### **Differences between import and require:**

const ScenarioHelper = require("../libs/ScenarioHelper");

const { sleep } = require('../commonLib.js');

import { sleep } from '../commonLib.js'

1. Module System:
   * require: Associated with CommonJS module system, which is the module system used in Node.js by default.
   * import: Associated with ES modules (ES6), which is the standardized module system introduced in ECMAScript 6 and supported by modern browsers and newer versions of Node.js.
2. Syntax:
   * require: Uses require() function to import modules. It's a function call and the module path is passed as a string.
   * import: Uses import keyword followed by the module specifier enclosed in curly braces {} or a default import without braces. The module specifier can be a string or a file path.
3. Static vs. Dynamic:
   * require: Works dynamically. Modules are loaded synchronously at runtime.
   * import: Works statically. Modules are loaded statically at compile-time (in the case of bundlers like webpack) or during module resolution (in the case of native ES module support).
4. Hoisting:
   * require: Does not hoist module imports. They are evaluated at runtime, so the order of require calls matters.
   * import: Hoists module imports to the top of the file. They are evaluated before any code is executed, so the order of import statements doesn't matter.
5. Default vs. Named Exports:
   * require: Supports both default and named exports, but you need to access named exports using property notation.
   * import: Supports default imports (import moduleName from 'module') and named imports (import { namedExport } from 'module') separately.
6. Dynamic Imports:
   * require: Does not support dynamic imports natively in Node.js (as of my last update). However, dynamic imports can be achieved using require with a string concatenation or via libraries like import-fresh.
   * import: Supports dynamic imports using the import() function, which allows you to import modules conditionally or asynchronously.
7. CommonJS vs. ES Modules:
   * require: CommonJS modules are loaded synchronously and have access to module.exports.
   * import: ES modules are loaded statically and have access to export and import keywords for defining and importing/exporting functionality.

In summary, require is associated with the CommonJS module system used in Node.js, while import is associated with the standardized ES modules system introduced in ECMAScript 6. They have different syntaxes, behaviors, and capabilities, but both serve the purpose of importing modules into JavaScript code.