4 Expressions

This chapter describes the manner in which TypeScript provides type inference and type checking for JavaScript expressions. TypeScript's type analysis occurs entirely at compile-time and adds no run-time overhead to expression evaluation.

TypeScript's typing rules define a type for every expression construct. For example, the type of the literal 123 is the Number primitive type, and the type of the object literal { a: 10, b: "hello" } is { a: number; b: string; }. The sections in this chapter describe these rules in detail.

In addition to type inference and type checking, TypeScript augments JavaScript expressions with the following constructs:

- Optional parameter and return type annotations in function expressions.
- Default parameter values and rest parameters in function expressions.
- Arrow function expressions.
- Super calls and member access.
- Type assertions.

Unless otherwise noted in the sections that follow, TypeScript expressions and the JavaScript expressions generated from them are identical.

4.1 Values and References

Expressions are classified as ***values*** or ***references***. References are the subset of expressions that are permitted as the target of an assignment. Specifically, references are combinations of identifiers (section 4.3), parentheses (section 4.7), and property accesses (section 4.10). All other expression constructs described in this chapter are classified as values.

4.2 The this Keyword

The type of `this` in an expression depends on the location in which the reference takes place:

- In a constructor, instance member function, instance member accessor, or instance member variable initializer, `this` is of the class instance type of the containing class.
- In a static member function or static member accessor, the type of `this` is the constructor function type of the containing class.
- In a function declaration or a standard function expression, `this` is of type Any.
- In the global module, `this` is of type Any.

In all other contexts it is a compile-time error to reference `this`.

In the body of an arrow function expression, references to `this` are rewritten in the generated JavaScript code, as described in section 4.9.2.