ExportAssignment:

`export` = *Identifier* `;`

The *Identifier* of an export assignment must name one or more entities declared at the top level in the external module. When an external module containing an export assignment is imported, the local alias introduced by the external import declaration takes on all meanings of the identifier named in the export assignment.

It is an error for an external module to contain more than one export assignment.

Assume the following example resides in the file 'point.ts':

```
export = Point;

class Point {
    constructor(public x: number, public y: number) { }
    static origin = new Point(0, 0);
}
```

When 'point.ts' is imported in another external module, the import alias references the exported class and can be used both as a type and as a constructor function:

```
import Pt = require("point");

var p1 = new Pt(10, 20);
var p2 = Pt.origin;
```

Note that there is no requirement that the import alias use the same name as the exported entity.

11.2.5 CommonJS Modules

The CommonJS Modules definition specifies a methodology for writing JavaScript modules with implied privacy, the ability to import other modules, and the ability to explicitly export members. A CommonJS compliant system provides a 'require' function that can be used to synchronously load other external modules to obtain their singleton module instance, as well as an 'exports' variable to which a module can add properties to define its external API.

The 'main' and 'log' example from section 11.2 above generates the following JavaScript code when compiled for the CommonJS Modules pattern:

File main.js:

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

File log.js: