- A property for each uniquely named static member accessor declaration in the class body.
- A property named 'prototype', the type of which is an instantiation of the class type with type Any supplied as a type argument for each type parameter.
- All base class constructor function type properties that are not overridden in the class.

Every class automatically contains a static property member named 'prototype', the type of which is the containing class with type Any substituted for each type parameter.

The example

```
class Pair<T1, T2> {
    constructor(public item1: T1, public item2: T2) { }
}

class TwoArrays<T> extends Pair<T[], T[]> { }
```

introduces two named types corresponding to

```
interface Pair<T1, T2> {
    item1: T1;
    item2: T2;
}

interface TwoArrays<T> {
    item1: T[];
    item2: T[];
}
```

and two constructor functions corresponding to

```
var Pair: {
    new <T1, T2>(item1: T1, item2: T2): Pair<T1, T2>;
}

var TwoArrays: {
    new <T>(item1: T[], item2: T[]): TwoArrays<T>;
}
```

Note that the construct signatures in the constructor function types have the same type parameters as their class and return the instance type of their class. Also note that when a derived class doesn't declare a constructor, type arguments from the base class reference are substituted before construct signatures are propagated from the base constructor function type to the derived constructor function type.

8.3 Constructor Declarations

A constructor declaration declares the constructor function of a class.

ConstructorDeclaration:
ConstructorOverloads$_{opt}$ *ConstructorImplementation*