Introduction

This document serves to outline the general design principles we have based the TypeScript language on. While it is by no means exhaustive, it aims to summarize the rules by which we've made many of the decisions that have shaped the language. Some of these rules are subjective, and at times are at odds with each other; reaching the right balance and making the right exceptions is the essence of how successful programming languages are designed.

Goals

1. Statically identify constructs that are likely to be errors.
2. Provide a structuring mechanism for larger pieces of code.
3. Impose no runtime overhead on emitted programs.
4. Emit clean, idiomatic, recognizable JavaScript code.
5. Produce a language that is composable and easy to reason about.
6. Align with current and future ECMAScript proposals.
7. Preserve runtime behavior of all JavaScript code.
8. Avoid adding expression-level syntax.
9. Use a consistent, fully erasable, structural type system.
10. Be a cross-platform development tool.
11. Do not cause substantial breaking changes from TypeScript 1.0.

Non-goals

1. Exactly mimic the design of existing languages. Instead, use the behavior of JavaScript and the intentions of program authors as a guide for what makes the most sense in the language.
2. Aggressively optimize the runtime performance of programs. Instead, emit idiomatic JavaScript code that plays well with the performance characteristics of runtime platforms.
3. Apply a sound or "provably correct" type system. Instead, strike a balance between correctness and productivity.
4. Provide an end-to-end build pipeline. Instead, make the system extensible so that external tools can use the compiler for more complex build workflows.
5. Add or rely on run-time type information in programs, or emit different code based on the results of the type system. Instead, encourage programming patterns that do not require run-time metadata.
6. Provide additional runtime functionality or libraries. Instead, use TypeScript to describe existing libraries.
7. Introduce behaviour that is likely to surprise users. Instead have due consideration for patterns adopted by other commonly-used languages.