Cheat Sheet – Directives

Directives Metadata

Like Components (which are Directives in the end), Directives are "normal" TypeScript/ JS classes, "transformed" into Directives.

```
@Directive({
    selector: '[myDirective]'
})
```

The @Directive decorator/ metadata shares a lot of configuration options with the @Component metadata. One important difference, though, is the missing of the **template** metadata.
Directives don't have a view and therefore don't offer this metadata.

The selector works like a CSS selector (the same if of course true for the @Component decorator). This means, that you select elements which should get affected by that directive like you would select elements to be styled in your CSS code.

Examples:

Selector	Selected Element (Example)
`.any-class`	`<div class="any-class">`
`#any-id`	`<div id="any-id">`
`tag-selector`	`<tag-selector>`
`[attribute]`	`<div attribute>`

Attribute Directives & Structural Directives

Attribute Directives are called like this, because they are applied like HTML attributes (example: <div myDirective>) and impact the element they are attached to (e.g. change the styling).
Examples:

```
<div [ngClass]="{'class-to-be-applied': true}
<input ngControl="controlName">
```

Structural Directives are called like this, because they change the structure of the DOM, not only the element on which they sit.
Examples:

```
<li *ngFor="let item of items">{{item}}</li>
<div *ngIf="condition">
```

De-Sugaring of Structural Directives

Angular 2 provides a nice syntax for structural directives (*ngFor). However, "*" is no fith databinding syntax. There are only the four mentioned in the respective lecture.
Instead, Angular 2 transforms the "*" syntax into a "less beautiful" syntax behind the scenes.
Example:

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

becomes

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