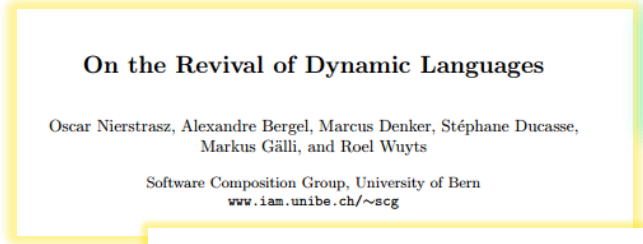
USING TYPESCRIPT

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

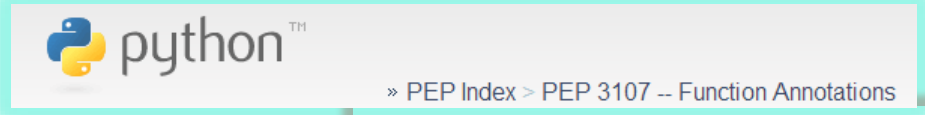
INTRO

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

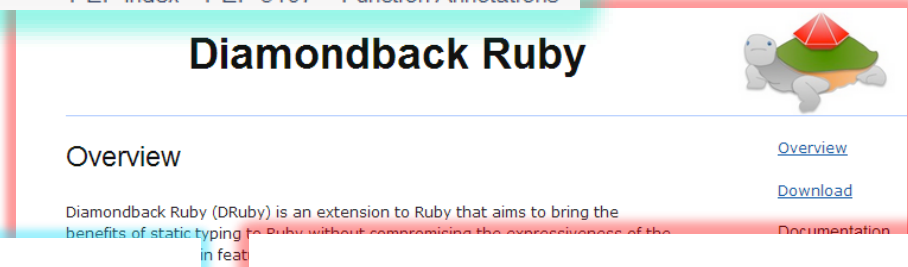

On the Revival of Dynamic Languages

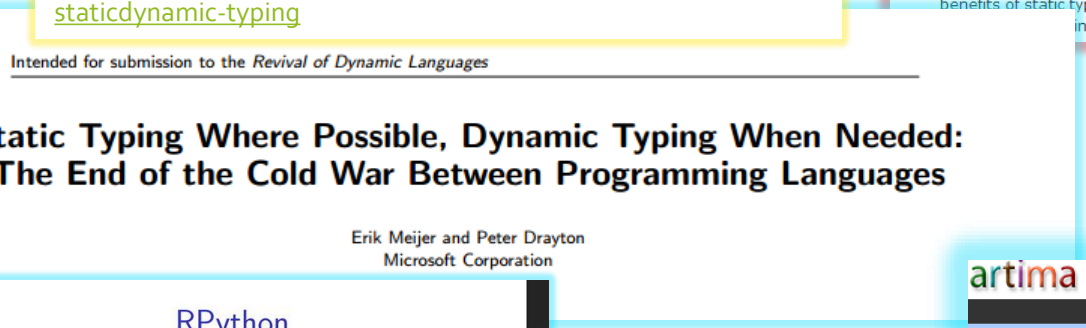
Oscar Nierstrasz, Alexandre Bergel, Marcus Denker, Stéphane Ducasse, Markus Gälli, and Roel Wuyts

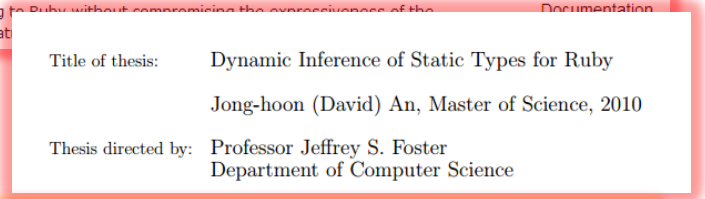
Software Composition Group, University of Bern
www.iam.unibe.ch/~scg

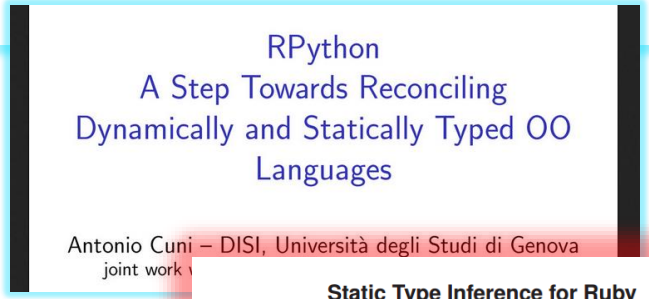


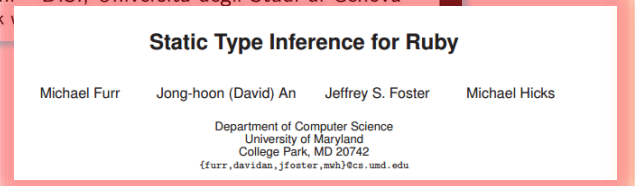
https:///sites.google.com/site/steveyegge2/is-weak-typing-strong-enough
http://herbsutter.com/2008/06/20/type-inference-vs-staticdynamic-typing

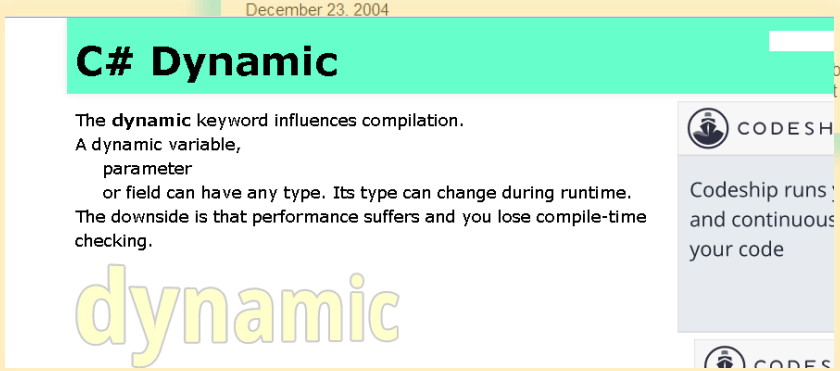












CONTENT OF THIS TALK

- 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://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

…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

NODE TOOLS 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!