Appendix C. Scoped Declarations

TypeScript declarations have a rich set of behaviors needed to model types and values, and as in JavaScript, they can be overloaded in a variety of ways. This appendix covers two of these behaviors, summarizing which declarations generate types (and which generate values), and which declarations can be merged.

Does It Generate a Type?

Some TypeScript declarations create a type, some create a value, and some create both. See to Table C-1 for a quick reference.

Table C-1. Does the declaration generate a type?

Keyword	Generates a type?	Generates a value?
`class`	Yes	Yes
`const`, `let`, `var`	No	Yes
`enum`	Yes	Yes
`function`	No	Yes
`interface`	Yes	No
`namespace`	No	Yes
`type`	Yes	No

Does It Merge?

Declaration merging is a core TypeScript behavior. Take advantage of it to create richer APIs, better modularize your code, and make your code safer.

Table C-2 is reprinted from "Declaration Merging"; it's a handy reference for which kinds of declarations TypeScript will merge for you.

Table C-2. Can the declaration be merged?

		To				
		Value	Class	Enum	Function	Types alias
From	Value	No	No	No	No	Yes
	Class	—	No	No	No	No
	Enum	—	—	Yes	No	No
	Function	—	—	—	No	Yes
	Type alias	—	—	—	—	No
	Interface	—	—	—	—	—
	Namespace	—	—	—	—	—
	Module	—	—	—	—	—