```
var x = 1;
var x: number;
if (x == 1) {
    var x = 2;
}
```

In the following example, all five variables are of the same type, '{ x: number; y: number; }'.

```
interface Point { x: number; y: number; }

var a = { x: 0, y: <number> undefined };
var b: Point = { x: 0; y: undefined };
var c = <Point> { x: 0, y: undefined };
var d: { x: number; y: number; } = { x: 0, y: undefined };
var e = <{ x: number; y: number; }> { x: 0, y: undefined };
```

5.2 If, Do, and While Statements

Expressions controlling 'if', 'do', and 'while' statements can be of any type (and not just type Boolean).

5.3 For Statements

Variable declarations in 'for' statements are extended in the same manner as variable declarations in variable statements (section 5.1).

5.4 For-In Statements

In a 'for-in' statement of the form

for (*Var* in *Expr*) *Statement*

Var must be an expression classified as a reference of type Any or the String primitive type, and *Expr* must be an expression of type Any, an object type, or a type parameter type.

In a 'for-in' statement of the form

for (var *VarDecl* in *Expr*) *Statement*

VarDecl must be a variable declaration without a type annotation that declares a variable of type Any, and *Expr* must be an expression of type Any, an object type, or a type parameter type.

5.5 Continue Statements

A 'continue' statement is required to be nested, directly or indirectly (but not crossing function boundaries), within an iteration ('do', 'while', 'for', or 'for-in') statement. When a 'continue' statement includes a target label, that target label must appear in the label set of an enclosing (but not crossing function boundaries) iteration statement.