8 Classes

TypeScript supports classes that are closely aligned with those proposed for ECMAScript 6, and includes extensions for instance and static member declarations and properties declared and initialized from constructor parameters.

NOTE: TypeScript currently doesn't support class expressions or nested class declarations from the ECMAScript 6 proposal.

8.1 Class Declarations

Class declarations introduce named types and provide implementations of those types. Classes support inheritance, allowing derived classes to extend and specialize base classes.

ClassDeclaration:
`class` *Identifier* *TypeParameters*$_{opt}$ *ClassHeritage* `{` *ClassBody* `}`

A *ClassDeclaration* declares a ***class type*** and a ***constructor function***, both with the name given by *Identifier*, in the containing module. The class type is created from the instance members declared in the class body and the instance members inherited from the base class. The constructor function is created from the constructor declaration, the static member declarations in the class body, and the static members inherited from the base class. The constructor function initializes and returns an instance of the class type.

The *Identifier* of a class declaration may not be one of the predefined type names (section 3.6.1).

A class may optionally have type parameters (section 3.4.1) that serve as placeholders for actual types to be provided when the class is referenced in type references. A class with type parameters is called a ***generic class***. The type parameters of a generic class declaration are in scope in the entire declaration and may be referenced in the *ClassHeritage* and *ClassBody*.

The following example introduces both a named type called 'Point' (the class type) and a member called 'Point' (the constructor function) in the containing module.

```
class Point {
    constructor(public x: number, public y: number) { }
    public length() { return Math.sqrt(this.x * this.x + this.y * this.y);
    static origin = new Point(0, 0);
}
```

The 'Point' type is exactly equivalent to