Angular 6 Training Course

Exercise N-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 **app.component.html** uses an ngFor to iterate over instances of the Tincan component.

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

- A product object is passed as an input to each instance.

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

- The component **tincan/tincan.component.ts** logs the object passed in to the console:

```
ngOnInit() {
console.log(this.product);
```

}

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"
        dLimit (more)="buyProduct($event)"
        maximum={{p.maximum}} >
</tincan>
```

- **dLimit** 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: '[dLimit]'
```

- Create a constructor method with debugging code.

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

- To use the directive, add it as an attribute of the Tincan instance. *Optionally document its purpose.*
  <tincan dLimit *ngFor="let p of products" [product]="p">
- "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) {
// Log the component instance that this directive is attached
to.
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';

export class LimitDirective implements OnInit { ....

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

- Add a maximum attribute. This value will be logged to the browser console.

```
<tincan dLimit maximum=4 .... ></tincan>
```

- Edit the products array in the main component to carry a maximum value for each object:

```
{ name:"Heinz",price:0.45,image:"heinz.jpg",info:"No added
sugar",maximum:2}
```

- Use this value in the template:

```
<tincan dLimit maximum={{p.maximum}} .... ></tincan>
```

Sense events from the directives parent element

- We can use the **HostListener decorator** in the directive.
- This will sense events that happen to the component 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 will emit a more event from the directive and listen for (more) events further up the component hierarchy.
- Define an output in the directive:

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

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

- Emit an event from selectProduct().

```
@HostListener('click') selectProduct(): void {
        this.more.emit(1);
}
```

- Listen for the event in app.component.html, and when it happens call method buyProduct in the main component.

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

- Create a new method in the main component to handle this event:

```
  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 limit the quantity ordered. Once the limit is reached, emit no further events.

- Define a quantity variable

```
quantity:number=0;
```

- Add conditional code to the selectProduct method:

```
@HostListener('click') selectProduct(): void {

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

Fade the parent component once maximum reached

- Style the parent component to look dimmed/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';
```

- This code maps the appearance of the CSS limit class to a boolean variable. When the boolean becomes true, the class will be attached to the parent component.

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

- Add logic to selectProduct to change the boolean once the maximum has been reached.

```
    if(this.quantity < this.maximum) {

    this.quantity++;
    this.more.emit( this.quantity );

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

- 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. Once the limit is reached the product becomes faded and no more items can be ordered.