Generating Closure Annotations from AtScript

*Status: (Draft)*

*Authors: misko@hevery.com*

*This document is published to the web as part of the public* [*Angular Design Docs*](https://drive.google.com/#folders/0BxgtL8yFJbacUnUxc3l5aTZrbVk) *folder*

# Objective

Integrate [AtScript](https://docs.google.com/document/d/11YUzC-1d0V1-Q3V0fQ7KSit97HnZoKVygDxpWzEYW0U/edit?usp=sharing) output with [ClosureCompiler](https://developers.google.com/closure/compiler/) as a backend.

# Background

AtScript is an extension to ES6 which adds types, metadata annotations, and [introspection](https://mail.google.com/mail/?extsrc=sync&client=h&plid=ACUX6DPYgibC0RAgKMqnK6QCyOs1OFElFeAJIuM) to JavaScript. One of the important goals of AtScript is to play nice with existing ecosystem, libraries, frameworks, and tools. One such tools / library is the Closure project. Closure has its own set of [type annotations in the form of comments](https://developers.google.com/closure/compiler/docs/js-for-compiler) which. One way to integrate the two is to allow the AtScript compiler to output a Closure comment annotate code. In this way a developer can chose to write any one file in AtScript and compile it together with the existing closure libraries.

# Prior Art

* Closure compiler:
  + PRO: Can be run without compilation / fast development
  + CON: Types are in the form of comments
* TypeScript
  + PRO: Types are first class citizens
  + CON: Requires compilation step; Type information not available at runtime.

# Detailed Design

**STRAWMAN INCOMPLETE PROPOSAL**

Below is an example of how code written in AtScript can be transpiled to Closure Compiler so that it can be further processed by it.

|  |  |
| --- | --- |
| **AtScript: house.ats** | **Closure Annotated: house.js** |
| import {Foo} from 'bar';  namespace('house');  export class Lock {  }  export class Door {  var lock:Lock;  constructor(lock:Lock) {  this.lock = lock;  }  } | goog.require('bar.Foo');  goog.provide('house.Lock');  goog.provide('house.Door');  baz.Lock = function() {};  /\*\*  \* @param {house.Lock} lock  \*/  baz.Door = function(lock) {  /\*\* @type {house.Lock} \*/  this.lock = lock;  } |

# Caveats

You may need to describe what you did not do or why simpler approaches don't work. Mention other things to watch out for (if any).

# Security Considerations

How you’ll be secure

# Performance Considerations / Test Strategy

How you’ll be fast.

# Work Breakdown

Description of development phases and approximate time estimates.