The scope of a type parameter extends over the entire declaration with which the type parameter list is associated, with the exception of static member declarations in classes.

Each type parameter has an associated type parameter ***constraint*** that establishes an upper bound for type arguments. Omitting a constraint corresponds to specifying the global interface type 'Object'. Type parameters may be referenced in type parameter constraints within the same type parameter list, including even constraint declarations that occur to the left of the type parameter.

The ***base constraint*** of a type parameter *T* is defined as follows:

- If *T* has no declared constraint, *T*'s base constraint is the global interface type 'Object'.
- If *T*'s declared constraint is a type parameter, *T*'s base constraint is that of the type parameter.
- Otherwise, *T*'s base constraint is *T*'s declared constraint.

In the example

```
interface G<T, U extends V, V extends Function> { }
```

the base constraint of 'T' is 'Object' and the base constraint of 'U' and 'V' is 'Function'.

For purposes of determining type relationships (section 3.8), type parameters appear to be subtypes of their base constraint. Likewise, in property accesses (section 4.10), `new` operations (section 4.11), and function calls (section 4.12), type parameters appear to have the members of their base constraint, but no other members.

It is an error for a type parameter to directly or indirectly be a constraint for itself. For example, both of the following declarations are invalid:

```
interface A<T extends T> { }

interface B<T extends U, U extends T> { }
```

3.4.2 Type Argument Lists

A type reference (section 3.6.2) to a generic type must include a list of type arguments enclosed in angle brackets and separated by commas. Similarly, a call (section 4.12) to a generic function may explicitly include a type argument list instead of relying on type inference.

TypeArguments:
    `<` *TypeArgumentList* `>`

TypeArgumentList:
    TypeArgument
    TypeArgumentList `,` *TypeArgument*

TypeArgument:
    Type