ES6 - Understand the Differences Between import and require

In the past, the function require()would be used to import the functions and code in external files and modules. While handy, this presents a problem: some files and modules are rather large, and you may only need certain code from those external resources.

ES6 gives us a very handy tool known as *import*. With it, we can choose which parts of a module or file to load into a given file, saving time and memory.

Consider the following example. Imagine that math\_array\_functions has about 20 functions, but I only need one, countItems, in my current file. The old require()approach would force me to bring in all 20 functions. With this new import syntax, I can bring in just the desired function, like so:

import { countItems } from "math\_array\_functions"

A description of the above code:

import { function } from "file\_path\_goes\_here"  
// We can also import variables the same way!

There are a few ways to write an import statement, but the above is a very common use-case.

**Note**  
The whitespace surrounding the function inside the curly braces is a best practice - it makes it easier to read the import statement.

**Note**  
The lessons in this section handle non-browser features. import, and the statements we introduce in the rest of these lessons, won't work on a browser directly. However, we can use various tools to create code out of this to make it work in browser.

**Note**  
In most cases, the file path requires a ./before it; otherwise, node will look in the node\_modules directory first trying to load it as a dependency.

Add the appropriate import statement that will allow the current file to use the capitalizeString function. The file where this function lives is called "string\_functions", and it is in the same directory as the current file.

Run the TestsReset All CodeAsk for help

valid import statement

"use strict";

capitalizeString("hello!");