- if *S* and *T* are type references to the same named type, inferences are made from each type argument in *S* to each type argument in *T*,
- otherwise, no inferences are made.

Referring to the example above, the 'List<T>' type reference in the 'next' property is classified as an open type reference and the 'List<List<T>>' type reference in the 'owner' property is classified as an infinitely expanding type reference.

3.9 Widened Types

In several situations TypeScript infers types from context, alleviating the need for the programmer to explicitly specify types that appear obvious. For example

```
var name = "Steve";
```

infers the type of 'name' to be the String primitive type since that is the type of the value used to initialize it. When inferring the type of a variable, property or function result from an expression, the ***widened*** form of the source type is used as the inferred type of the target. The widened form of a type is the type in which all occurrences of the Null and Undefined types have been replaced with the type any.

The following example shows the results of widening types to produce inferred variable types.

```
var a = null;                   // var a: any
var b = undefined;              // var b: any
var c = { x: 0, y: null };      // var c: { x: number, y: any }
var d = [ null, undefined ];    // var d: any[]
```

3.10 Best Common Type

In some cases a ***best common type*** needs to be inferred from a set of types. In particular, return types of functions with multiple return statements and element types of array literals are found this way.

For an empty set of types, the best common type is an empty object type (the type {}).

For a non-empty set of types { T_1, T_2, ..., T_n }, the best common type is the one T_x in the set that is a supertype of every T_n. It is possible that no such type exists or more than one such type exists, in which case the best common type is an empty object type.