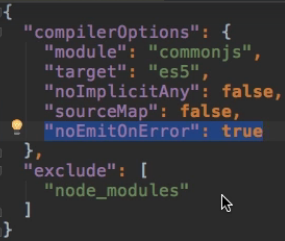
**Introduction**

**How Code Gets Compiled**  

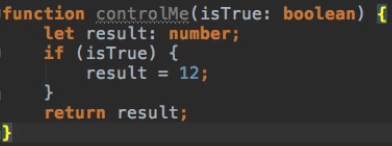
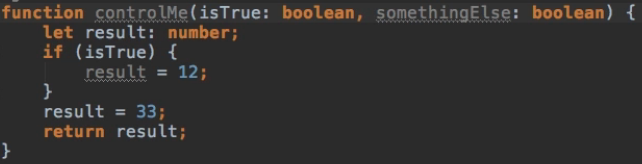

  
  
  
=> Even though we got a compilation error, it was assigned here.  
=> So why is it compiling if we get an error?  
=> Simply because it’s the default behavior.  
=> The TypeScript compiler warns you, it gives you an error, but it compiles it nonetheless, to give you a chance to run your code anyway because maybe it was wrong, maybe there is something, some import or you have another dependency in the index.html file through another script import which TypeScript isn’t able to know about but your code still works after it has been compiled.  
\* So then you’re still not writing a really TypeScript-friendly application, as you’re deliberately passing on some of the advantages it gives you but TypeScript won’t keep you from executing your code.  
\* So that’s actually not a bad behavior.  
\* **You can suppress it though, the tsconfig.json file allows you to define some compiler options**.

**Changing the Compiler Behavior on Errors**  
\* I’ll also come back to this file later when we talk about the workflows.  
  
\* The base options are:  
**“exclude”: [ “node\_modules” ]**  
=> Don’t compile anything in the node\_modules folder because that is not our code.  
**“module”: “commonjs”**  
=> Resolve our modules to be commonjs format - I’ll come back to this in the namespace and modules section.  
**“target”: “es5”**  
=> The target version to which you want to compile is ES5, which runs in pretty much all browsers.  
**“noEmitOnError”: true**  
=> **Your js file doesn’t get created in case of any errors**.

**Debugging your TypeScript Code using Source Maps**  
\* There are a lot of options you can define here and to be honest, a lot of them are options you’ll rarely or never use.  
**“sourceMap”: true**  
=> Compilation creates .js.map file which is the source map for the .js file.  
=> What’s an advantage of the source map?  
=> In Google Dev Tools > Sources > app.ts - so even though we’re running the js file, we got access to the .ts file and we can even set a breakpoint in the dev tools and reload the page to pause at that breakpoint.  
=> So we can debug our TypeScript code directly in the browser, directly in the TypeScript file.  
\* **This is a great addition and a great and very important feature which makes finding our errors much easier**.  
\* **If you are developing a project, you probably might want to use it**.

**Avoiding implicit “Any”**  
\* **This will be of type any**.  
**“noImplicitAny”: true**  
=> It’s really a good warning which kind of reminds you that maybe you want to assign a type here, so it might get you to write better code.

**More Compiler Options**  
<http://www.typescriptlang.org/docs/handbook/tsconfig-json.html>  
\* I will come back to this a bit in the workflow section.  
\* **You can have a look at this to fine tune it to your specific needs**.   
\* **The default tsconfig.json works for a lot of projects**.  
\* **Compiler Options**  
<http://www.typescriptlang.org/docs/handbook/compiler-options.html>  
\* Describes all the options as you might chain them to your tsc command or without the leading dashes this would be how you would use them in the tsconfig file then.

**Compiler Improvements with TypeScript 2.0**  
\* With the release of TypeScript 2.0, the compiler also got a little bit smarter.  
  
\* When we use tsc to compile it, we don’t get an error.  
\* But there might be a problem - the result might not be initialized when we return it.  
\* But it is implictly null here because it should be of type number and if we never initialize it, well then it is like null.  
\* If we use **“strictNullChecks”: true**, we now get an error that result is used before it is assigned.  
  
\* This is one of the things where the compiler got smarter.  
\* **It now analyzes the flow of your code better, it analyzes which parts of your code you actually can reach and so on**.  
\* There’s another addition which allows you to create cleaner code - it’s a specific compiler option.  
**“noUnusedParameters”: true**  
  


**Resource: Compiler Documentation**  
\* You can find a detailed documentation on the TypeScript Compiler Config File (tsconfig.json) here: <http://www.typescriptlang.org/docs/handbook/tsconfig-json.html>   
Details on the Compiler Options can be found here: <http://www.typescriptlang.org/docs/handbook/compiler-options.html>

**Module Summary**  
\* You should now be confident about using this compiler, compiling your TypeScript code and understand what actually happens behind the scenes.

**Resources**  
How to configure tsconfig.json  
<http://www.typescriptlang.org/docs/handbook/tsconfig-json.html>  
Compiler options  
<http://www.typescriptlang.org/docs/handbook/compiler-options.html>