```
interface Point {
    x: number;
    y: number;
    length(): number;
}
```

The 'Point' member is a constructor function whose type corresponds to the declaration

```
var Point: {
    new(x: number, y: number): Point;
    origin: Point;
};
```

The context in which a class is referenced distinguishes between the class instance type and the constructor function. For example, in the assignment statement

```
var p: Point = new Point(10, 20);
```

the identifier 'Point' in the type annotation refers to the class instance type, whereas the identifier 'Point' in the `new` expression refers to the constructor function object.

8.1.1 Class Heritage Specification

The heritage specification of a class consists of optional `extends` and `implements` clauses. The `extends` clause specifies the base class of the class and the `implements` clause specifies a set of interfaces for which to validate the class provides an implementation.

ClassHeritage:
    ClassExtendsClause$_{opt}$ *ImplementsClause*$_{opt}$

ClassExtendsClause:
    `extends` *ClassType*

ClassType:
    TypeReference

ImplementsClause:
    `implements` *ClassOrInterfaceTypeList*

A class that includes an `extends` clause is called a ***derived class***, and the class specified in the `extends` clause is called the ***base class*** of the derived class. When a class heritage specification omits the `extends` clause, the class does not have a base class. However, as is the case with every object type, type references (section 3.3.1) to the class will appear to have the members of the global interface type named 'Object' unless those members are hidden by members with the same name in the class.

The following constraints must be satisfied by the class heritage specification or otherwise a compile-time error occurs: