

# USING TYPESCRIPT

# ABOUT ME

- FORTRAN, Pascal, C/C++, C#, JS, Typescript
- Engineer, currently at Pearson as UX developer
- Been working with typescript since November 2012
- Projects I currently work on have ~40k lines of ts

# INTRO

- Dynamic languages with optional static typing: Dart, Typescript,...
- A -somewhat- new idea...
- Where did this idea come from?

## On the Revival of Dynamic Languages

Oscar Nierstrasz, Alexandre Bergel, Marcus Denker, Stéphane Ducasse,  
Markus Gälli, and Roel Wuyts

Software Composition Group, University of Bern  
[www.iam.unibe.ch/~scg](http://www.iam.unibe.ch/~scg)

<https://sites.google.com/site/steveyegge2/is-weak-typing-strong-enough>  
<http://herbsutter.com/2008/06/20/type-inference-vs-staticdynamic-typing>

Intended for submission to the *Revival of Dynamic Languages*

## Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages

Erik Meijer and Peter Drayton  
Microsoft Corporation

## RPython A Step Towards Reconciling Dynamically and Statically Typed OO Languages

Antonio Cuni – DISI, Università degli Studi di Genova  
joint work

### Static Type Inference for Ruby

Michael Furr   Jong-hoon (David) An   Jeffrey S. Foster   Michael Hicks

Department of Computer Science  
University of Maryland  
College Park, MD 20742  
(furr,davidan,jfoster,mh)@cs.umd.edu



» PEP Index > PEP 3107 -- Function Annotations

## Diamondback Ruby



### Overview

Diamondback Ruby (DRuby) is an extension to Ruby that aims to bring the benefits of static typing to Ruby without compromising the expressiveness of the language.

[Overview](#)

[Download](#)

[Documentation](#)

Title of thesis:     Dynamic Inference of Static Types for Ruby  
  
Jong-hoon (David) An, Master of Science, 2010  
  
Thesis directed by:   Professor Jeffrey S. Foster  
                                 Department of Computer Science

[Home](#) [Blog](#) [Examples](#) [Tutorial](#) [Code](#) [Wiki](#) [FAQ](#) [Roadmap](#) [About](#) [Contact](#)

## mypy

The mypy programming language is an experimental Python variant that aims to combine the benefits of dynamic (or "duck") typing and static typing. Our goal is to have the expressive power and convenience of Python combined with compile-time type checking. The long-term goal is to also support efficient

All Things Python  
Adding Optional Static Typing to Python  
by Guido van Rossum  
December 23, 2004

## C# Dynamic

The **dynamic** keyword influences compilation.

A dynamic variable,

parameter

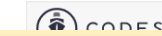
or field can have any type. Its type can change during runtime.

The downside is that performance suffers and you lose compile-time checking.

dynamic



Codeship runs  
and continuous  
your code



# CONTENT OF THIS TALK

- Background
- Anders Hejlsberg
- What typescript gives you: ES6 stuff, typing, tooling, documentation of intend
- Where to get definition files => definitely typed
- Demo

# ANDERS HEJLSBERG

- Language designer (his own words)
- Turbo Pascal, Borland
- Delphi
- C#
- “Can’t explain it in 5min -> not a good idea”, maybe
- “Typescript filters out the semantic subset of java script that makes sense”



# WHAT YOU GET ...

- EcmaScript6 stuff: module, class, extends, arrow function
  - [https://github.com/lukehoban/es6features?utm\\_source=javascriptweekly&utm\\_medium=email#arrows](https://github.com/lukehoban/es6features?utm_source=javascriptweekly&utm_medium=email#arrows)
  - [https://wiki.mozilla.org/ES6\\_plans](https://wiki.mozilla.org/ES6_plans)
  - => handling of this pointer (lexically scoped)
  - ~ like coffee script, but not a new language
- “Optional” static typing, interface, implements
  - interfaces open ended, multiple files can contribute
  - annotations
- Static typing: not that there is anything wrong with that
  - inference, flowing the types via generics
  - inverse inference
  - lib.d.ts, the JS runtime lib and DOM
  - JS has type information but only at run time
  - typescript, not provably type safe
  - [http://en.wikipedia.org/wiki/type\\_system](http://en.wikipedia.org/wiki/type_system)

# ...MORE

- Statement completion
  - Sublime Text and many others, when working with large libraries helpful
- Documentation of intend
  - example: rest service format described via interface
  - write a test for what you actually get from the service against the interface
  - if you have to go back and talk with the backend people about their data, you have a piece of code as proof
- No TS “engine”/VM, no runtime library
  - impact only at design time
- Superset of JS
  - for people who know java script a shallow learning curve
- Debugging in Chrome, Firefox, Visual Studio, IntelliJ, Webstorm...
  - directly in TS files via source maps
- Upgrading to a newer version of a library?
  - upgrade the .d.ts file and fix the errors
- Handling of java script module systems AMD/CommonJS
  - via compiler flag



# NODETOOLS FOR VISUAL STUDIO

- Install
- NPM manager
- Debugging
  - Break on exception
  - Immediate window context in broken context of app
  - Remote debugging (also on Linux) RemoteDebug.js
- Profiling

# DEMO

- Simple examples
  - Sanity
- ES6 features: module, class, arrow
- 
- More examples
  - inference, reverse inference, refactoring
  - changing an interface from a staging to a production scenario, rest calls, js module systems
- Code re-use
  - name spaces/modules simplify code reuse
  - more utility functions used in many places, changing these becomes very simple
- Unit tests
  - saves unit tests.
  - Convert d3.js GIST , debugging in chrome with source map

# SUMMARY

- Transpiler -> JS
- Optional static typing
- ES 6 features
- IDEs and debuggers
- Saves unit tests and lets you sleep better

QUESTIONS ?

- Thank you!