Custom Directives

Component directive is used to create HTML template. Attribute directive changes the appearance and behavior of DOM element. Structural directive changes the DOM layout by adding and removing DOM elements.

In angular we create these directives using `@Directive()` decorator. It has a `selector` metadata that defines the custom directive name. Angular also provides built-in directives. The built-in attribute directives are `NgStyle`, `NgClass` etc. These directives change the appearance and behavior of HTML elements. The built-in structural directives are `NgFor` and `NgIf` etc. These directives can add and remove HTML elements from the DOM layout. Custom directives are created using following syntax.

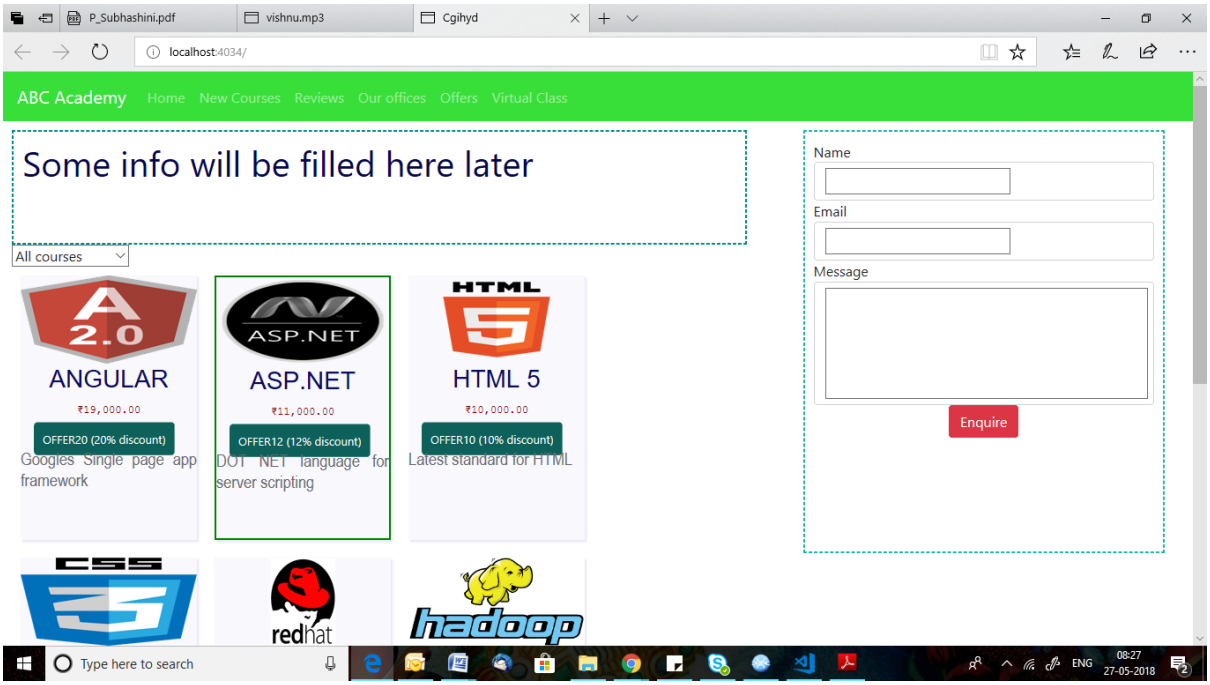
Custom Attribute directive

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

@Directive({
  selector: '[highlight]'
})
export class HighlightDirective {

  @Input('highlight') price;
  constructor(private el:ElementRef) {

   }
  @HostListener('mouseover')
   changeBorder(){
     if(this.price>15000)
        this.el.nativeElement.style.border="2px solid red";
     else if(this.price>=11000)
        this.el.nativeElement.style.border="2px solid  green";
      else
        this.el.nativeElement.style.border="2px solid  black";
      }

  @HostListener('mouseleave')
    retainOriginal(){
      this.el.nativeElement.style.border="none";
    }


}

```

In the above example, the directive selector is highlight and if it is applied on a component or a HTML element , the host will be listened for mouse events and when we take mouseover the host element the color will change to red, green and black according to the price of the course which we pass as parameter.

Here is how it is applied

```
<div class="course"  [highlight]="current.price" >
  <div class="coursepic">
     <img src="../../../../assets/images/{{current.image}}">
```



You can find the selected course bordered with green color

Custom Structural directive

Structural directive is used to change the DOM layout by adding and removing DOM elements. Find the steps to create custom structural directive.

1. Create a class decorated with `@Directive()`.
2. Assign the structural directive name using `selector` metadata of `@Directive()` decorator enclosed with bracket `[]` .
3. Create a setter method decorated with `@Input()`. We need to take care that the method name should be same as directive name.
4. Configure custom structural directive class in application module in the `declarations` metadata of `@NgModule` decorator.

To create custom structural directive we need to use `TemplateRef` and `ViewContainerRef` etc that will help to change the DOM layout.

TemplateRef : It represents an embedded template that can be used to instantiate embedded views.

ViewContainerRef: It represents a container where one or more views can be attached.

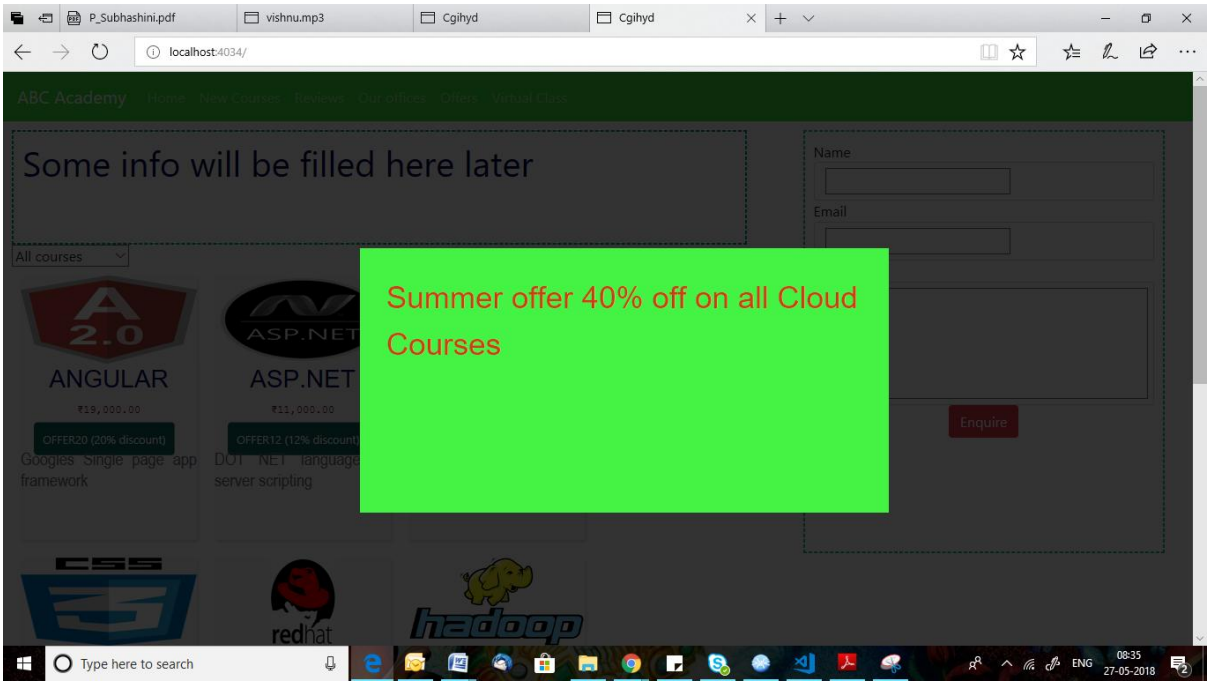
Now let us create custom structural directives.

```
import { Directive,Input } from '@angular/core';
import { TemplateRef,ViewContainerRef} from '@angular/core';
import {OfferService} from '../offer.service';
@Directive({
  selector: '[offer]'
})
export class OfferbannerDirective {

  constructor(private tref:TemplateRef<any>,
    private vc:ViewContainerRef,private os:OfferService) {

    }

    @Input() set offer(data){
        if(this.os.offerday)
        {
   this.vc.createEmbeddedView(this.tref)
   setTimeout(()=>this.vc.clear(),data*1000);
        }

      else{
        this.vc.clear();
      }

    }


}

```

```
import { Directive,Input } from '@angular/core';
import { TemplateRef,ViewContainerRef} from '@angular/core';
import {OfferService} from '../offer.service';
@Directive({
  selector: '[offer]'
})
export class OfferbannerDirective {

  constructor(private tref:TemplateRef<any>,
    private vc:ViewContainerRef,private os:OfferService) {

    }

    @Input() set offer(data){
        if(this.os.offerday)
        {
   this.vc.createEmbeddedView(this.tref)
   setTimeout(()=>this.vc.clear(),data*1000);
        }

      else{
        this.vc.clear();
      }

    }


}

```

In the above example, If we pass number of second to the offer custom structural directive, the directive will show the content if and only if the service offerday variable return true. Even if the service offerday returns true, the element will be displayed only for that many number of seconds which we passed.

```
<div class="offer" *offer="5">
    {{offermessage}}
</div>
```



In the above screen, the banner will be shown for 5 seconds and it will fade away automatically.