Ecma/TC39/2014/049

Types and Type Annotations

A Stage 0 Proposal

Jonathan Turner | Brian Terlson

GOALS

- Short term
  - Reserve syntax used by TypeScript, Flow, etc. for some form of annotation
  - Venue for collaboration among interested committee members
- Long term
  - Consensus on a shared syntax for many varied type annotation implementations
  - Consensus on a shared semantics for type checking or annotations

GOALS

- Additionally, a shared syntax for interface definitions for documenting API boundaries (.d.ts files)

Examples & Demo

Rationale: Why Type Annotations?

- Toolability
  - Closure
  - TypeScript
  - Flow
  - JSDoc
- Performance
  - Asm.js
  - Hidden classes/runtime type inference
- API specification
  - DefinitelyTyped/.d.ts
  - WebIDL
  - JSDoc
- Runtime checks/guarantees
  - Guards
  - Contracts

Rationale: Why Standardize?

- Unify syntax used by many varied types/annotations tools and libraries
- Various tools already exist using a common syntax
- Carve out syntax within which many parties can experiment
- Venue for collaboration on developing a common subset of semantics that supports varied approaches already shipped and more to come.
- .d.ts files see a lot of use in the community – standardizing would allow other tools to take advantage of this

Prior Art

- In JS
  - TypeScript, SafeTypeScript
  - Flow
  - Using different syntax: Closure Compiler, Traceur Types
- Elsewhere
  - Python PEP 3107 / Python 3.5 Plans
  -