DeclarationElement:
`export`$_{opt}$ *InterfaceDeclaration*
`export`$_{opt}$ *ImportDeclaration*
ExportAssignment
`export`$_{opt}$ *ExternalImportDeclaration*
`export`$_{opt}$ *AmbientDeclaration*

When a TypeScript program is compiled, all of the program's source files are processed together. Statements and declarations in different source files can depend on each other, possibly in a circular fashion. By default, a JavaScript output file is generated for each implementation source file in a compilation, but no output is generated from declaration source files.

The source elements permitted in a TypeScript implementation source file are a superset of those supported by JavaScript. Specifically, TypeScript extends the JavaScript grammar's existing *VariableDeclaration* (section 5.1) and *FunctionDeclaration* (section 6.1) productions, and adds *InterfaceDeclaration* (section 7.1), *ClassDeclaration* (section 8.1), *EnumDeclaration* (section 9.1), *ModuleDeclaration* (section 10.1), *ImportDeclaration* (section 10.3), *ExternalImportDeclaration* (section 11.2.2), *ExportAssignment* (section 11.2.4), and *AmbientDeclaration* (section 12.1) productions.

Declaration source files are restricted to contain declarations only. Declaration source files can be used to declare the static type information associated with existing JavaScript code in an adjunct manner. They are entirely optional but enable the TypeScript compiler and tools to provide better verification and assistance when integrating existing JavaScript code and libraries in a TypeScript application.

Implementation and declaration source files that contain no import or export declarations form the single ***global module***. Entities declared in the global module are in scope everywhere in a program. Initialization order of the source files that make up the global module ultimately depends on the order in which the generated JavaScript files are loaded at run-time (which, for example, may be controlled by <script/> tags that reference the generated JavaScript files).

Implementation and declaration source files that contain at least one external import declaration, export assignment, or top-level exported declaration are considered separate ***external modules***. Entities declared in an external module are in scope only in that module, but exported entities can be imported into other modules using import declarations. Initialization order of external modules is determined by the module loader being and is not specified by the TypeScript language. However, it is generally the case that non-circularly dependent modules are automatically loaded and initialized in the correct order.

External modules can additionally be declared using *AmbientModuleDeclarations* in the global module that directly specify the external module names as string literals. This is described further in section 12.1.6.

11.1.1 Source Files Dependencies

The TypeScript compiler automatically determines a source file's dependencies and includes those dependencies in the program being compiled. The determination is made from "reference comments" and external import declarations as follows: