ConstructorOverloads:
    ConstructorOverload
    ConstructorOverloads *ConstructorOverload*

ConstructorOverload:
    PublicOrPrivate$_{opt}$ `constructor` `(` *ParameterList*$_{opt}$ `)` `;`

ConstructorImplementation:
    PublicOrPrivate$_{opt}$ `constructor` `(` *ParameterList*$_{opt}$ `)` `{` *FunctionBody* `}`

A class may contain at most one constructor declaration. If a class contains no constructor declaration, an automatic constructor is provided, as described in section 8.3.3.

Overloads and the implementation of a constructor may include a public or private accessibility modifier, but only public constructors are supported and private constructors result in an error.

If a constructor declaration includes overloads, the overloads determine the construct signatures of the type given to the constructor function object, and the constructor implementation signature must be assignable to that type. Otherwise, the constructor implementation itself determines the construct signature. This exactly parallels the way overloads are processed in a function declaration (section 6.2).

The function body of a constructor is permitted to contain return statements. If return statements specify expressions, those expressions must be of types that are assignable to the instance type of the class.

The type parameters of a generic class are in scope and accessible in a constructor declaration.

8.3.1 Constructor Parameters

Similar to functions, only the constructor implementation (and not constructor overloads) can specify default value expressions for optional parameters. It is a compile-time error for such default value expressions to reference `this`. For each parameter with a default value, a statement that substitutes the default value for an omitted argument is included in the JavaScript generated for the constructor function.

A parameter of a *ConstructorImplementation* may be prefixed with a `public` or `private` modifier. This is called a ***parameter property declaration*** and is shorthand for declaring a property with the same name as the parameter and initializing it with the value of the parameter. For example, the declaration

```
class Point {
    constructor(public x: number, public y: number) {
        // Constructor body
    }
}
```

is equivalent to writing