```
import log = require("log");
log.message("hello");
```

File log.ts:

```
export function message(s: string) {
    console.log(s);
}
```

The import declaration in the 'main' module references the 'log' module and compiling the 'main.ts' file causes the 'log.ts' file to also be compiled as part of the program. At run-time, the import declaration loads the 'log' module and produces a reference to its module instance through which it is possible to reference the exported function.

TypeScript supports two patterns of JavaScript code generation for external modules: The CommonJS Modules pattern (section 11.2.5), typically used by server frameworks such as node.js, and the Asynchronous Module Definition (AMD) pattern (section 11.2.6), an extension to CommonJS Modules that permits asynchronous module loading, as is typical in browsers. The desired module code generation pattern is selected through a compiler option and does not affect the TypeScript source code. Indeed, it is possible to author external modules that can be compiled for use both on the server side (e.g. using node.js) and on the client side (using an AMD compliant loader) with no changes to the TypeScript source code.

### 11.2.1 External Module Names

External modules are identified and referenced using external module names. The following definition is aligned with that provided in the CommonJS Modules 1.0 specification.

- An external module name is a string of "terms" delimited by forward slashes.
- External module names may not have file-name extensions like ".js".
- External module names may be "relative" or "top-level". An external module name is "relative" if the first term is "." or "..".
- Top-level names are resolved off the conceptual module name space root.
- Relative names are resolved relative to the name of the module in which they occur.

For purposes of resolving external module references, TypeScript associates a file path with every external module. The file path is simply the path of the module's source file without the file extension. For example, an external module contained in the source file 'C:\src\lib\io.ts' has the file path 'C:/src/lib/io' and an external module contained in the source file 'C:\src\ui\editor.d.ts' has the file path 'C:/src/ui/editor'.

An external module name in an import declaration is resolved as follows:

- If the import declaration specifies a relative external module name, the name is resolved relative to the directory of the referencing module's file path. The program must contain a module with the resulting file path or otherwise an error occurs. For example, in a module with the file path