Angular 5 Training Course

Exercise O-directives

- This exercise will create a **custom directive**.
- A custom directive can be added as an attribute of a component instance in a template.

Setup

- Locate and rebuild the **useful/directives** project.

```
npm install
ng serve --open
```

Review the existing project

- The main template contains this code:

```
<tincan *ngFor="let p of products" [product]="p"></tincan>
```

- Tincan is a component that displays information about a can of beans.
- The **built-in ngFor directive** is used to iterate over an array of product objects. Each object is passed as an input into the component.
- Each product object looks like this:

```
    { name:"Heinz",
        price:0.45,
        image:"heinz.jpg",
        info:"No added sugar",
        maximum:4
    },
```

Custom directive

- We want to build a custom directive.
- It will be attached as an attribute of a instance.
- It will limit the maximum quantity that can be purchased.
- The syntax to use the directive will look like this:

```
<tincan
        *ngFor="let p of products"
        [product]="p"
        limit (more)="buyProduct($event)"
        maximum={{p.maximum}} >
</tincan>
```

- **Limit** is the name of the directive.
- **Maximum** is an input defined within the directive.
- The **(more)** event is emitted by the directive when the user clicks on a tin-can.
- The main component listens for this event and runs its buyProduct method.
- The limit directive only emits an event if the user selected less than the maximum.
- The limit directive styles its parent tincan component as faded once the maximum has been reached.

Create a new directive.

- Use the Angular CLI tools to create a new directive.

```
ng generate directive directive/limit
```

- *Note: app.module.ts has been updated to include this directive.*
- Review **src/app/directive/limit.directive.ts**
- Simplify the name of the directive:

```
selector: '[limit]'
```

- Create a constructor method with debugging code.

```
constructor() {
    console.log("Limit Directive");
}
```

- To use the directive, we need to attach it as an attribute to elements in our template.

```
<tincan
limit
*ngFor="let p of products" [product]="p">
```

```
</tincan>
```

- "Limit Directive" should be logged to the browser console.

Parent element of a directive.

- We can add code within the directive to sense the component instance it is attached to.

```
import { ElementRef } from '@angular/core';

constructor(elem: ElementRef) {
console.log(elem.nativeElement)
```

}

- The tincan component instances will be logged to the console.

Maximum input into directive

- We want to pass **the maximum quantity allowed** into the directive as an **input**.
- Define the input in the directive.

```
import { Input } from '@angular/core';
@Input() maximum;
```

- Add debugging in ngOnInit.

```
import { OnInit } from '@angular/core';

ngOnInit() {
    console.log(this.maximum);
}
```

- Add an attribute to the main template to pass in the maximum value.

```
limit maximum={{p.maximum}}
```

Sense events from the directives parent element

- We can use the **HostListener decorator** in the directive.
- This will sense events that happen to the parent element that the directive is attached to.

```
import {HostListener} from '@angular/core';

@HostListener('click') selectProduct(): void {
    console.log('click',this);
}
```

- Data will be logged to the browser console when the user selects a can of beans.

```
click LimitDirective {maximum: "4"}
```

Emit events from the directive.

- We can emit events from the directive and listen for that event further up the component hierarchy.
- Define an **output** of type **EventEmitter**.

```
import { Output,EventEmitter } from '@angular/core';

@Output() more:EventEmitter<number> = new EventEmitter();
```

- Emit an event from selectProduct().

```
this.more.emit(1);
```

- Listen for the event in the main template.

```
(more)="buyProduct($event)"
```

- Add a new method the main component:

```
  buyProduct( e ) {
  console.log("buyProduct",e);
  }
```

- Clicking on a can of beans logs **buyProduct 1** to the console.

Add logic to limit quantity purchased

- Add logic to check the quantity ordered.

- Once the limit is reached, emit no further events.

```
quantity:number=0;

if(this.quantity < this.maximum) {
    this.quantity++;
    this.more.emit( this.quantity );
}
```

- Style the parent component as inactive once the limit is reached.
- The HostBinding decoration allows us to apply CSS styling to the TinCan component from within the directive.

```
import { HostBinding } from '@angular/core';

@HostBinding('class.limit') isActive:boolean = false;
```

- This code set isActive to true once the quantity limit has been reached.
- This dynamically attaches the limit class to the parent TinCan component.

```
this.isActive = (this.quantity >= this.quantity);
```

- This rule in the main component CSS uses a contextual selector to style the cans which have reached their quantity limit.

```
.shop .limit{
    opacity: 0.5;
    cursor: default;
}
```

- The directive now works and limits the quantity that can be purchased for any one item.