Instance and static members in a class are likewise in separate declaration spaces. Thus the following is permitted:

```
class C {
    x: number;          // Instance member
    static x: string;   // Static member
}
```

2.4 Scopes

The ***scope*** of a name is the region of program text within which it is possible to refer to the entity declared by that name without qualification of the name. The scope of a name depends on the context in which the name is declared. The contexts are listed below in order from outermost to innermost:

- The scope of an entity declared in the global module is the entire program text.
- The scope of an entity declared in an external module is the source file of that external module.
- The scope of an exported entity declared in an internal module is the body of that module and every internal module with the same root and the same qualified name relative to that root.
- The scope of a non-exported entity declared within an internal module declaration is the body of that internal module declaration.
- The scope of a type parameter declared in a class or interface declaration is that entire declaration, including constraints, extends clause, implements clause, and declaration body, but not including static member declarations.
- The scope of a member declared in an enum declaration is the body of that declaration and every enum declaration with the same root and the same qualified name relative to that root.
- The scope of a type parameter declared in a call or construct signature is that entire signature declaration, including constraints, parameter list, and return type. If the signature is part of a function implementation, the scope includes the function body.
- The scope of a parameter, local variable, or local function declared within a function declaration (including a constructor, member function, or member accessor declaration) or function expression is the body of that function declaration or function expression.

Scopes may overlap, for example through nesting of modules and functions. When the scopes of two entities with the same name overlap, the entity with the innermost declaration takes precedence and access to the outer entity is either not possible or only possible by qualifying its name.

When an identifier is resolved as a *TypeName* (section 3.6.2), only classes, interfaces, enums, and type parameters are considered and other entities in scope are ignored.

When an identifier is resolved as a *ModuleName* (section 3.6.2), only modules are considered and other entities in scope are ignored.

When an identifier is resolved as a *PrimaryExpression* (section 4.3), only instantiated modules (section 10.1), classes, enums, functions, variables, and parameters are considered and other entities in scope are ignored.