```
exports.message = function(s) {
    console.log(s);
}
```

An external import declaration is represented in the generated JavaScript as a variable initialized by a call to the 'require' function provided by the module system host. A variable declaration and 'require' call is emitted for a particular imported module only if the imported module is referenced as a *PrimaryExpression* somewhere in the body of the importing module. If an imported module is referenced only as a *ModuleName* or *TypeQueryExpression*, nothing is emitted.

TODO: Runtime dependency generation works transitively, i.e. if 'bar' is a module alias that references a module alias 'foo' and 'bar' is referenced in a value position, then 'foo' is considered referenced in a value position too.

An example:

File geometry.ts:

```
export interface Point { x: number; y: number };

export function point(x: number, y: number): Point {
    return { x: x, y: y };
}
```

File game.ts:

```
import g = require("geometry");
var p = g.point(10, 20);
```

The 'game' module references the imported 'geometry' module in an expression (through its alias 'g') and a 'require' call is therefore included in the emitted JavaScript:

```
var g = require("geometry");
var p = g.point(10, 20);
```

Had the 'game' module instead been written to only reference 'geometry' in a type position

```
import g = require("geometry");
var p: g.Point = { x: 10, y: 20 };
```

the emitted JavaScript would have no dependency on the 'geometry' module and would simply be

```
var p = { x: 10, y: 20 };
```

11.2.6 AMD Modules

The Asynchronous Module Definition (AMD) specification extends the CommonJS Modules specification with a pattern for authoring asynchronously loadable modules with associated dependencies. Using the AMD pattern, modules are emitted as calls to a global 'define' function taking an array of dependencies,