* - [Instructor] At this point in the course, I've shown you all of the TypeScript syntax that you'll need to know in order to use TypeScript successfully in your day to day development.
* In this chapter, I'm going to shift the focus to using the syntax you've already learned, but in a different way.
* And in fact, I'm going to start by going back to a JavaScript feature, **modules**.
* Put simply, JavaScript modules enable you **to link code that is distributed across many files**.
* Let me demonstrate.
* Here is an old school JavaScript web application stored in several files.
* And in our webpage, we use HTML script tags to download and load each of these files into the browser's memory.
* As we do, the ***contents of each file get loaded into the same shared memory space.***
* *That means that when I define a function in one file and then define another function in another file with the* ***same name****, and then load both of these files like I'm doing here, the last function to be defined* ***overwrites*** *any others defined before it.*
* With modules, however, **each file shares its own memory space**, so any function or variables that are defined within that module stay within that memory space and do not leak out, avoiding these kinds of collisions altogether.
* What's more, **modules can load from other modules.**
* And since just about all modern browsers understand modules, they're even able to download these other modules on the fly based only on these kinds of references.
* This means that we no longer have to worry about loading every file of our application up front, but instead, let the browser load them for us as they're needed.
* Now that I've shown modules in action, check out the next video, where I take a few steps back and show you how to write module-based code from scratch and how TypeScript can help you do it.