Understanding the TypeScript Code

Section 1, Lecture 7

In case you feel lost with all that TypeScript code - here's a brief article to get you started with it. Note that I have a complete TypeScript module in this course.

You may also come back to this article after the next few videos :).

The important thing first: All files have `.ts` as file extension because we write TypeScript code in these files.

If you never saw TypeScript before it might look strange, but it's actually only JavaScript + extra Features. Y**ou can write any JS code in a TypeScript file and it will work!**

```
1. import { Component } from '@angular/core';
2. @Component({
3.     selector: 'app-root',
4.     templateUrl: 'app.component.html',
5.     styleUrls: ['app.component.css']
6. })
7. export class AppComponent {
8.     title = 'I changed it!';
9. }
```

Imports

First, we import code from another module. In this case, it's an Angular 2 module (as you can tell by the `@angular`), but that might also be one of our own modules.

Now what is a module? Basically everything you export can be called "a module". You could also simply refer to it as "some code".

One important note here is: No matter if you import some `@angular` package/ module or code from your own file, you never add the file extension in the import.

For example, an import from your own file would look like this (you'll see this in later videos):

```
import { MyOtherComponent } from 'myother.component';
```

See? No `.ts` at the end!

Decorators (`@Component`)

Decorators are basically functions which get attached to other code - in this case to a class. A decorator simply leads to the execution of some code in the background which "does something". Here, the `@Component()` decorator takes a JavaScript object as an argument and uses this argument to add some metadata (in the background) to this class.

That makes this class a Component which Angular 2 can recognize.

Angular 2 uses Decorators a lot (not only for Components) and you may simply keep in mind that Decorators *"do something in the background and transform whatever they are attached to, to something else Angular 2 knows (like a Component)".*

class

Classes is a feature added by TypeScript which makes the creation of objects easier. You can think of a class as a blueprint for JS objects. Note that you rarely instantiate classes (= create objects based on them) yourself in an Angular 2 app. Most of the time, Angular 2 will do that for you.

Types

Something you don't see in the above code, but TypeScript has its name because you can assign Types to properties, variables etc.

A type declaration looks like this:

```
someProperty: string = 'Hello';
```

This makes sure that only string values may be assigned to someProperty.