Towards Agda on the Web

Using GHC WASM backend, Language Server Protocol, and VS Code for the Web

Liang-Ting Chen • AIM XXX VII (20 Nov 2023) • Institute of Information Science, Academia Sinica

Problems

Installing Agda is Unreasonably Complicated

#5725, #6866, #ux:installation

- Our TA includes programmers, *mathematicians, computer scientists*, CS/math students, and more.
- Familiarity with Haskell toolchain is implicitly assumed.
- Compiling AGDA with `text-icu` is tricky because of `icu4c`.
- More information does not help (#6866).
- LEAN 4's quick installation has only 3 steps
    - Get VS CODE, Open VS CODE, Get LEAN 4 extension

Maintaining Binary Distributions for Different Platforms?

#5202

- License (maybe a non-issue?)
- Installing from prepackaged Agda is fine …
- except for Windows users.
    - Maintaining a GitHub workflow to compile the Windows bindist is costly.
    - Packaging AGDA using Windows installer requires expertise.
- Still requires many steps to set up AGDA and AGDA mode.
    - Where is `GHCup`-equivalent?

Updating Documentation is Costly and not Fun

#6866, again

- More information is sometime less useful.
- Relying on pull requests is not sustainable (less organised).
- `hello world` example confuses many people.
- Lean 4 has just enough information for casual users
  - https://lean-lang.org/lean4/doc/quickstart.html

Agda Mode for Emacs isn’t Eternal

#5917, #6953, #6983

- Agda mode was implemented by Makoto Takeyama (not active) and is maintained by Nils Anders Danielsson.
  - The user base of Emacs is small.
  - Very few Elisp-fluent developers
- Philip Kaludercic (@phikal) has recently contributed a lot of PRs, but many of them are stalled.
- Agda mode is broken with Emacs >28 and
  - WSL only provides Emacs >28 (discussion)

Agda Mode for VS Code is not Maintainable

https://github.com/banacorn/agda-mode-vscode

- Implemented originally by Ting-Gian Lua (@banacorn, only active for AIM) and
- Currently maintained by Zong-You Shih and me.
- Agda mode for VS Code is written in ReScript (!).
    - The user base of ReScript is tiny.

Summary

- Installing Agda is more complicated than the typical practice.
- Maintaining binary distributions is subtle (especially for Windows).
- More information does not help.
- IMO, Agda modes for Emacs and VS Code seem eroding.

Vision

Agda Web

Essential criteria

- Agda should be usable in any modern browser.
    - Cross-platform, incl. Linux, macOS, Windows, Android, and iOS
- Installation should still be possible and easy.
- Any library from a Git repository should be usable without manual download.
- (License notice is available on Hackage.)

Agda Web

Desirable criteria

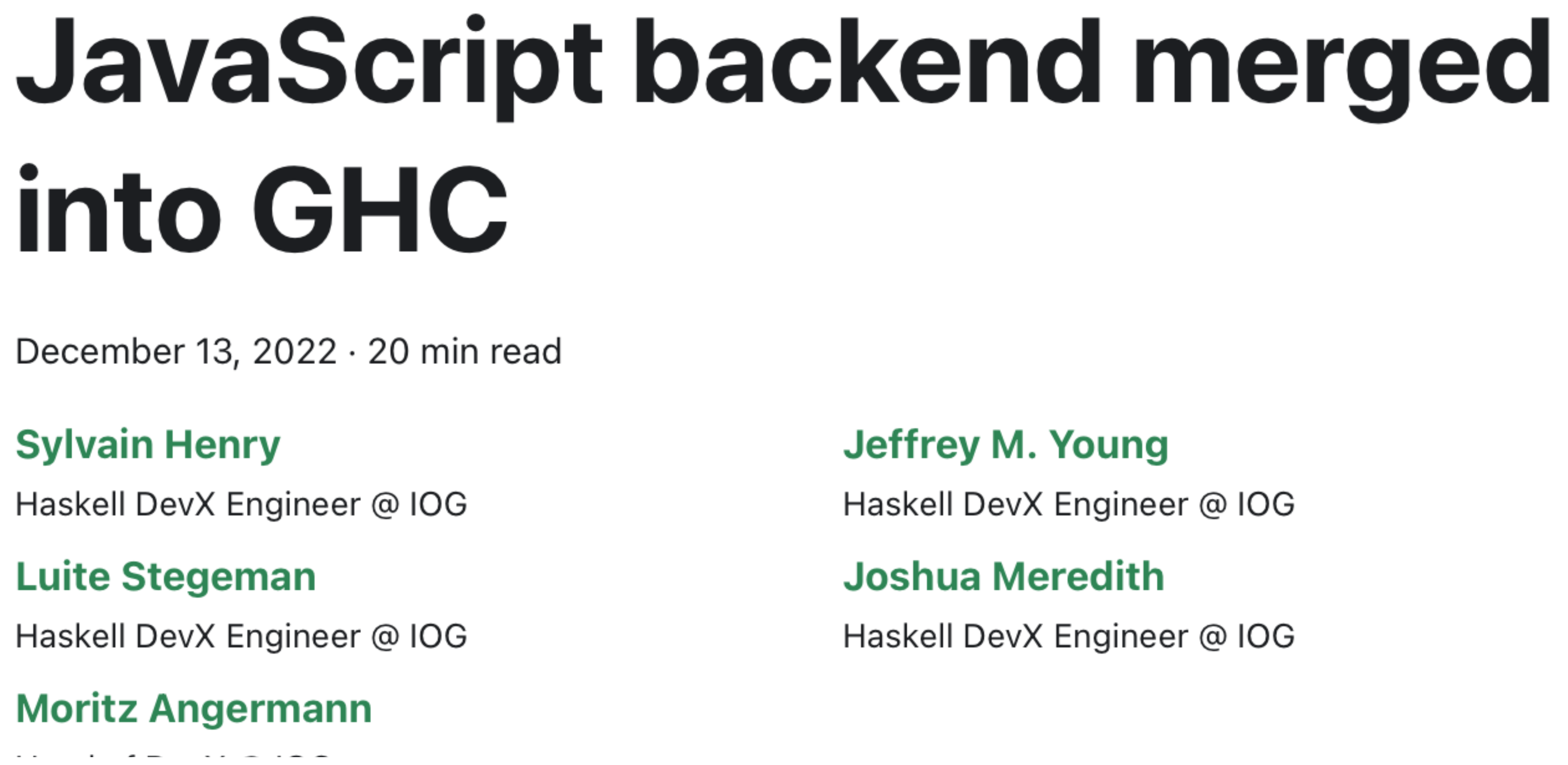
- All features supported by the Agda mode for Emacs should also be supported.
- Only one language should be used for dev.
- No additional repo to maintain— still `agda/agda` instead of `agda/web-agda`.
- Can be used for interactive textbook
    - e.g., https://lovettsoftware.com/NaturalNumbers/

But, how?

Technologies

GHC JavaScript and WASM Backends

Thanks to Tweag and IOG



JavaScript backend merged into GHC

December 13, 2022 · 20 min read

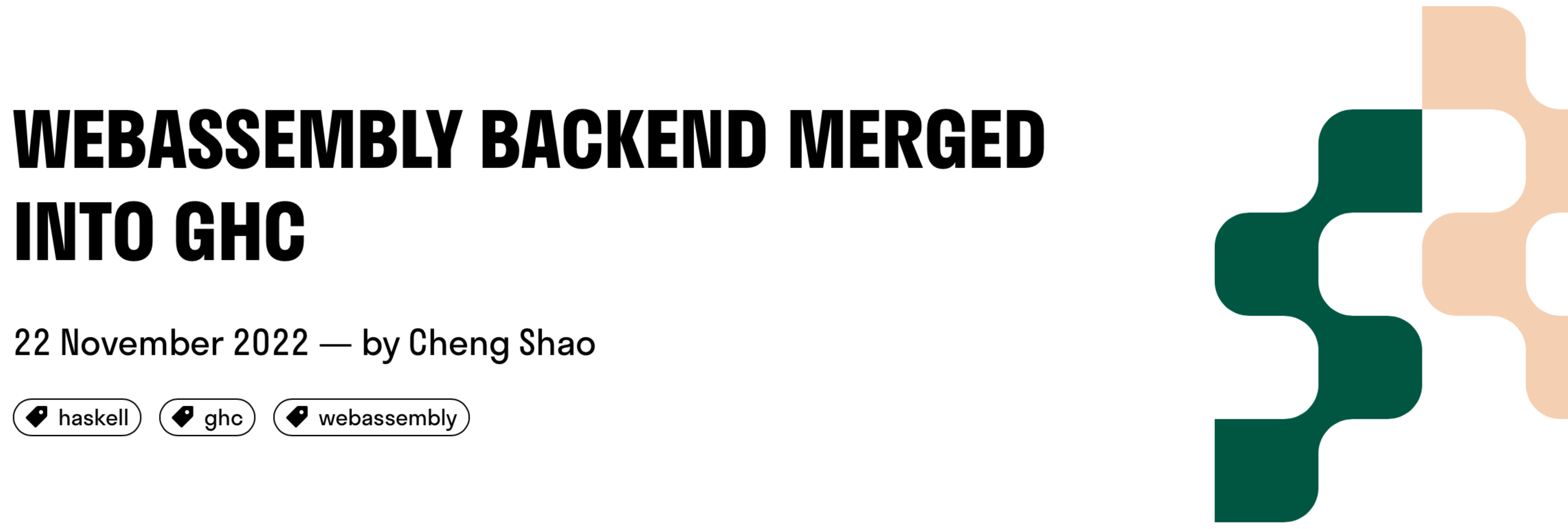
Sylvain Henry
Haskell DevX Engineer @ IOG

Jeffrey M. Young
Haskell DevX Engineer @ IOG

Luite Stegeman
Haskell DevX Engineer @ IOG

Joshua Meredith
Haskell DevX Engineer @ IOG

Moritz Angermann
[illegible]



WEBASSEMBLY BACKEND MERGED INTO GHC

22 November 2022 — by Cheng Shao

haskell ghc webassembly

GHC JavaScript and WASM Backends

Haskell for the Web

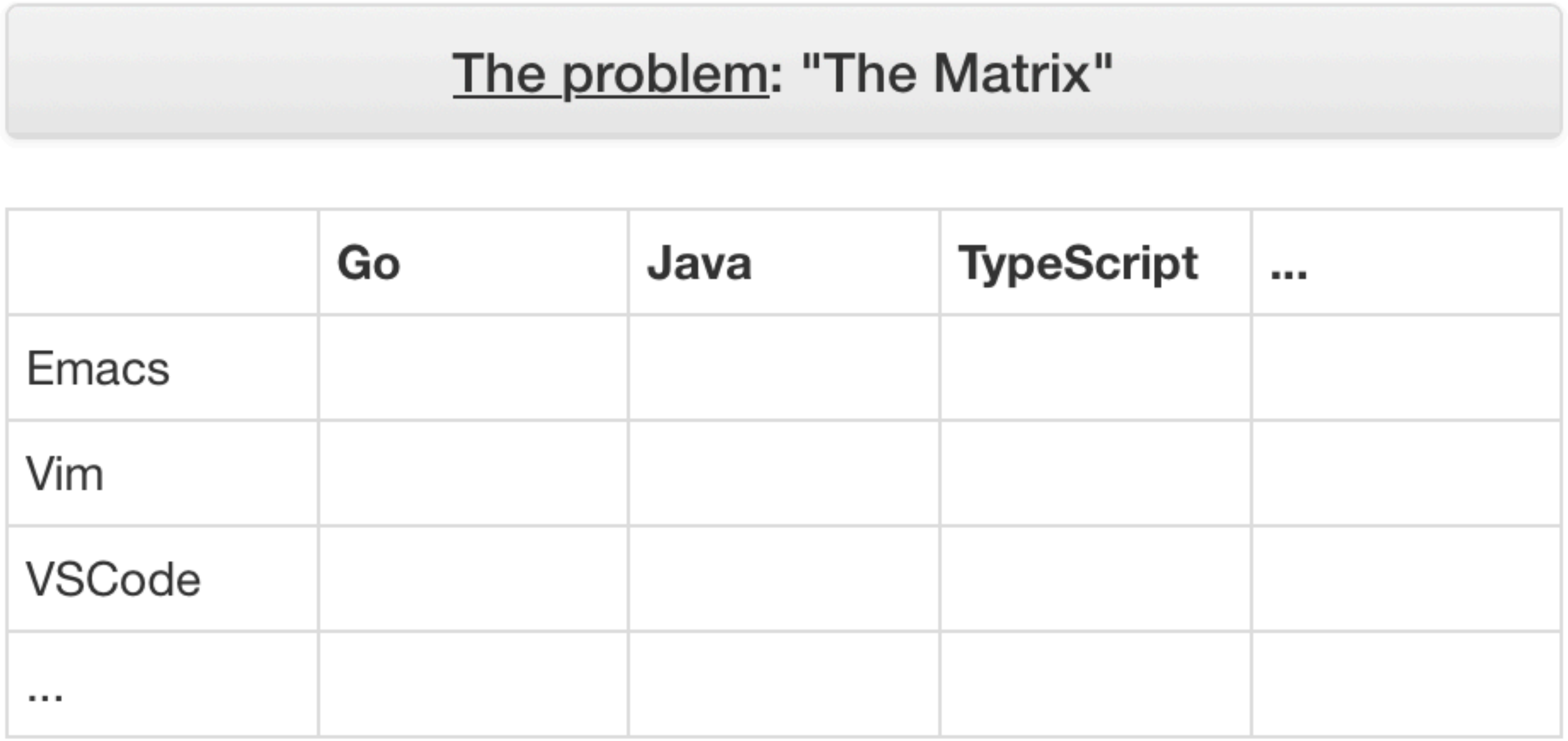
- VS Code extensions are JavaScript programs
  - … can be implemented in Haskell using (new) GHC JS backend.
- LSP is a Haskell package for implementing LSP server
  - … and be compiled into WASM to run in a browser.
- In theory, Agda ecosystem can all be written in Haskell and compiled to JS/ WASM.

Language Server Protocol

One ‘Server’ to Rule Them (editors) All

The Matrix Problem

Each editor implements their own language support

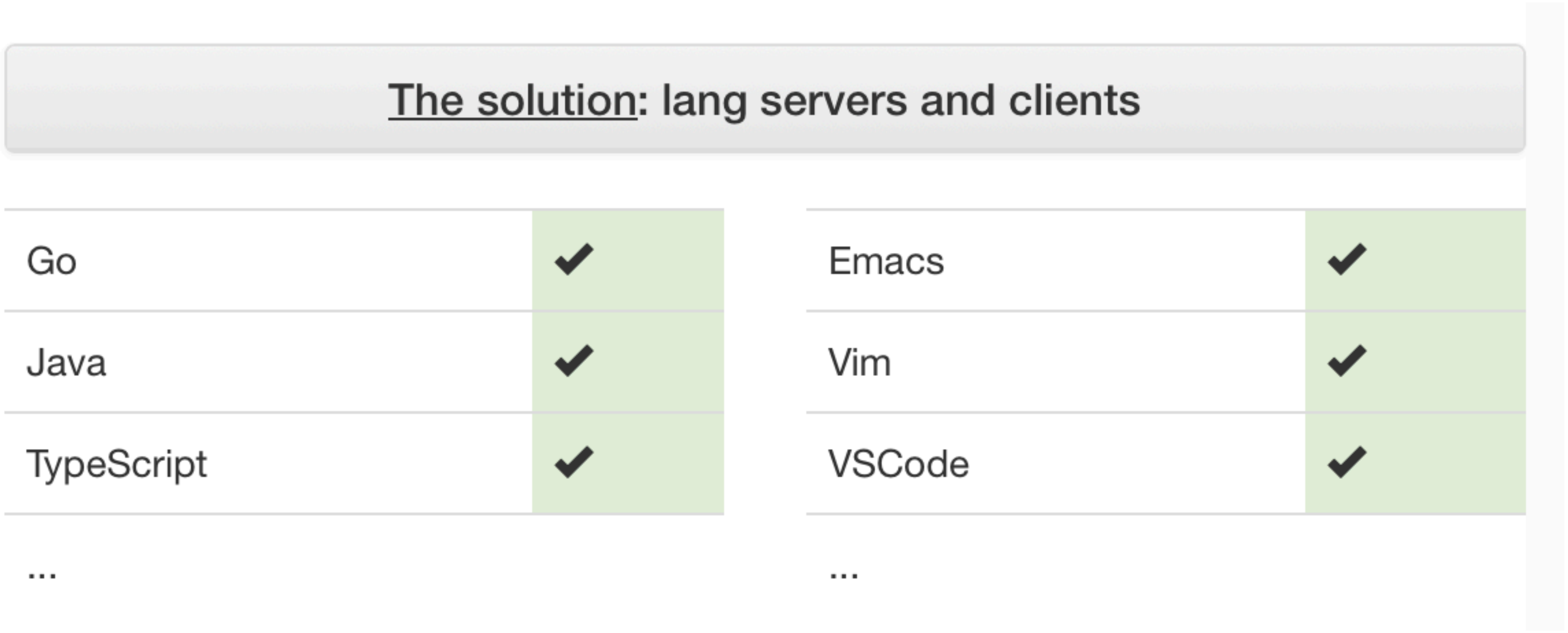


The problem: "The Matrix"

	Go	Java	TypeScript	...
Emacs				
Vim				
VSCode				
...				

The LSP Solution

Each editor (resp. language) needs only one client (resp. server).



The solution: lang servers and clients

Go	✓
Java	✓
TypeScript	✓
...	

Emacs	✓
Vim	✓
VSCode	✓
...	

Language Server Protocol

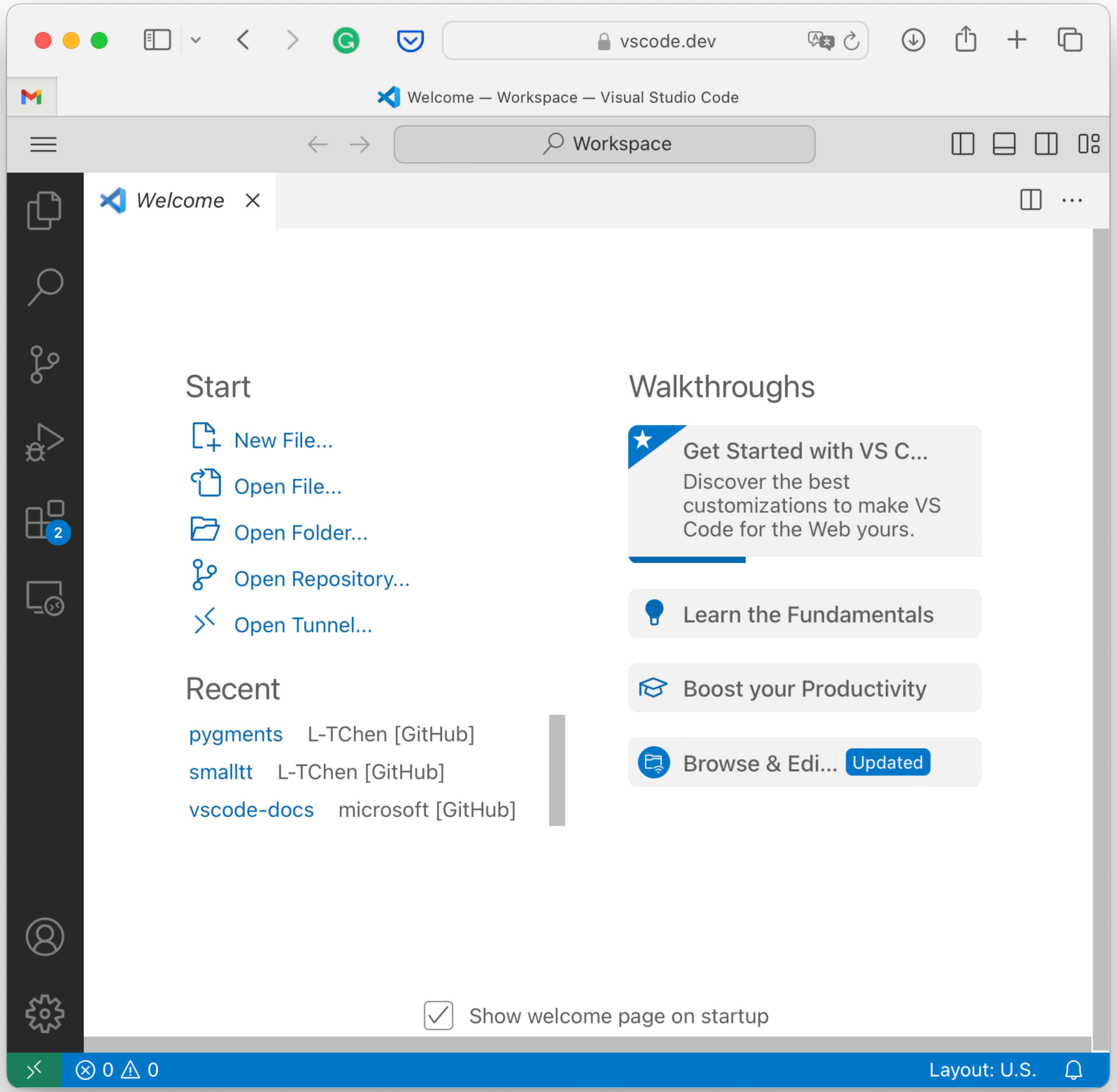
A protocol for common IDE features

1. highlighting
2. formatting
3. code action (interactive programming)
4. type-checking
5. Goto definition, finding references, renaming identifiers, etc.

- Language clients are implemented in editors incl. Emacs, Vim, VS Code, etc.

Visual Studio Code for the Web

https://vscode.dev/


vscode.dev

Welcome — Workspace — Visual Studio Code

Workspace

Welcome

Start

New File...
Open File...
Open Folder...
Open Repository...
Open Tunnel...

Recent

pygments	L-TChen [GitHub]
smalltt	L-TChen [GitHub]
vscode-docs	microsoft [GitHub]

Walkthroughs

Get Started with VS C...
Discover the best customizations to make VS Code for the Web yours.

Learn the Fundamentals

Boost your Productivity

Browse & Edi... Updated

Show welcome page on startup

0 0 Layout: U.S.

Visual Studio Code for the Web

https://vscode.dev/

- Requirements
  - Any modern browser
  - (Optional) Support for File System Access API

- Features
  - LSP
  - Extensions
  - Remote GitHub repository
  - Local folders on supported browsers)
  - Even on mobile devices!

Visual Studio Code for the Web

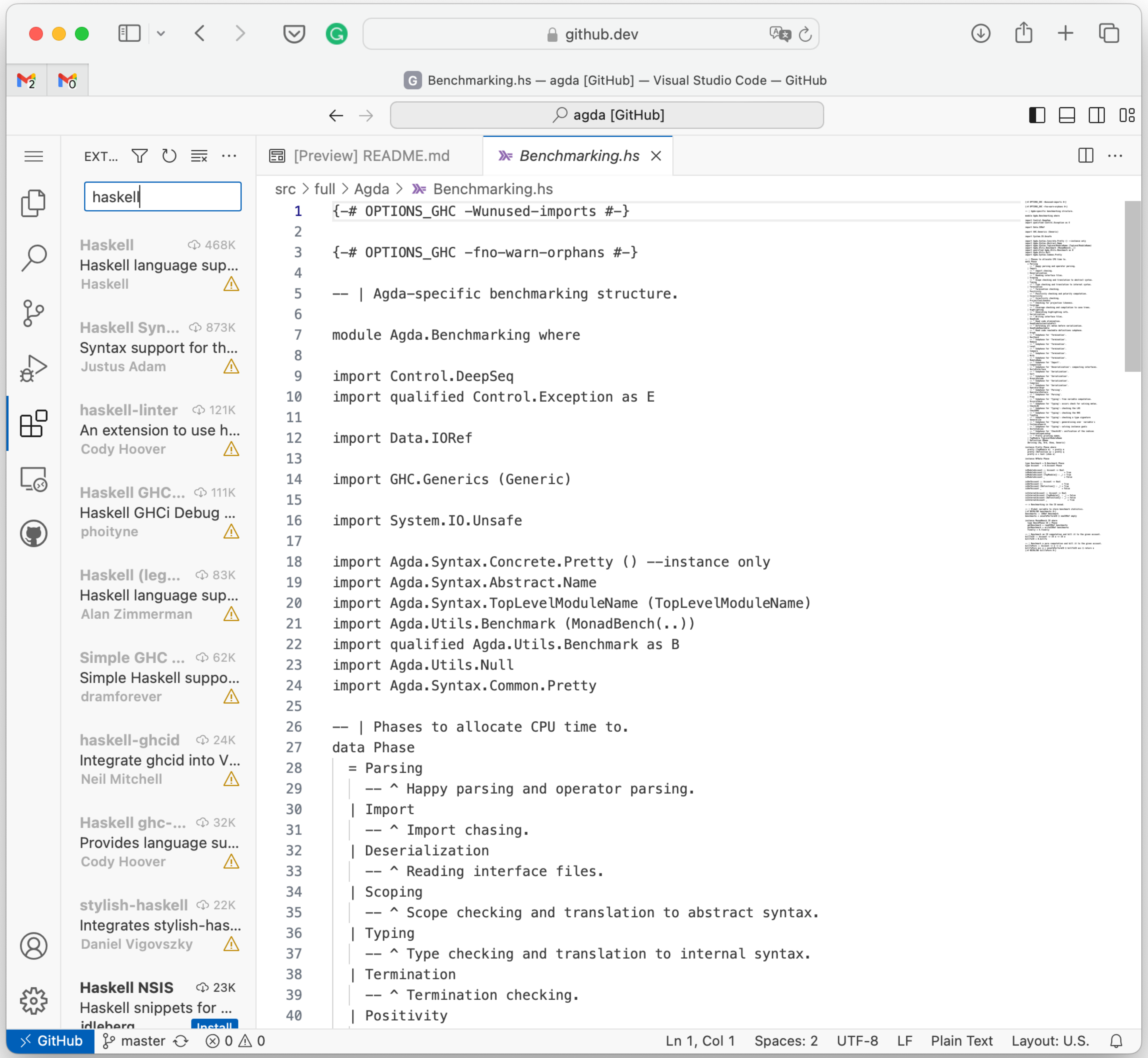
https://vscode.dev/

- VS Code for the Web can be ‘installed’ to run locally.
- Supported by
  - Safari on macOS/iOS, or
  - Chrome on Windows/macOS/Linux
- JavaScript/WASM is the new Java (with WebAssembly System Interface).

Github.dev

A variant of VS Code Web

- Press `.` in any GitHub repository
- https://docs.github.com/en/codespaces/the-githubdev-web-based-editor



```haskell
{-# OPTIONS_GHC -Wunused-imports #-}

{-# OPTIONS_GHC -fno-warn-orphans #-}

-- | Agda-specific benchmarking structure.

module Agda.Benchmarking where

import Control.DeepSeq
import qualified Control.Exception as E

import Data.IORef

import GHC.Generics (Generic)

import System.IO.Unsafe

import Agda.Syntax.Concrete.Pretty () --instance only
import Agda.Syntax.Abstract.Name
import Agda.Syntax.TopLevelModuleName (TopLevelModuleName)
import Agda.Utils.Benchmark (MonadBench(..))
import qualified Agda.Utils.Benchmark as B
import Agda.Utils.Null
import Agda.Syntax.Common.Pretty

-- | Phases to allocate CPU time to.
data Phase
  = Parsing
    -- ^ Happy parsing and operator parsing.
  | Import
    -- ^ Import chasing.
  | Deserialization
    -- ^ Reading interface files.
  | Scoping
    -- ^ Scope checking and translation to abstract syntax.
  | Typing
    -- ^ Type checking and translation to internal syntax.
  | Termination
    -- ^ Termination checking.
  | Positivity
```

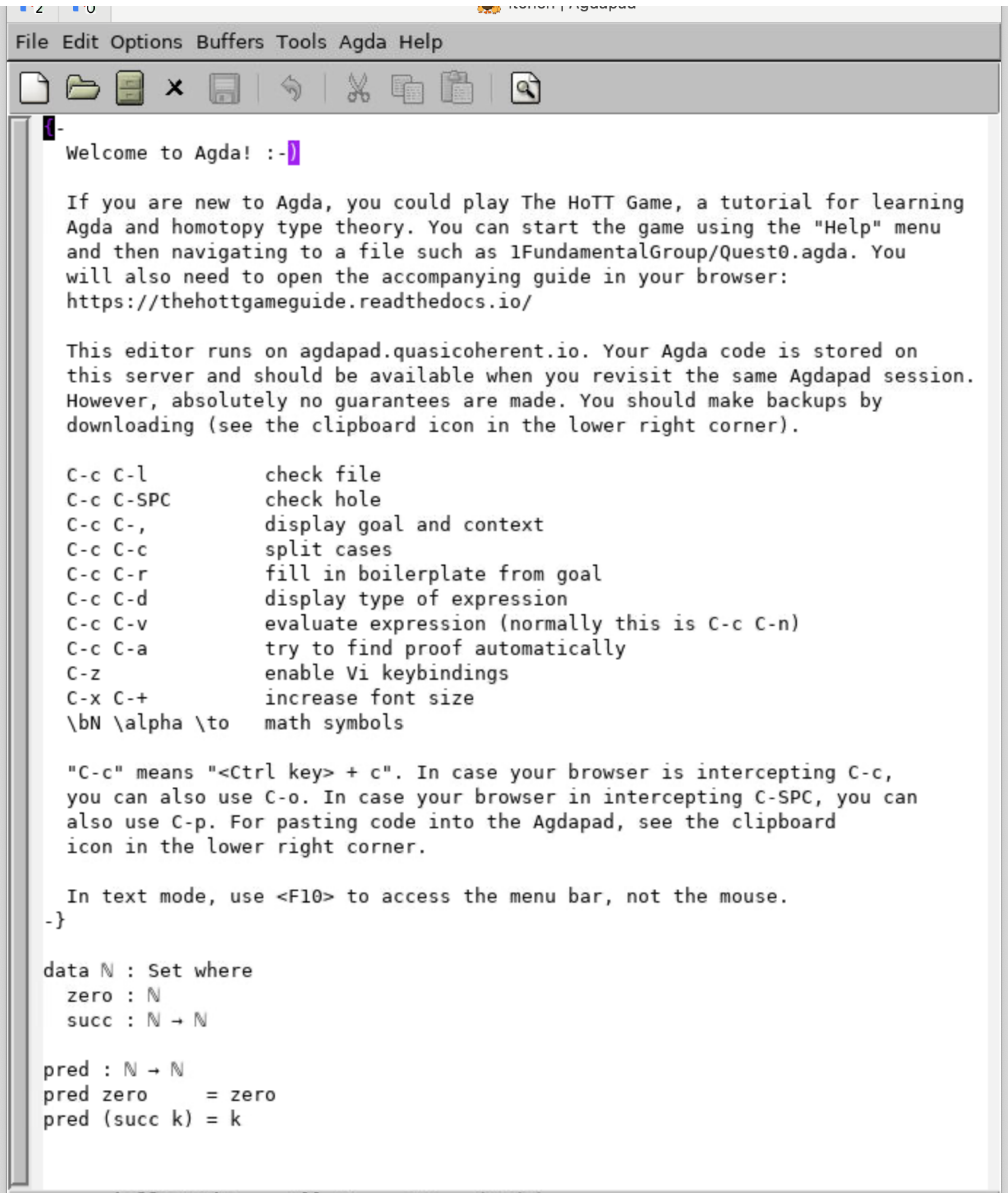
CodeMirror

CodeMirror

Case Studies

Agda Pad

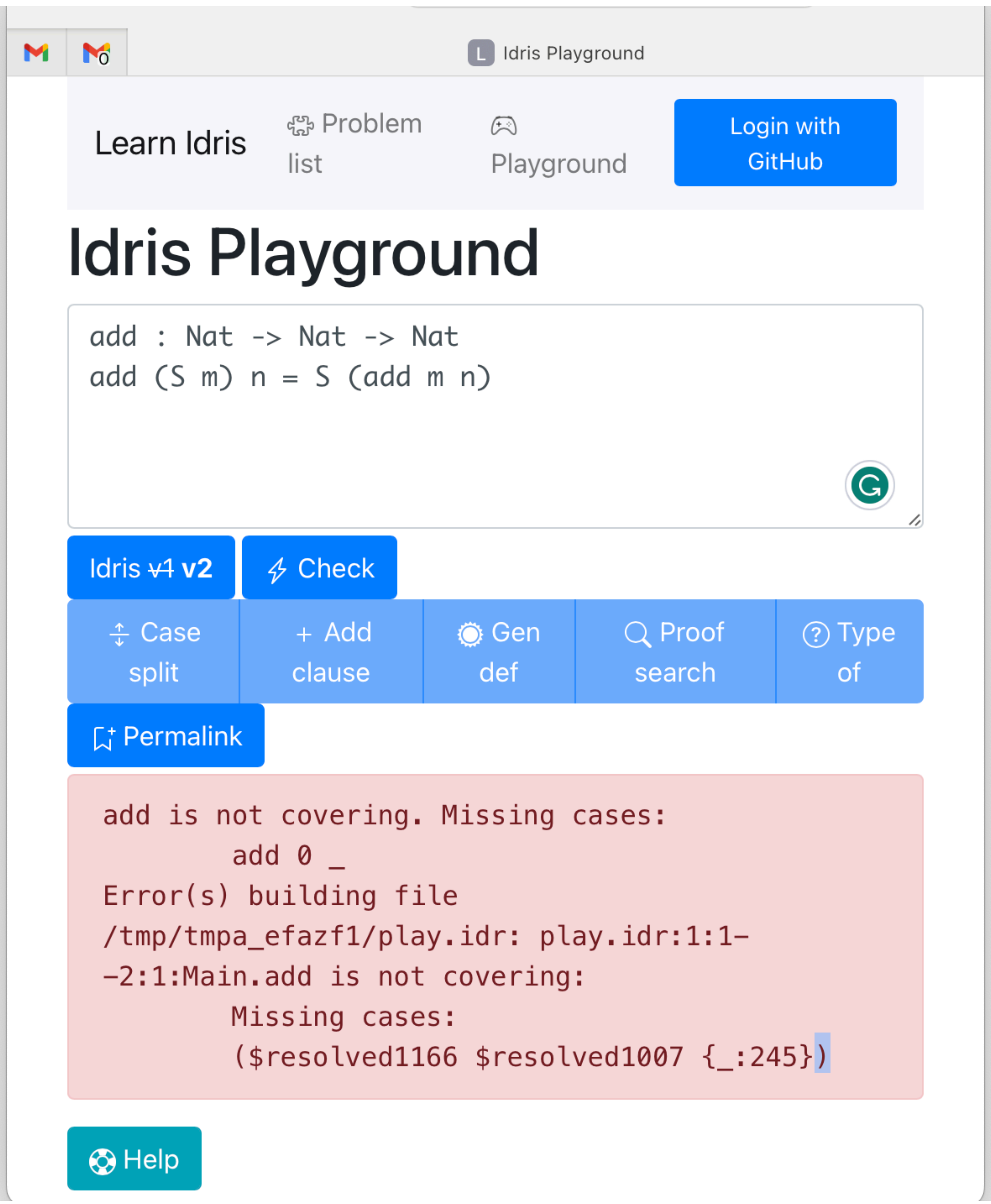
- Agda Pad is a VNC server running on an Emacs instance.
- https://agdapad.quasicoherent.io/



```
{-
  Welcome to Agda! :-)

  If you are new to Agda, you could play The HoTT Game, a tutorial for learning
  Agda and homotopy type theory. You can start the game using the "Help" menu
  and then navigating to a file such as 1FundamentalGroup/Quest0.agda. You
  will also need to open the accompanying guide in your browser:
  https://thehottgameguide.readthedocs.io/

  This editor runs on agdapad.quasicoherent.io. Your Agda code is stored on
  this server and should be available when you revisit the same Agdapad session.
  However, absolutely no guarantees are made. You should make backups by
  downloading (see the clipboard icon in the lower right corner).

  C-c C-l          check file
  C-c C-SPC        check hole
  C-c C-,          display goal and context
  C-c C-c          split cases
  C-c C-r          fill in boilerplate from goal
  C-c C-d          display type of expression
  C-c C-v          evaluate expression (normally this is C-c C-n)
  C-c C-a          try to find proof automatically
  C-z              enable Vi keybindings
  C-x C-+          increase font size
  \bN \alpha \to   math symbols

  "C-c" means "<Ctrl key> + c". In case your browser is intercepting C-c,
  you can also use C-o. In case your browser in intercepting C-SPC, you can
  also use C-p. For pasting code into the Agdapad, see the clipboard
  icon in the lower right corner.

  In text mode, use <F10> to access the menu bar, not the mouse.
-}

data ℕ : Set where
  zero : ℕ
  succ : ℕ → ℕ

pred : ℕ → ℕ
pred zero     = zero
pred (succ k) = k
```

Idris Playground

- Idris runs on a server (Google Cloud Run)
- Costly—Servers are billed when someone is using the website.
- The interface is nicer.
- https://learn-idris.net/play

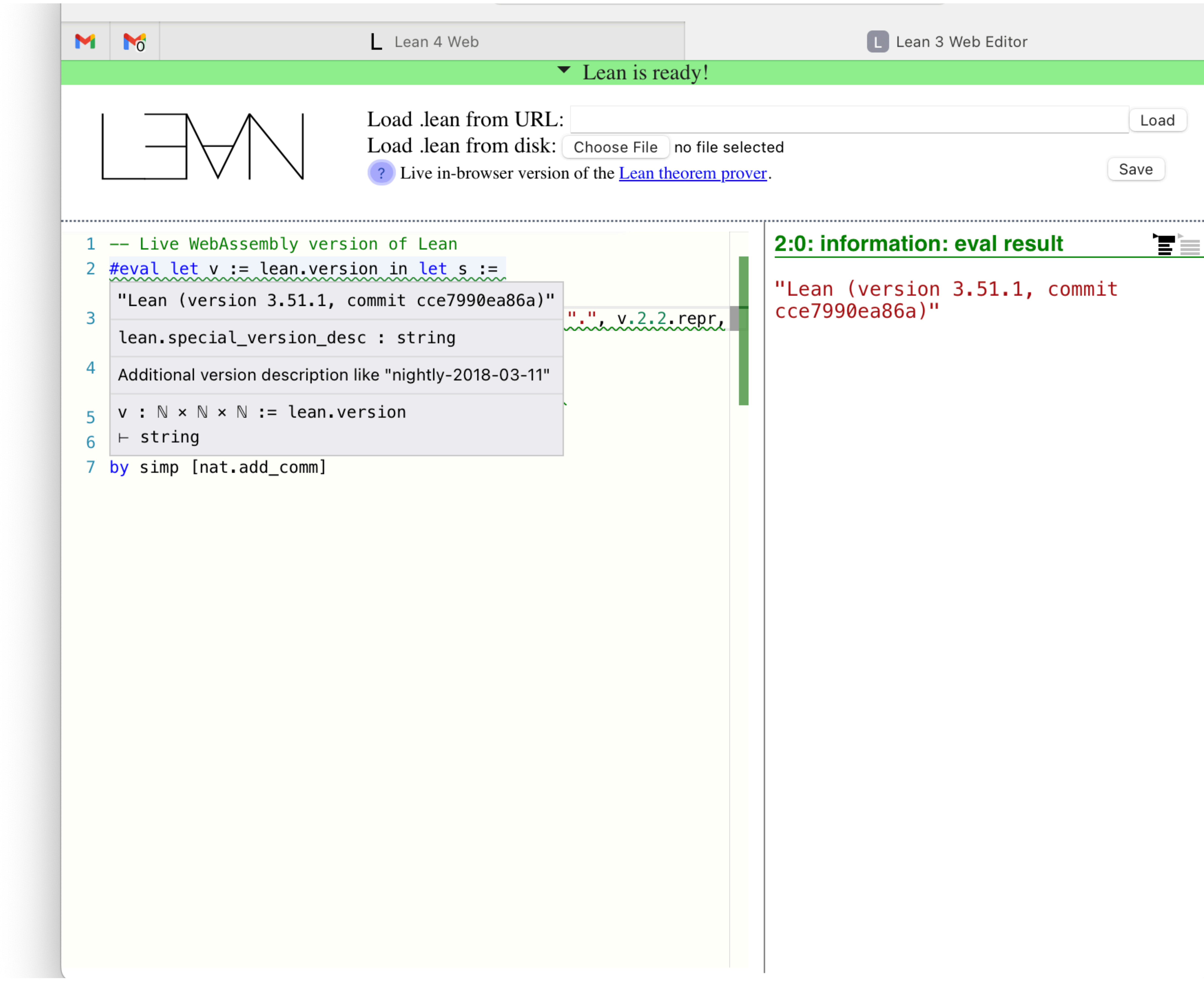


```
add : Nat -> Nat -> Nat
add (S m) n = S (add m n)
```

```
add is not covering. Missing cases:
        add 0 _
Error(s) building file
/tmp/tmpa_efazf1/play.idr: play.idr:1:1-
-2:1:Main.add is not covering:
        Missing cases:
        ($resolved1166 $resolved1007 {_:245})
```

Lean 3 Web

- https://github.com/leanprover-community/lean-web-editor
- Lean 3 compiled to WASM and runs in the browser
- Language server is supported.
- Built by Emscripten.



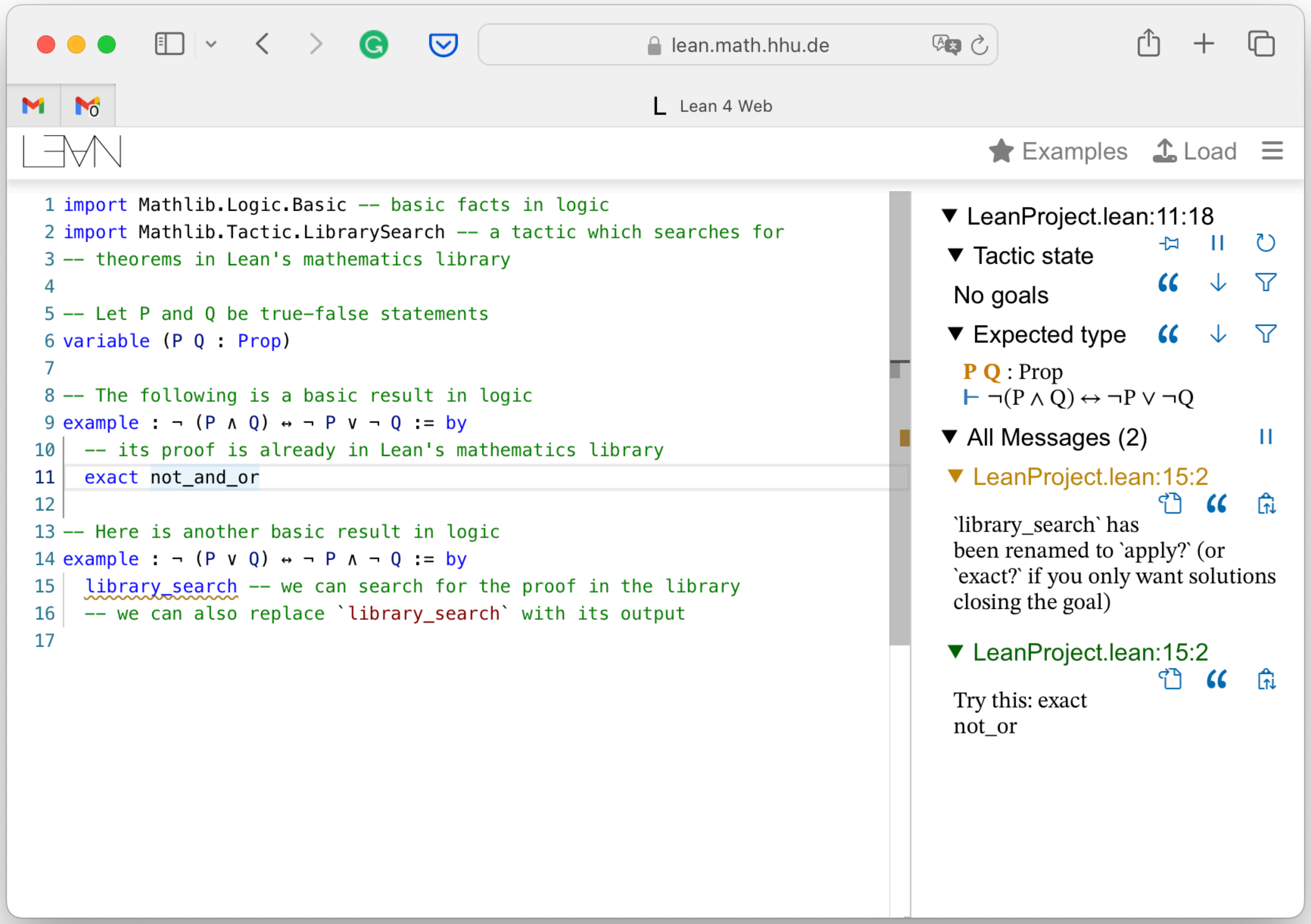
```
-- Live WebAssembly version of Lean
#eval let v := lean.version in let s :=
"Lean (version 3.51.1, commit cce7990ea86a)"           ".", v.2.2.repr,
lean.special_version_desc : string
Additional version description like "nightly-2018-03-11"
v : ℕ × ℕ × ℕ := lean.version
⊢ string
by simp [nat.add_comm]
```

2:0: information: eval result

```
"Lean (version 3.51.1, commit
cce7990ea86a)"
```

Lean 4 Web

- https://github.com/leanprover-community/lean4web
- Runs on a web server, not in the browser



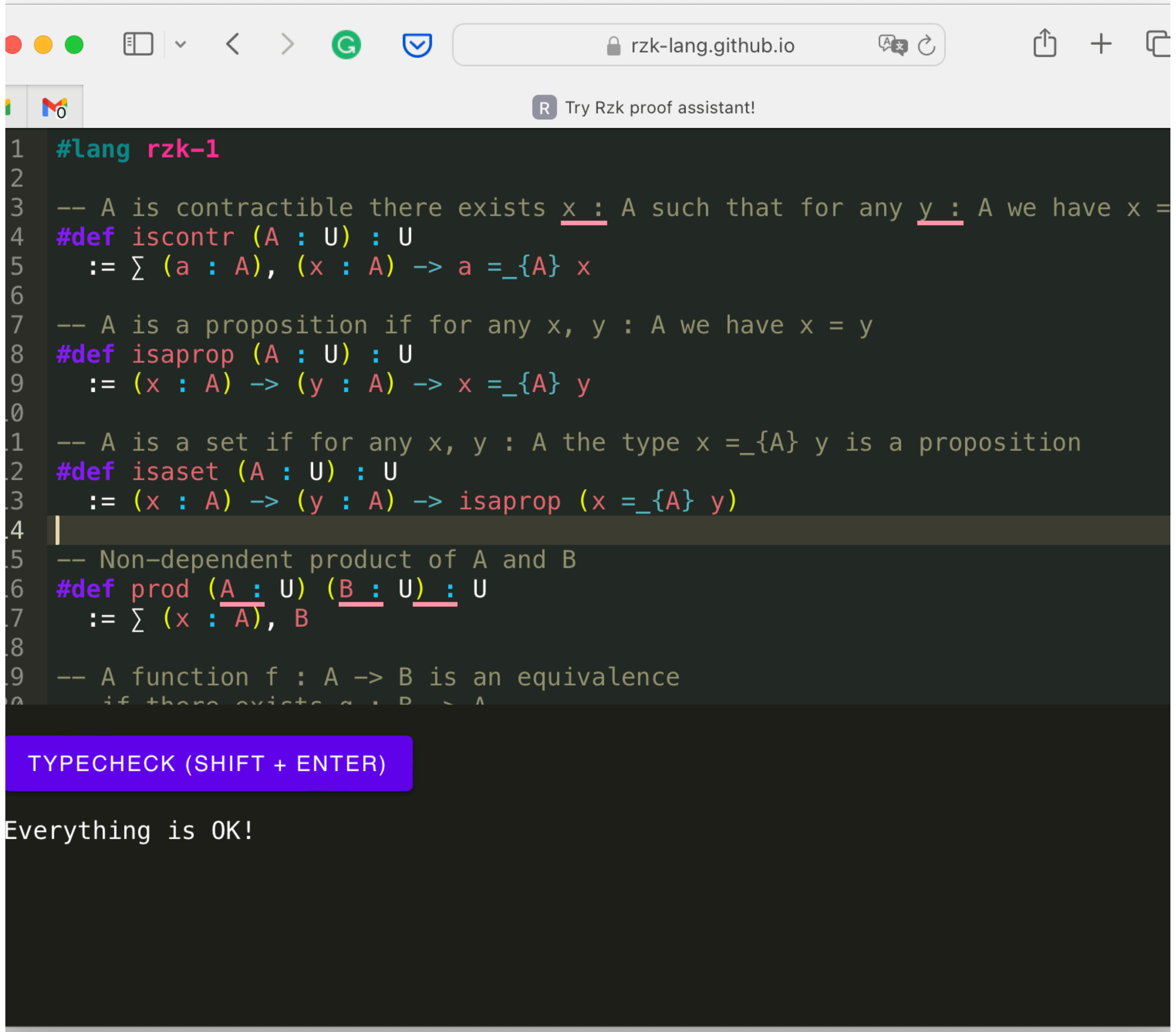
```
import Mathlib.Logic.Basic -- basic facts in logic
import Mathlib.Tactic.LibrarySearch -- a tactic which searches for
-- theorems in Lean's mathematics library

-- Let P and Q be true-false statements
variable (P Q : Prop)

-- The following is a basic result in logic
example : ¬ (P ∧ Q) ↔ ¬ P ∨ ¬ Q := by
  -- its proof is already in Lean's mathematics library
  exact not_and_or

-- Here is another basic result in logic
example : ¬ (P ∨ Q) ↔ ¬ P ∧ ¬ Q := by
  library_search -- we can search for the proof in the library
  -- we can also replace `library_search` with its output
```

▼ LeanProject.lean:11:18
▼ Tactic state
No goals
▼ Expected type
P Q : Prop
⊢ ¬(P ∧ Q) ↔ ¬P ∨ ¬Q
▼ All Messages (2)
▼ LeanProject.lean:15:2
`library_search` has been renamed to `apply?` (or `exact?` if you only want solutions closing the goal)
▼ LeanProject.lean:15:2
Try this: exact not_or

Rzk’s Online Playground

A proof assistant for ∞-category theory

- Based on CodeMirror
- GHCJS is used to compile Rzk
- Released via GitHub workflow
- https://rzk-lang.github.io/rzk/develop/playground/

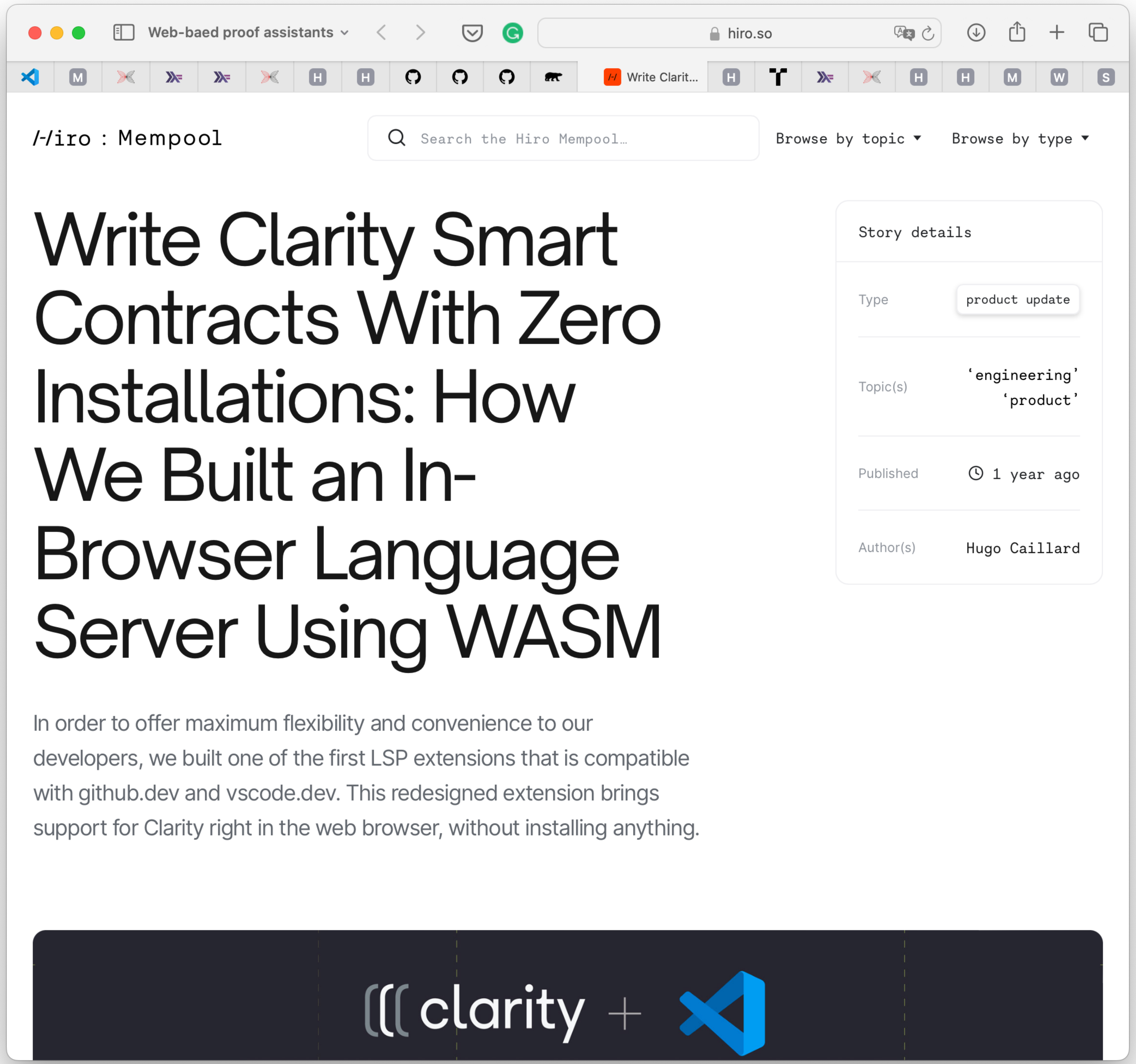


```
#lang rzk-1

-- A is contractible there exists x : A such that for any y : A we have x =
#def iscontr (A : U) : U
  := ∑ (a : A), (x : A) -> a =_{A} x

-- A is a proposition if for any x, y : A we have x = y
#def isaprop (A : U) : U
  := (x : A) -> (y : A) -> x =_{A} y

-- A is a set if for any x, y : A the type x =_{A} y is a proposition
#def isaset (A : U) : U
  := (x : A) -> (y : A) -> isaprop (x =_{A} y)

-- Non-dependent product of A and B
#def prod (A : U) (B : U) : U
  := ∑ (x : A), B

-- A function f : A -> B is an equivalence
```

```
Everything is OK!
```

Clarity

Everything works in VS Code for the Web



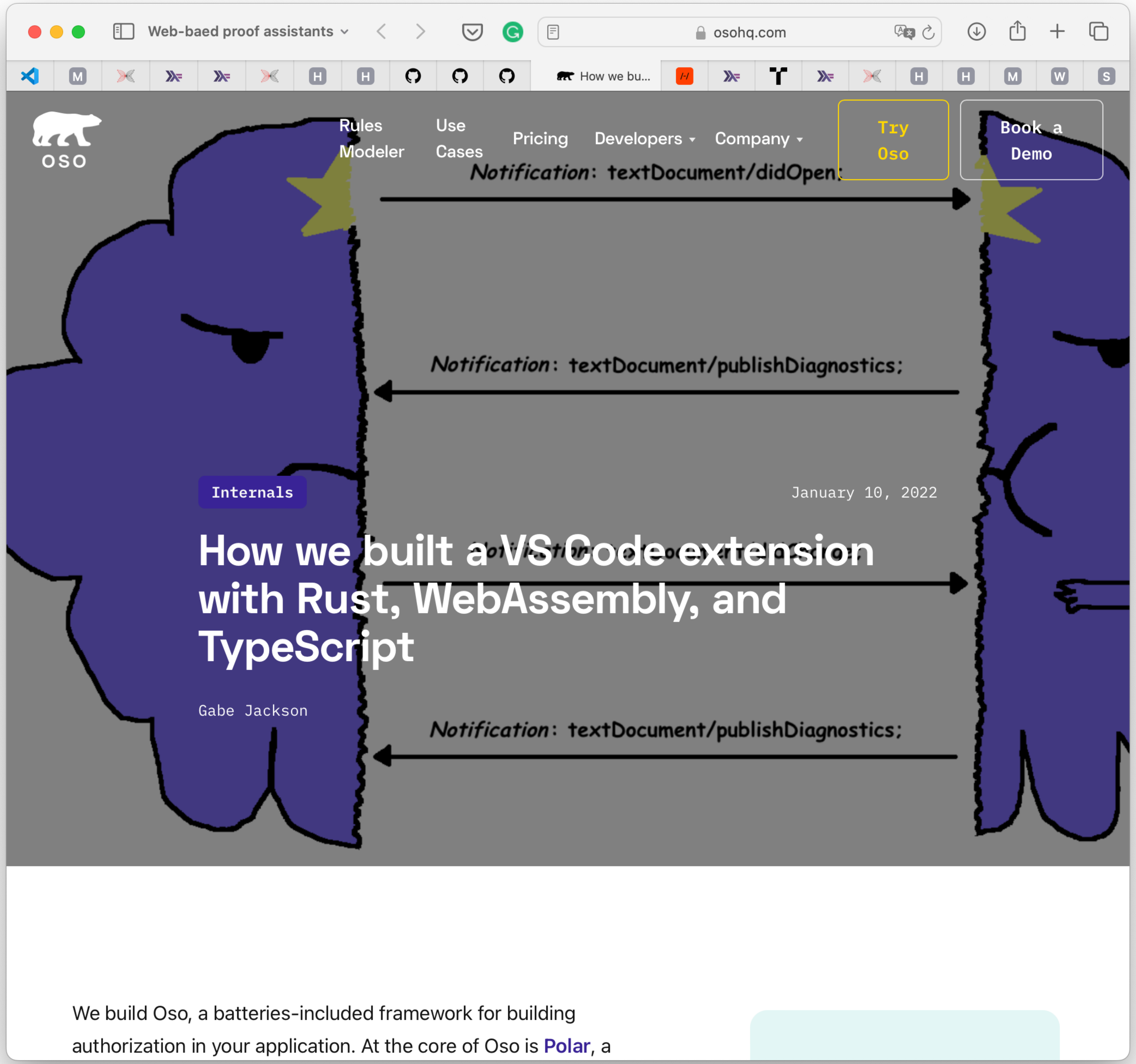
Write Clarity Smart Contracts With Zero Installations: How We Built an In-Browser Language Server Using WASM

In order to offer maximum flexibility and convenience to our developers, we built one of the first LSP extensions that is compatible with github.dev and vscode.dev. This redesigned extension brings support for Clarity right in the web browser, without installing anything.

Story details	
Type	product update
Topic(s)	'engineering' 'product'
Published	1 year ago
Author(s)	Hugo Caillard

Polar

Everything works in VS Code for the Web


Web-baed proof assistants
osohq.com
How we bu...
OSO
Rules Modeler
Use Cases
Pricing
Developers
Company
Try Oso
Book a Demo
Notification: textDocument/didOpen;
Notification: textDocument/publishDiagnostics;
Notification: textDocument/publishDiagnostics;

Internals

January 10, 2022

How we built a VS Code extension with Rust, WebAssembly, and TypeScript

Gabe Jackson

We build Oso, a batteries-included framework for building authorization in your application. At the core of Oso is Polar, a

Plan

Overview

Web Agda should be Fully Functional, not a Playground

- Use Haskell to implement the entire ecosystem, incl.
  - Language server for Agda (editor-independent)
  - VS Code extension for Agda
- The Quickstart should include only one step:
  - Go to http://agda.github.io/web-agda/
- Use Git repository to import libraries
- Native binaries can be compiled from the *same* codebase.

Things to Do

- Replace Haskell bindings to other languages with bindings to WASM
- Refactor Agda into smaller packages to reduce its size
    - agda-utils
    - `agda-type` (syntaxes)
    - `agda-core` (for parsing, type checking, termination, interaction)
    - agda-int (for interaction instances)
    - agda-backend
- Lots of performance tuning
- Implement the language server for Agda with GHC WASM (`agda-lsp`)
- Implement a new VS Code extension for Agda with GHCJS
- Add the Git support for library management and
- …etc.

Discussion

Useful Information

GHC JS

- Available through GHCup-0.1.19.5 RC
- https://discourse.haskell.org/t/ann-ghcup-0-1-19-5-release-candidate-ghc-js-cross-support/6995

GHC WASM

- Bindist for Linux is available as CI/CD artefacts
    - https://gitlab.haskell.org/ghc/ghc-wasm-meta/-/artifacts
- macOS users need to compile their own binaries.
    - https://gitlab.haskell.org/ghc/ghc-wasm-meta

Cabal

- Cabal can be configured to compile a package with GHC JS/WASM
  - Options: `with-compiler`, `with-hc-pkg`
  - https://cabal.readthedocs.io/en/3.4/cabal-project.html

Language Server Protocol

- `lsp` Haskell package: https://hackage.haskell.org/package/lsp
- The official homepage: https://microsoft.github.io/language-server-protocol/
- Agda language server: https://github.com/banacorn/agda-language-server

VS Code Extension

- Your first VS code extension: https://code.visualstudio.com/api/get-started/your-first-extension