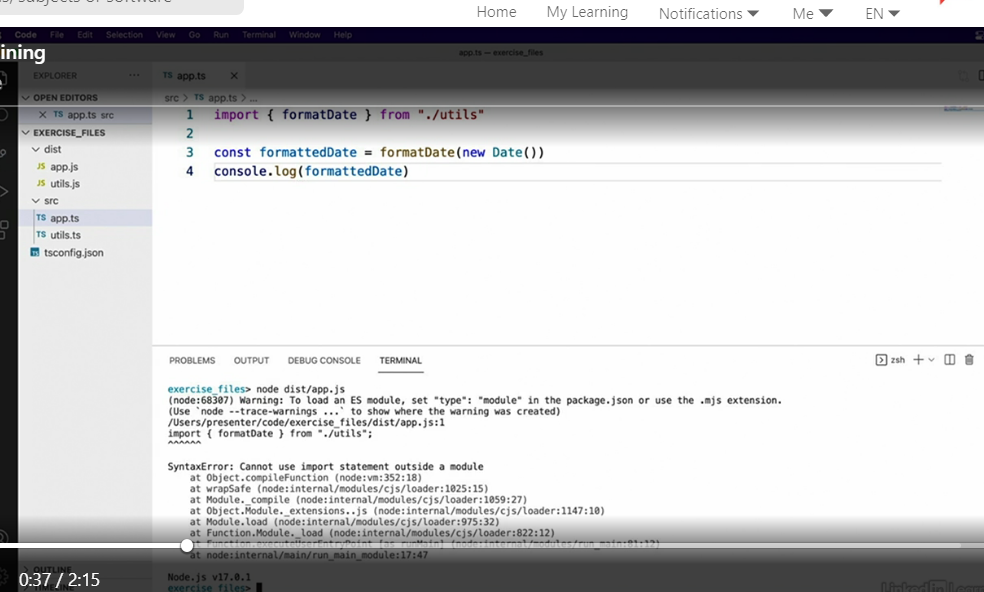
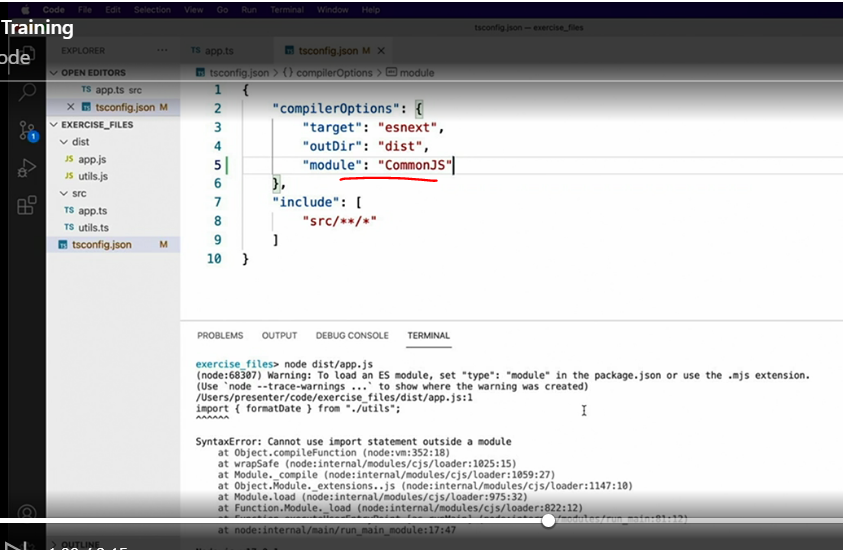
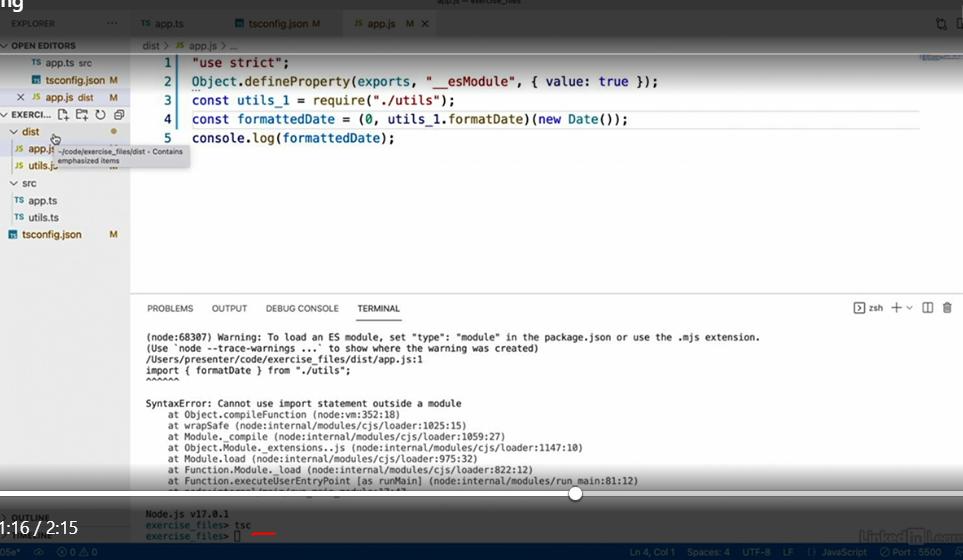
* - [Jess] If you're learning TypeScript, you're probably looking to run your code in either the browser or Node.js.
* Unfortunately, the code I've been showing won't actually run in either of those environments as it is, but in this video, I'll show you how to fix that.
* I'll start with Node.js, since that's the easiest.
* As of this recording, *Node.js expects module imports to use a different syntax* than the one I've been showing here, which means that if I try to build and run this application, I see this error.



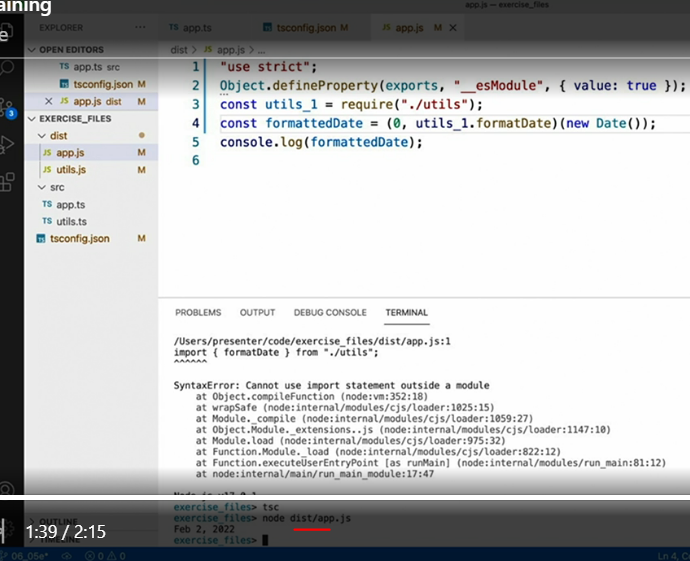
* Even though it's not obvious, this is Node's way of telling me that it doesn't support this import syntax in .js files.
* However, having TypeScript output the code that can run in Node is as simple as tweaking a *TypeScript config setting.*
* Simply set the **module** configuration setting to **commonJS**, the type of module system that Node uses.



* Now, when I save this, run the TypeScript compiler again, and check out the output code, I can see that it uses the require imports syntax to make the module imports, which is the way that Node. js expects imports to be declared.



* And I'll run the app again real quick, just to prove that it does work this time.
* So that's compiling modules for Node applications.



* Sadly, compiling for web applications is not so straightforward.
* While it is true that modern web browsers do have the capability to understand and load module-based code, it's not something I ever use in practice.

Diagram

Description automatically generated

* For module code that runs in the browser, I highly recommend that you use a **separate bundling and minification tool**, such as the wildly popular Webpack or the low configuration Parcel tool.
* In addition to understanding modules, these tools will also help optimize your code for faster downloads and execution speeds in the browser.