4.6 Array Literals

In the absence of a contextual type, the type of an array literal is `C[]`, where *C* is the Undefined type (section 3.2.6) if the array literal is empty, or the best common type of the element expressions if the array literal is not empty.

When an array literal is contextually typed (section 4.19) by an object type containing a numeric index signature of type *T*, each element expression is contextually typed by *T* and the type of the array literal is the best common type of *T* and the types of the element expressions.

4.7 Parentheses

A parenthesized expression

`(` *Expression* `)`

has the same type and classification as the *Expression* itself. Specifically, if the contained expression is classified as a reference, so is the parenthesized expression.

4.8 The super Keyword

The `super` keyword can be used in expressions to reference base class properties and the base class constructor.

CallExpression: *(Modified)*
  …
  `super` `(` *ArgumentList*$_{opt}$ `)`
  `super` `.` *IdentifierName*

4.8.1 Super Calls

Super calls consist of the keyword `super` followed by an argument list enclosed in parentheses. Super calls are only permitted in constructors of derived classes, as described in section 8.3.2.

A super call invokes the constructor of the base class on the instance referenced by `this`. A super call is processed as a function call (section 4.12) using the construct signatures of the base class constructor function type as the initial set of candidate signatures for overload resolution. Type arguments cannot be explicitly specified in a super call. If the base class is a generic class, the type arguments used to process a super call are always those specified in the `extends` clause that references the base class.

The type of a super call expression is Void.

The JavaScript code generated for a super call is specified in section 8.6.2.

4.8.2 Super Property Access

A super property access consists of the keyword `super` followed by a dot and an identifier. Super property accesses are used to access base class member functions from derived classes and are permitted in