Structural Directives

- The Structural directive changes the DOM layout by adding and removing DOM elements
- Angular has several built-in structural directives, namely `NgIf`, `NgSwitch`, and `NgFor`
- When working with structural directives, we should ask ourselves to think carefully about the consequences of adding and removing elements and of creating and destroying components
- Angular uses the html5 `<template>` tag to add or remove DOM elements
- By default, Angular replaces `<template>` with `<script>` tag if no behavior is attached
- The `*` before a directive name is a shorthand for including the directive content in the `<template>` tag
- Below you can see the built-in `NgIf` directive with and without the asterisks `*`:

With `*`

```
<p *ngIf="condition"></p>
```

Without `*`

```
<template [ngIf]="condition">
  <p></p>
</template>
```

Notice how the `<p>` tag is wrapped with a `<template>` and the condition is bound to the `[ngIf]` property of the directive

TODO (writing a custom structural directive)

```
@Directive({
  selector: '[myUnless]'
})
class UnlessDirective {
```

```
5
6     constructor(
7       private tRef: TemplateRef,
8       private vContainer: ViewContainerRef
9     ) { }
10
11    @Input() set myUnless(condition: boolean) {
12      if (!condition) {
13        this.vContainer.createEmbeddedView(this.tRef);
14      } else {
15        this.vContainer.clear();
16      }
17    }
18  }
```

TemplateRef: TODO: details

ViewContainerRef: TODO: details

`@Input() set myUnless(condition: boolean) {}` : TODO: details