# 2 Basic Concepts

The remainder of this document is the formal specification of the TypeScript programming language and is intended to be read as an adjunct to the ECMAScript Language Specification (specifically, the ECMA-262 Standard, 5th Edition). This document describes the syntactic grammar added by TypeScript along with the compile-time processing and type checking performed by the TypeScript compiler, but it only minimally discusses the run-time behavior of programs since that is covered by the ECMAScript specification.

## 2.1 Grammar Conventions

The syntactic grammar added by TypeScript language is specified throughout this document using the existing conventions and production names of the ECMAScript grammar. In places where TypeScript augments an existing grammar production it is so noted. For example:

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

The '*( Modified )*' annotation indicates that an existing grammar production is being replaced, and the '…' references the contents of the original grammar production.

## 2.2 Namespaces and Named Types

TypeScript supports ***named types*** that can be organized in hierarchical ***namespaces***. Namespaces are introduced by module declarations and named types are introduced by class, interface, and enum declarations. Named types are denoted by qualified names that extend from some root module (possibly the global module) to the point of their declaration. The example

```
module X {
    export module Y {
        export interface Z { }
    }
    export interface Y { }
}
```

declares two interface types with the qualified names 'X.Y.Z' and 'X.Y' relative to the root module in which 'X' is declared.

In a qualified type name all identifiers but the last one refer to namespaces and the last identifier refers to a named type. Named type and namespace names are in separate declaration spaces and it is therefore possible for a named type and a namespace to have the same name, as in the example above.