3.6.2 Type References

A type reference references a named type or type parameter through its name and, in the case of a generic type, supplies a type argument list.

TypeReference:
    TypeName *TypeArguments*$_{opt}$

TypeName:
    Identifier
    ModuleName . *Identifier*

ModuleName:
    Identifier
    ModuleName . *Identifier*

A *TypeReference* consists of a *TypeName* that a references a named type or type parameter. A reference to a generic type must be followed by a list of *TypeArguments* (section 3.4.2).

Resolution of a *TypeName* consisting of a single identifier is described in section 2.4.

Resolution of a *TypeName* of the form *M.N*, where *M* is a *ModuleName* and *N* is an *Identifier*, proceeds by first resolving the module name *M*. If the resolution of *M* is successful and the resulting module contains an exported named type *N*, then *M.N* refers to that member. Otherwise, *M.N* is undefined.

Resolution of a *ModuleName* consisting of a single identifier is described in section 2.4.

Resolution of a *ModuleName* of the form *M.N*, where *M* is a *ModuleName* and *N* is an *Identifier*, proceeds by first resolving the module name *M*. If the resolution of *M* is successful and the resulting module contains an exported module member *N*, then *M.N* refers to that member. Otherwise, *M.N* is undefined.

A type reference to a generic type is required to specify exactly one type argument for each type parameter of the referenced generic type, and each type argument must be assignable to (section 3.8.4) the constraint of the corresponding type parameter or otherwise an error occurs. An example:

```
interface A { a: string; }

interface B extends A { b: string; }

interface C extends B { c: string; }

interface G<T, U extends B> {
    x: T;
    y: U;
}
```