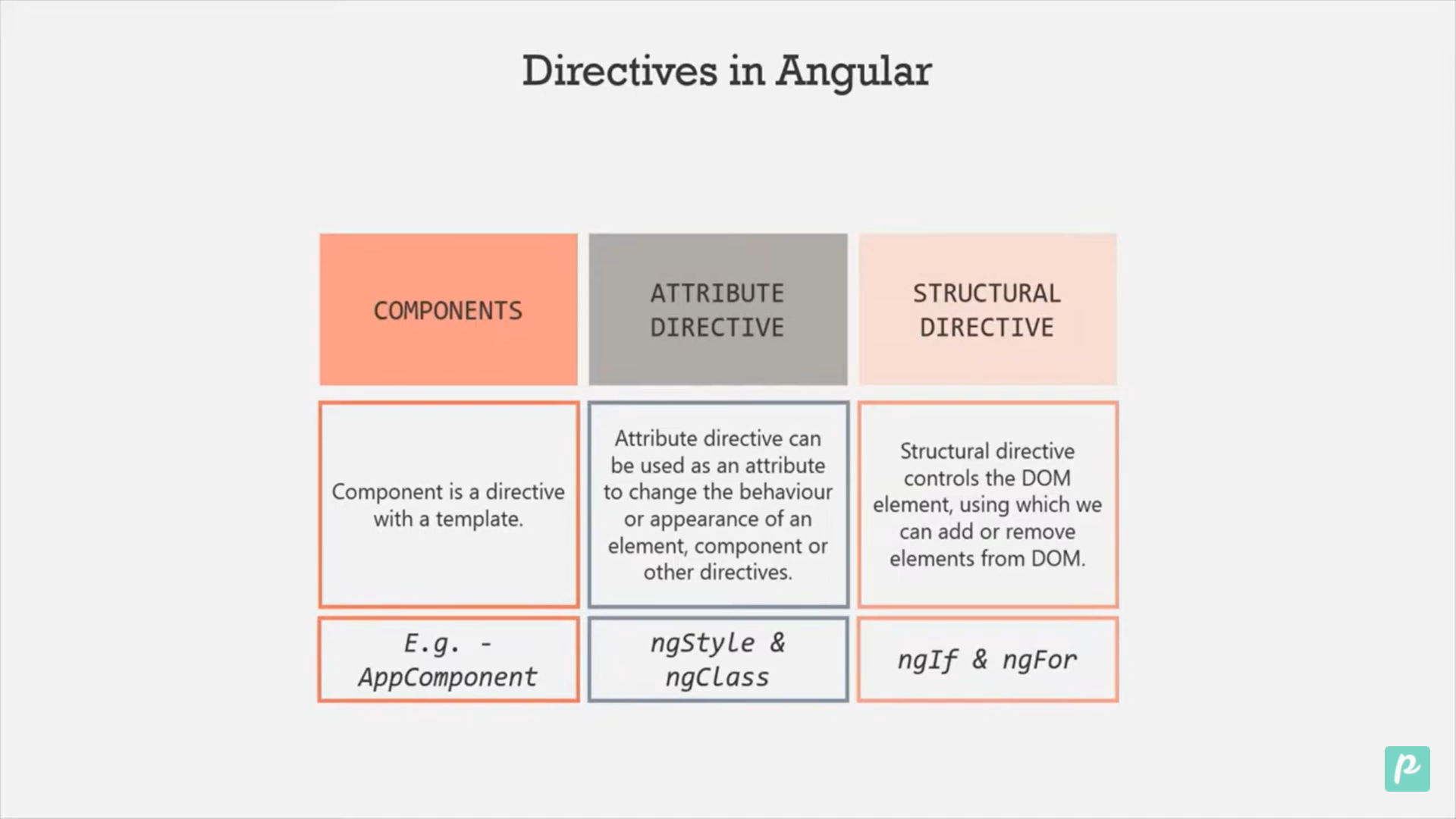
Step3: in child component view

<ng-content></ng-content>

Step4: in child component class.

Custom attribute Directives.





Step1: Create a folder for CustomDirectives.

Step2: Create a file of name setBackground.directive.ts

Step3: Create a class ,export it and use @directive()

import { Directive, ElementRef } from '@angular/core';

@Directive({

  selector: '[setbackground]',

})

export class SetBackGround {

  constructor(element: ElementRef) {

    element.nativeElement.style.backgroundColor = 'red';

  }

}

Step 4: Initalize it in appmodule.ts in declaration section.

Step 5: now we can use it.

<app-demo>

  <p #access setbackground>We will see the child component</p>

</app-demo>

Step 6: Writing logic in constructor is not a good practice instead of that we should write it in ngOninit()

import { Directive, ElementRef, OnInit } from '@angular/core';

@Directive({

  selector: '[setbackground]',

})

export class SetBackGround implements OnInit {

  private element: ElementRef;

  constructor(element: ElementRef) {

    this.element = element;

  }

  ngOnInit(): void {

    this.element.nativeElement.style.backgroundColor = 'red';

  }

}