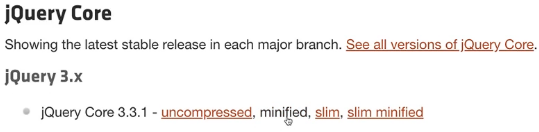
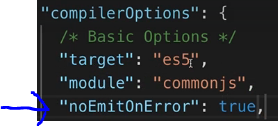
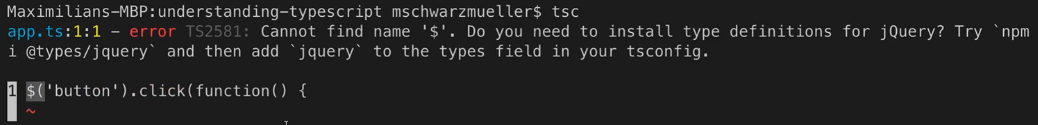
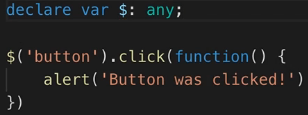
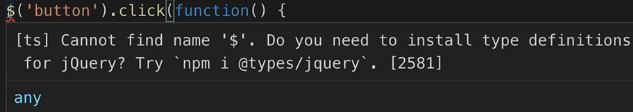
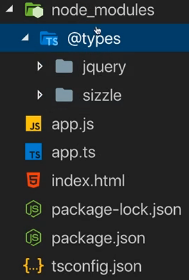
**Introduction**  
\* Throughout this course, you learned a lot about the different types we can use in TypeScript and how we can use TypeScript to enhance our projects in general.  
\* Now there certainly are projects though, where you use some Third-party libraries which don’t use TypeScript, which are not written in TypeScript.  
\* How can you use such normal JavaScript libraries in your TypeScript project and still take advantage of all these type definitions and the enhanced IDE support and so on?  
\* This section has been updated and therefore I’m using Visual Studio Code, it’s a normal code editor, normal IDE.

**Using jQuery**  
\* We can download the jQuery and add it to our project or we can use one of the CDN links to basically pull it in from a server where it’s hosted.  
\* I’m going to take the minified version of the latest jQuery version, copy that script tag.  
  
  
\* My goal is to select this button with the help of jQuery and then do something.  
\* **Let’s do it inline just as an example**.  
  
\* This works but this is normal JavaScript/jQuery.  
\* Let’s copy it into app.ts.  
**tsc --init**  


\* We get an error:  
  
\* **This makes sense because we never defined $ as a method**.

**Declaring Variables**  
\* **TypeScript doesn’t understand that this $ method is made available globally because we added the jQuery import**.  
\* We have a couple of ways to inform TypeScript that it is available.  
1) **Easiest fix** => you can always use this if you have a global method or variable of which you know that it exists but TypeScript doesn’t know it.  
  
\* Now it compiles successfully.  
\* This is a cheap way of making this work but it is a way you can definitely use if you have no other way or if you just want ot get it to work quickly.  
\* But it’s not an ideal way because with this line here, we almost pass no information at all to TypeScript about this $ variable, it doesn’t even know whether it’s a function or not, leave alone that it would know what we can do with the result of that function.

**Understanding Declaration Files**  
\* One better way to solve this which my IDE even suggests here:  
  
\* **Install** **Type** **Definitions**.  
=> **Type Definitions is a feature TypeScript offers**.  
\* You can basically add your own declaration files which hold no script logic but which hold type and function and so on definitions which TypeScript automatically considers when compiling your code.  
**jquery.d.ts** => it has to be **.d.ts**  
\* **In the file:**  
  
\* Implicitly TypeScript includes all .d.ts files in its compilation step and therefore when I run tsc, it still works because it automatically considers that file.  
\* **We just outsources that line into its own separate file but it’s important to know that TypeScript considers such files because you can define more complex statements**.  
\* This is actually an advanced feature which you mostly only need when you create Libraries you want to share with other people on your own.  
\* You can write quite advanced Type Definitions and you’ll learn more about that in the official docs.  
<http://www.typescriptlang.org/docs/handbook/declaration-files/introduction.html>  
\* **Other people already did that for jQuery and a lot of other popular JavaScript libraries**.  
\* Search for DefinitelyTyped jquery  
<https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/jquery>  
=> **Here you find type definitions for jQuery**. In the jQuiery.d.ts file you see Interfaces and so on.  
\* Here’s our click function for example:  
  
\* This file adds all the logic you need to add to make TypeScript fully aware of what’s behind that $ variable and behind the object that that function call returns for example.  
\* **Let’s remove our jquery.d.ts file and now we can simply install these Type Definitions with the help of npm**.  
**npm init**

**Using Public Type Definitions**  
**npm install --save-dev @types/jquery**  
  
\* The cool thing is that the TypeScript compiler will automatically without any extra configuration, consider all these @types folders in the node\_modules and will include the Type Definitions.  
  
\* **Now both the compilation and the IDE support works, we get autocompletion because now my IDE fully understands what’s behind that $ because TypeScript now fully understands it**.  
\* This is the way to go.  
\* This is what I strongly recommend doing when using Third-party libraries which are not written in TypeScript.  
\* You should install the respective Type Definitions which are available for basically every library that is at least a bit important and popular.

**Wrap Up**

**Resources**  
Declaration Files  
<http://www.typescriptlang.org/docs/handbook/declaration-files/introduction.html>  
DefinitelyTyped jquery  
<https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/jquery>