Modules

**A background on modules**

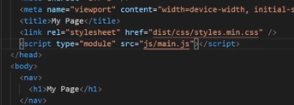
JavaScript programs started off pretty small — most of its usage in the early days was to do isolated scripting tasks, providing a bit of interactivity to your web pages where needed, so large scripts were generally not needed. Fast forward a few years and we now have complete applications being run in browsers with a lot of JavaScript, as well as JavaScript being used in other contexts (Node.js, for example).

It has therefore made sense in recent years to start thinking about providing mechanisms for splitting JavaScript programs up into separate modules that can be imported when needed. Node.js has had this ability for a long time, and there are a number of JavaScript libraries and frameworks that enable module usage (for example, other CommonJS and AMD-based module systems like RequireJS, and more recently Webpack and Babel).

The good news is that modern browsers have started to support module functionality natively, and this is what this article is all about. This can only be a good thing — browsers can optimize loading of modules, making it more efficient than having to use a library and do all of that extra client-side processing and extra round trips.

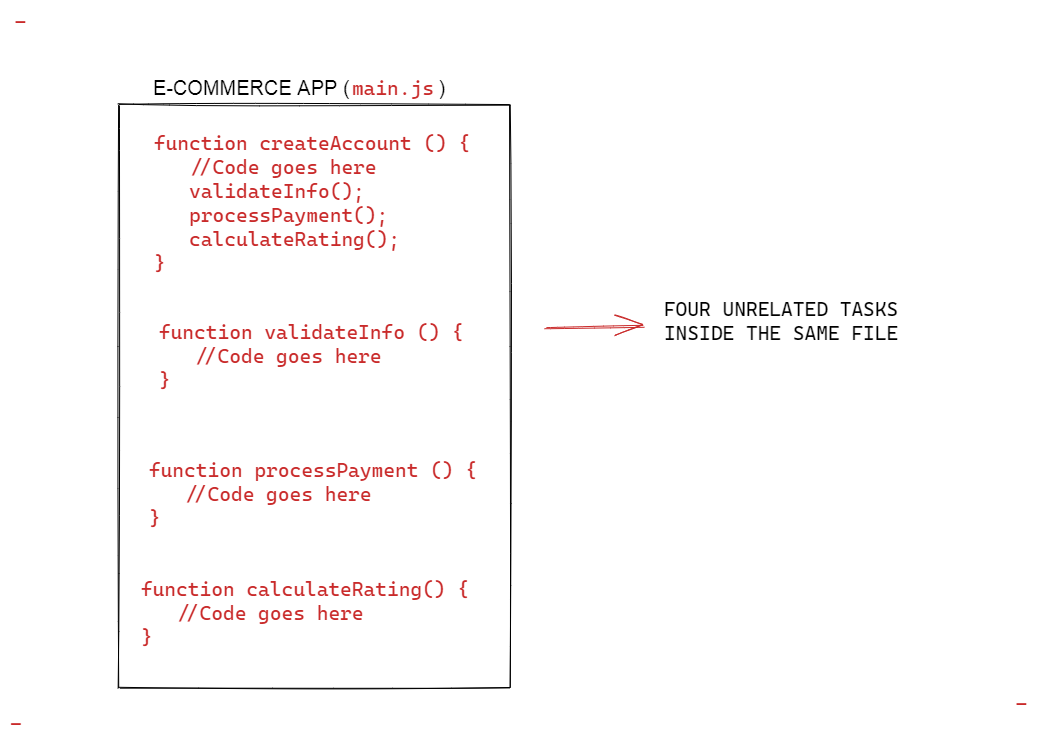
Use of native JavaScript modules is dependent on the import and export statements; these are supported in browsers as shown in the compatibility table below.

To apply modules into HTML



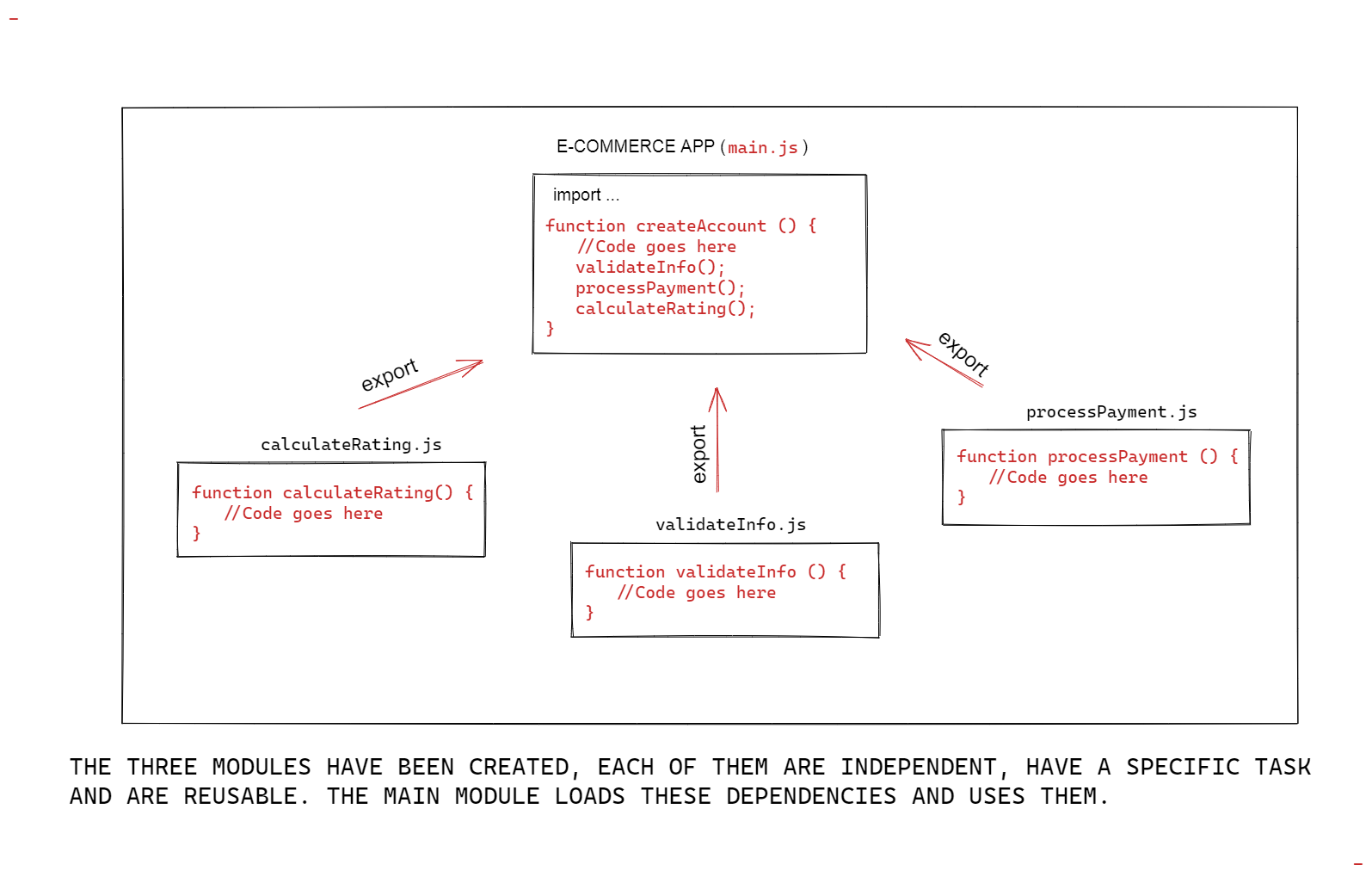
Modules should be:

1. **Independent/Self-contained:** A module has to be as detached from other dependencies as possible.
2. **Specific:** A module needs to be able to perform a single or a related group of tasks. The core essence of creating them in the first place is to create separate functionalities. One module, one (kind of) task.
3. **Reusable:** A module has to be easy to integrate into various kinds of programs to perform its task.

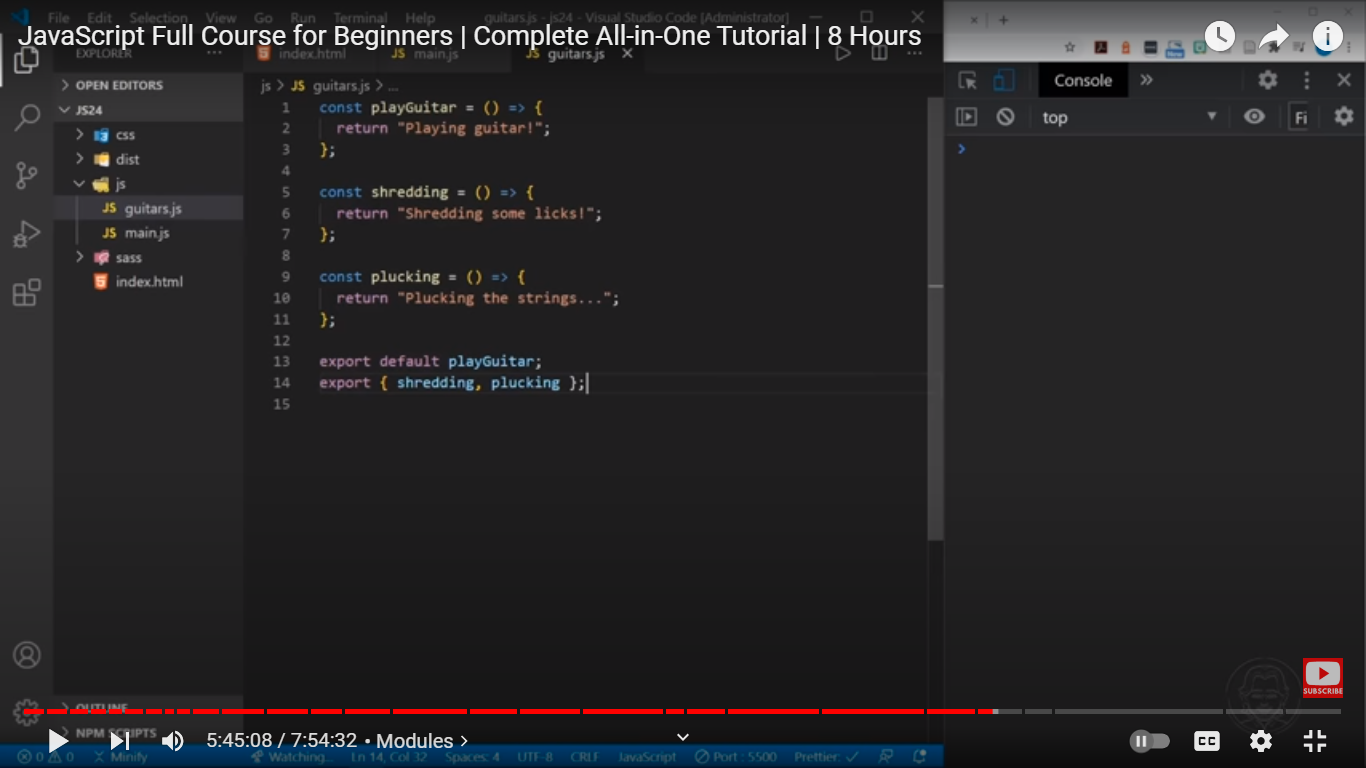


Instead of having all of those unrelated programs together in one module/file, it is a better practice to create several files, or modules, for each of those tasks. In such a case, the modules become dependencies.

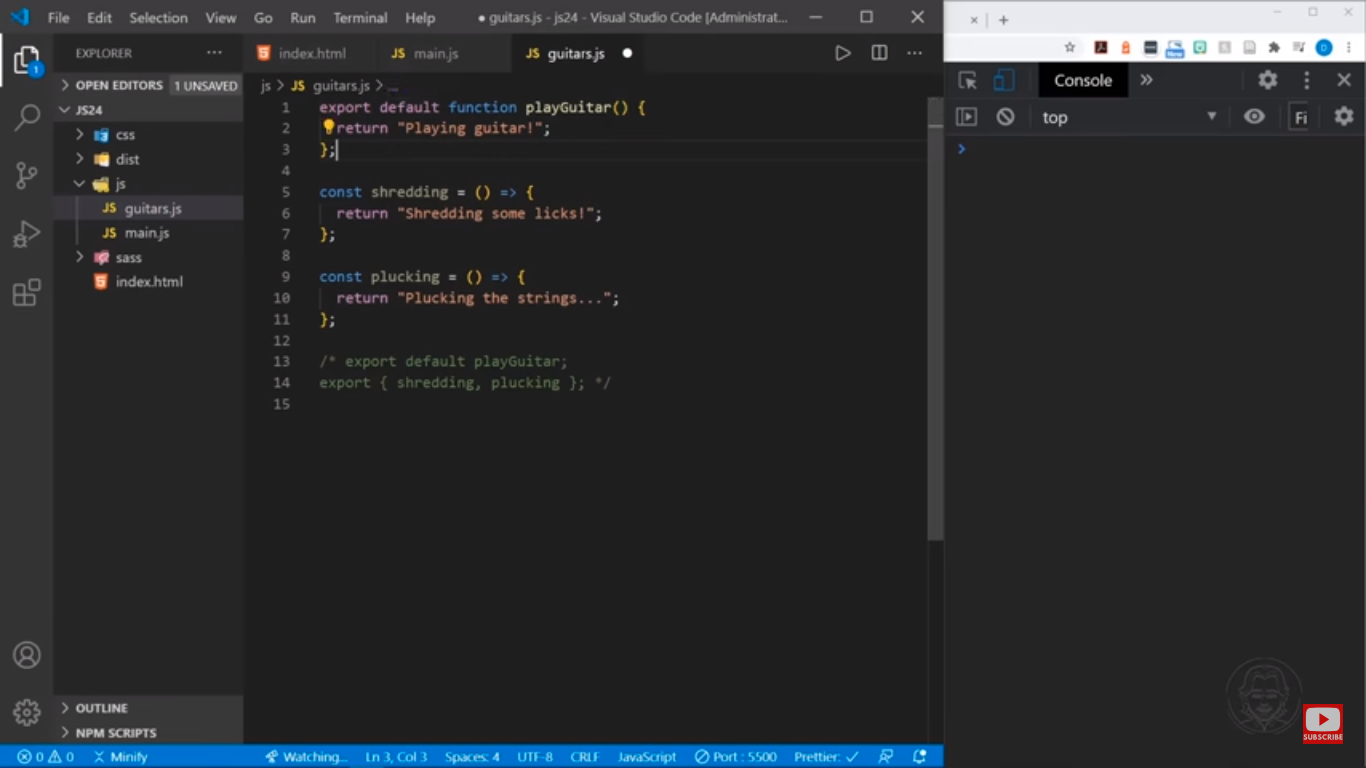
Then from the main app or program, you simply import/load the dependencies (i.e the modules you need) and execute them accordingly. As a result, your main app becomes cleaner and more minimal.

main.js has been broken down into four modules

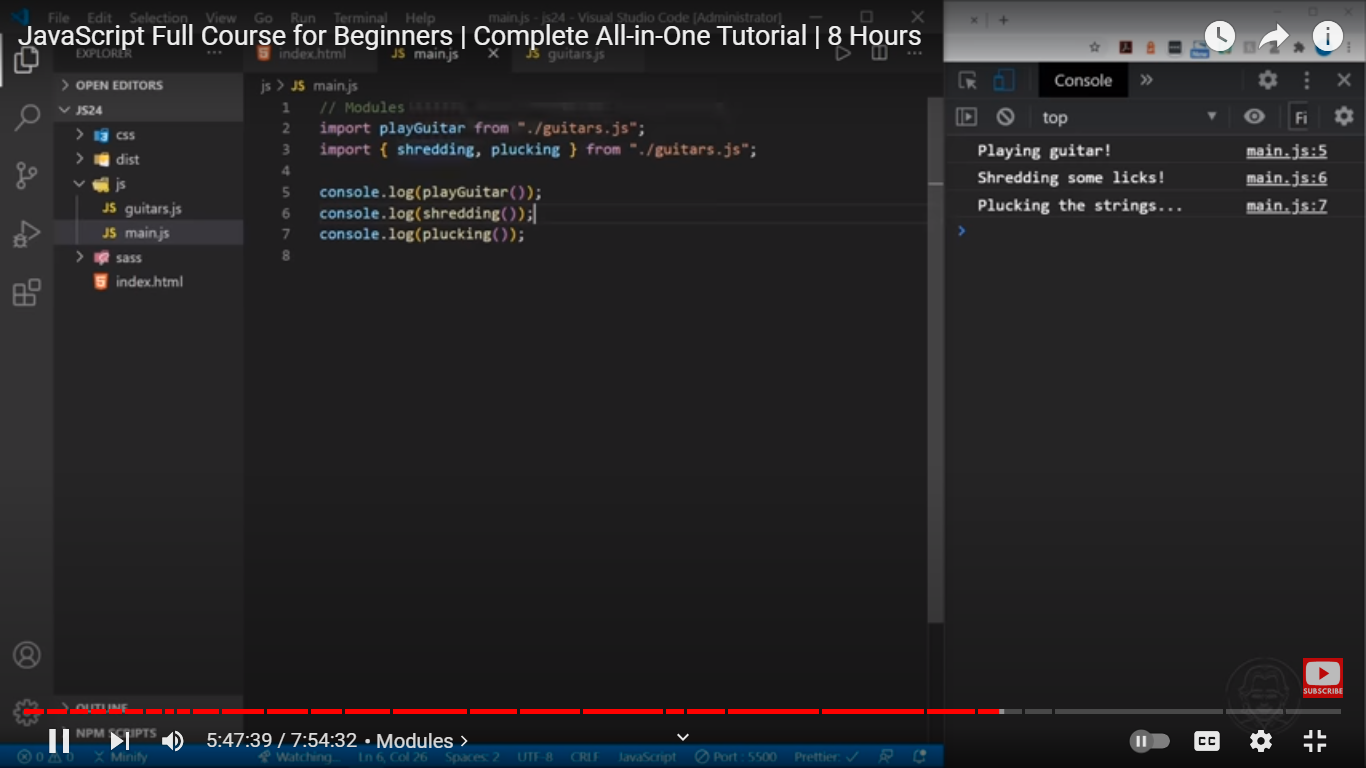
Exporting all modules

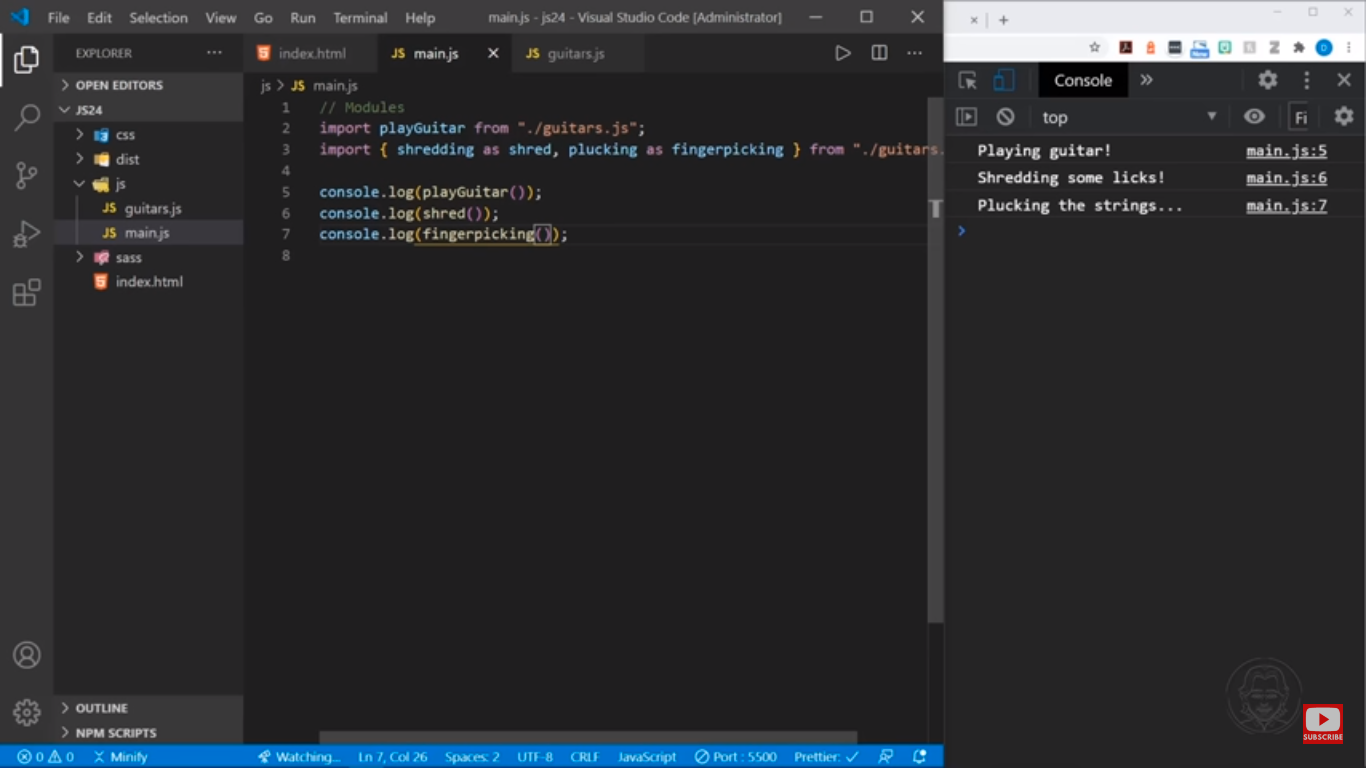


Exporting indiviual modul in js

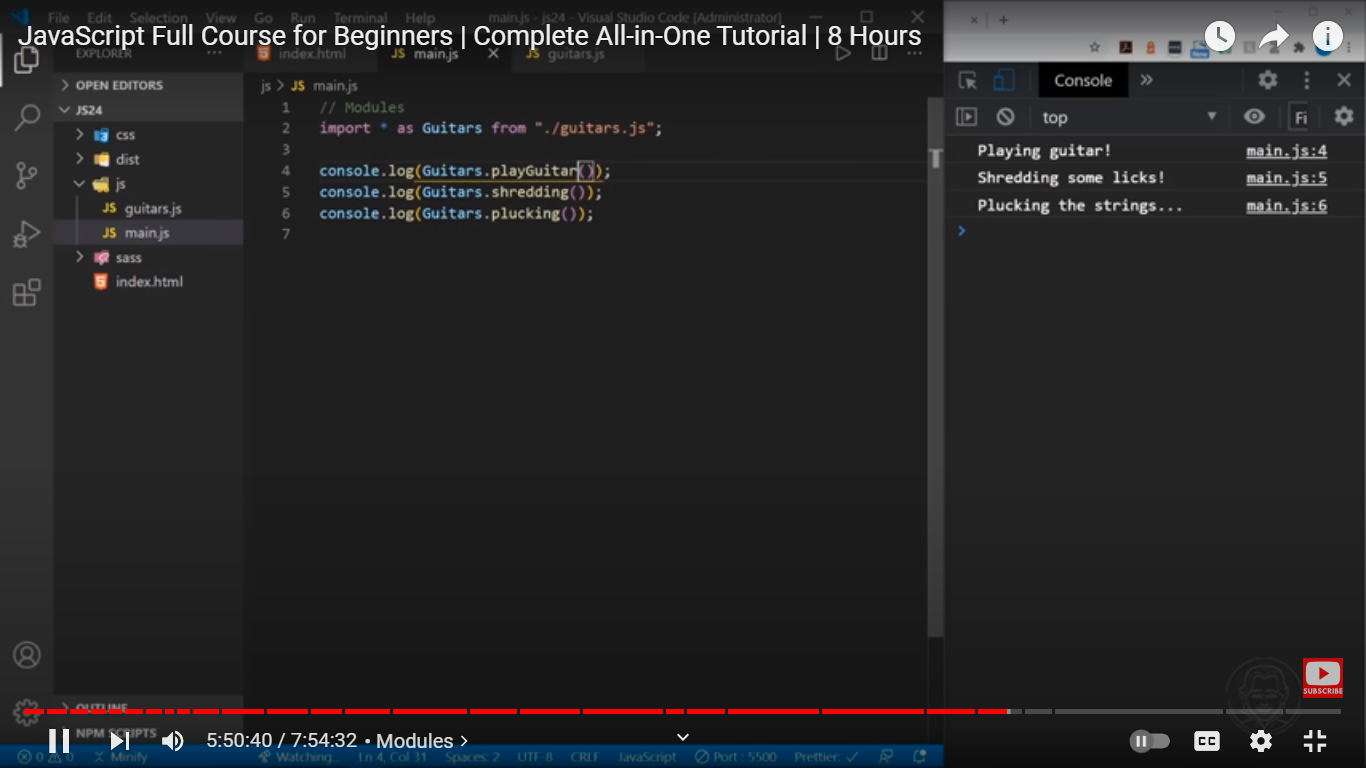
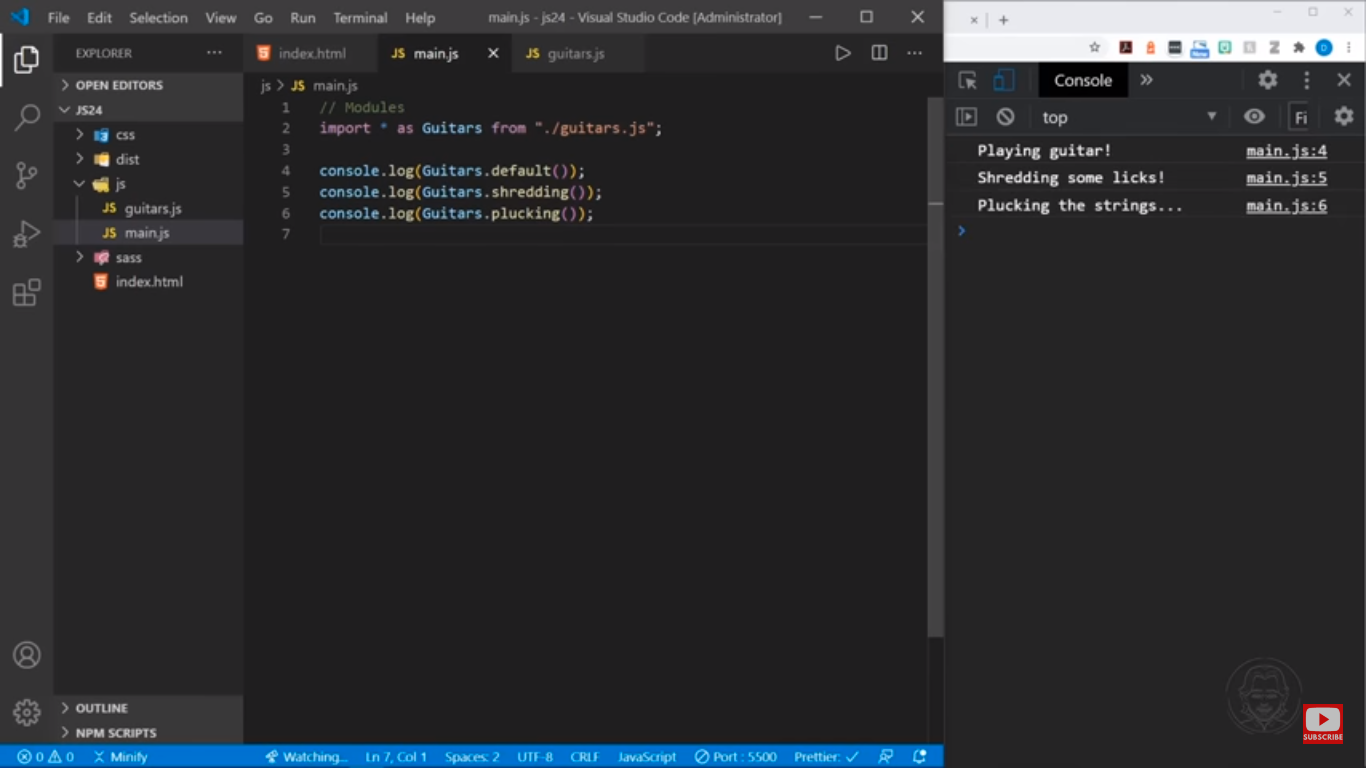


Importing modules





Import all the modules



Example

