specified as external module names, and a callback function containing the module body. The global 'define' function is provided by including an AMD compliant loader in the application. The loader arranges to asynchronously load the module's dependencies and, upon completion, calls the callback function passing resolved module instances as arguments in the order they were listed in the dependency array.

The "main" and "log" example from above generates the following JavaScript code when compiled for the AMD pattern.

File main.js:

```
define(["require", "exports", "log"], function(require, exports, log) {
    log.message("hello");
}
```

File log.js:

```
define(["require", "exports"], function(require, exports) {
    exports.message = function(s) {
        console.log(s);
    }
}
```

The special 'require' and 'exports' dependencies are always present. Additional entries are added to the dependencies array and the parameter list as required to represent imported external modules. Similar to the code generation for CommonJS Modules, a dependency entry is generated 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*, no dependency is generated for that module.

11.3 Code Generation

TODO: Finish this section.