Towards a Corpus Study of the Dynamic Type



Dibri Nsofor [University of Utah] [Engine > Systems > Programmability]



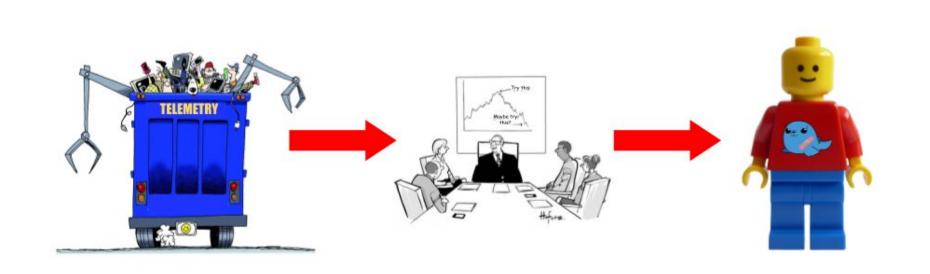
Managers: Aaron Weiss, Andy Freisen, Mitesh Shah

Research Question and Gradual Typing

Types make code safer and easier to maintain.

Retrofitting a type system to a dynamically typed language exposes interesting design challenges. How can language designers decide the types these languages deserve?





Instrument telemetry. Allow language designers react to user patterns. We can address misuse (provide better educational content), improve linting suggestions and documentation towards growing a gradual type system.

we collect:

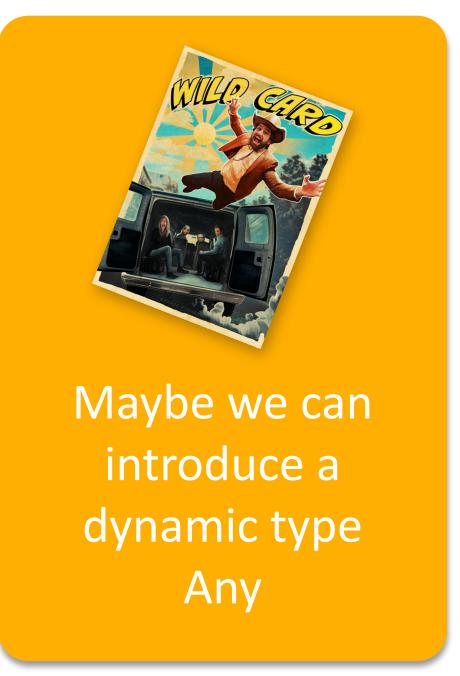
- Snippets of code
- Active typing modes
- Type signatures
- Patterns of interest



We collate the data once a week for a manual analysis. We have examined 542,850 records across 2124 experiences that adopt the any type.

What should this return? An Adhoc Union, a string, a custom object?

```
1 local function div(n: number, d: number)
        return "div0"
    else
        return n/d
```



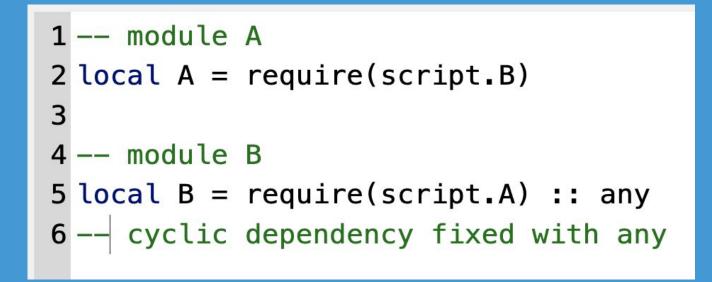
Progress

We look through for repeat patterns that stand out. Here are some interesting ones:

Anys to model primitive top types

Suggestion: Include support for outer shape checks.

```
3 type Callback = (...any) -> (...any) -- bad
5 function higherOrder(count: number, func: Callback, ...)
     if count > 0 then
         func(...)
```



Circular dependencies resolved with Anys

Suggestion: Maintain own type modules to avoid cycles.

Interactions with the Data Model

Suggestion: IDE support for jumps to Creator Docs and shared type libraries.

3 local handle = Instance.new("Handle")

Suggestions

- New diagnostics for Luau: Warnings
 - Raise warnings instead of errors for conditions that will not affect runtime behaviour. E.g. Luau can raise warnings for cyclic type dependencies to prevent the casts to any.
- Shared Self Type
- Bounded Generics
- Allow users to constrain bounds for generic types. This could also be an interesting approach to implementing a shared self type solution.
- Shared typing libraries.
- The Roblox types are not precise enough. We should open source this typing effort and leverage community involvement to enhance precision.
- Address User Misconceptions
- Common misuse of the type system stems from a knowledge gap. We have assembled a typing guide for Roblox creators to address common anti-patterns. Preview: go/typing-guide
- Better IDE support to jump to Data Model documentation.
- Magic Functions and Instance declarations should offer support to jump straight to necessary documentation.

Future Work

- Collect more data, test hypotheses.
- Study user misconceptions, provide lints and design educational content for creators.
- Explore Non Strict Mode design challenge.
- An approachable yet trustworthy type checker.

