- Properties in a class declaration may be designated public or private, while properties declared in other contexts are always considered public. Private members are only accessible within the class body containing their declaration, as described in section 8.2.2, and private properties match only themselves in subtype and assignment compatibility checks, as described in section 3.8.
- Properties in an object type literal or interface declaration may be designated required or optional, while properties declared in other contexts are always considered required. Properties that are optional in the target type of an assignment may be omitted from source objects, as described in section 3.8.4.

Call and construct signatures may be ***specialized*** (section 3.7.2.4) by including parameters with string literal types. Specialized signatures are used to express patterns where specific string values for some parameters cause the types of other parameters or the function result to become further specialized.

3.4 Type Parameters

A type parameter represents an actual type that the parameter is bound to in a generic type reference or a generic function call. Type parameters have constraints that establish upper bounds for their actual type arguments.

Since a type parameter represents a multitude of different type arguments, type parameters have certain restrictions compared to other types. In particular, a type parameter cannot be used as a base class or interface.

3.4.1 Type Parameter Lists

Class, interface, and function declarations may optionally include lists of type parameters enclosed in < and > brackets. Type parameters are also permitted in call signatures of object, function, and constructor type literals.

TypeParameters:
    `<` *TypeParameterList* `>`

TypeParameterList:
    TypeParameter
    TypeParameterList `,` *TypeParameter*

TypeParameter:
    Identifier *Constraint*$_{opt}$

Constraint:
    `extends` *Type*

Type parameter names must be unique. A compile-time error occurs if two or more type parameters in the same *TypeParameterList* have the same name.