

Textbooks that run

Interactive 0-install CS education in the browser



KC Sivaramakrishnan
26th September 2026
IndiaFOSS 6.0, Bengaluru



Story of the book

Functional Programming with OCaml

The screenshot shows the NPTEL website interface for the course 'NOC: Functional Programming with OCaml, IIT Madras' by Prof. K. C. Sivaramakrishnan. The page features a sidebar with a 'Course Details' section and a list of weeks (Week 1 to Week 8). Week 1 is expanded, showing a 'Course introduction: what you'll learn, how it's run' section. The main content area displays the course title, professor's name, and a 'Swayam Certification' button. Below this, the 'Course introduction: what you'll learn, how it's run' section is highlighted, with a 'Video' tab selected. The video player shows 'Lec 1 _ Course introduction: what you'll learn, how it's run' with a red play button. The video content includes a list of bullet points under the heading 'Why OCaml?':

- Functional-first, with a serious type system.
- Practical:
 - Jane Street, Bloomberg, Meta (Hack, Flow), Docker, Tarides, Ahrefs, Semgrep.
 - WebAssembly reference interpreter
 - CompCert verified C compiler
 - Rust's first compiler was written in OCaml.
- Fast: native code close to C; GC; no UB in the safe fragment.
- Great teaching language: ideas reappear in Rust, Scala, Haskell, Swift, Kotlin, TypeScript.

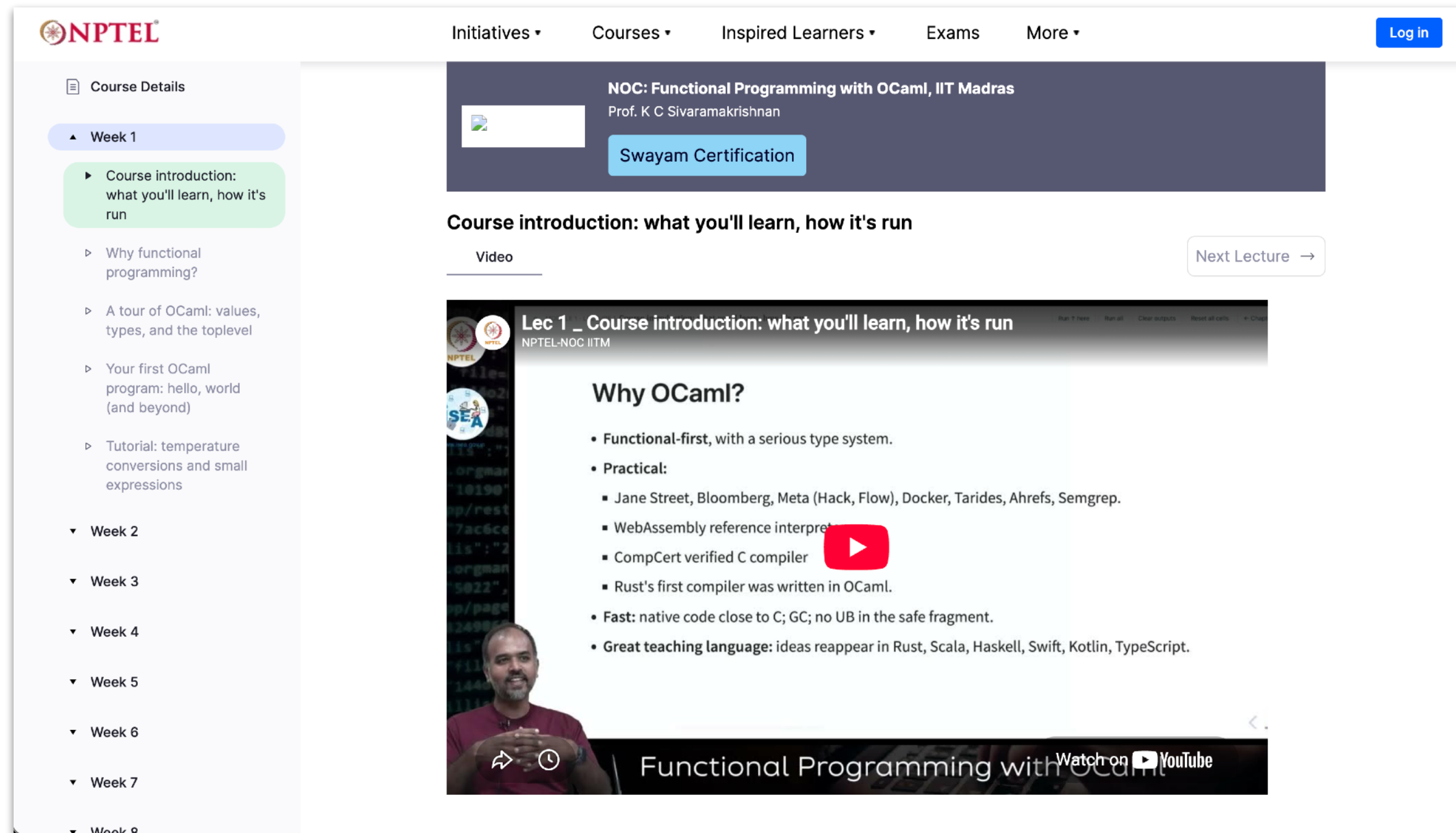
The video player also shows a 'Watch on YouTube' button and a 'Next Lecture' button.

<https://nptel.ac.in/courses/106106002>

1227 Enrolled

Story of the book

Functional Programming with OCaml



The screenshot shows the NPTEL course page for "NOC: Functional Programming with OCaml, IIT Madras" by Prof. K C Sivaramakrishnan. The page includes a sidebar with a course outline, a main header with the course title and a "Swayam Certification" button, and a video player for the first lecture. The video player shows a thumbnail of the professor and the title "Lec 1 _ Course introduction: what you'll learn, how it's run".

NPTEL

Initiatives Courses Inspired Learners Exams More Log in

Course Details

Week 1

- Course introduction: what you'll learn, how it's run
- Why functional programming?
- A tour of OCaml: values, types, and the toplevel
- Your first OCaml program: hello, world (and beyond)
- Tutorial: temperature conversions and small expressions

Week 2

Week 3

Week 4

Week 5

Week 6

Week 7

Week 8

NOC: Functional Programming with OCaml, IIT Madras

Prof. K C Sivaramakrishnan

Swayam Certification

Course introduction: what you'll learn, how it's run

Video

Next Lecture →

Lec 1 _ Course introduction: what you'll learn, how it's run

NPTEL-NOC IITM

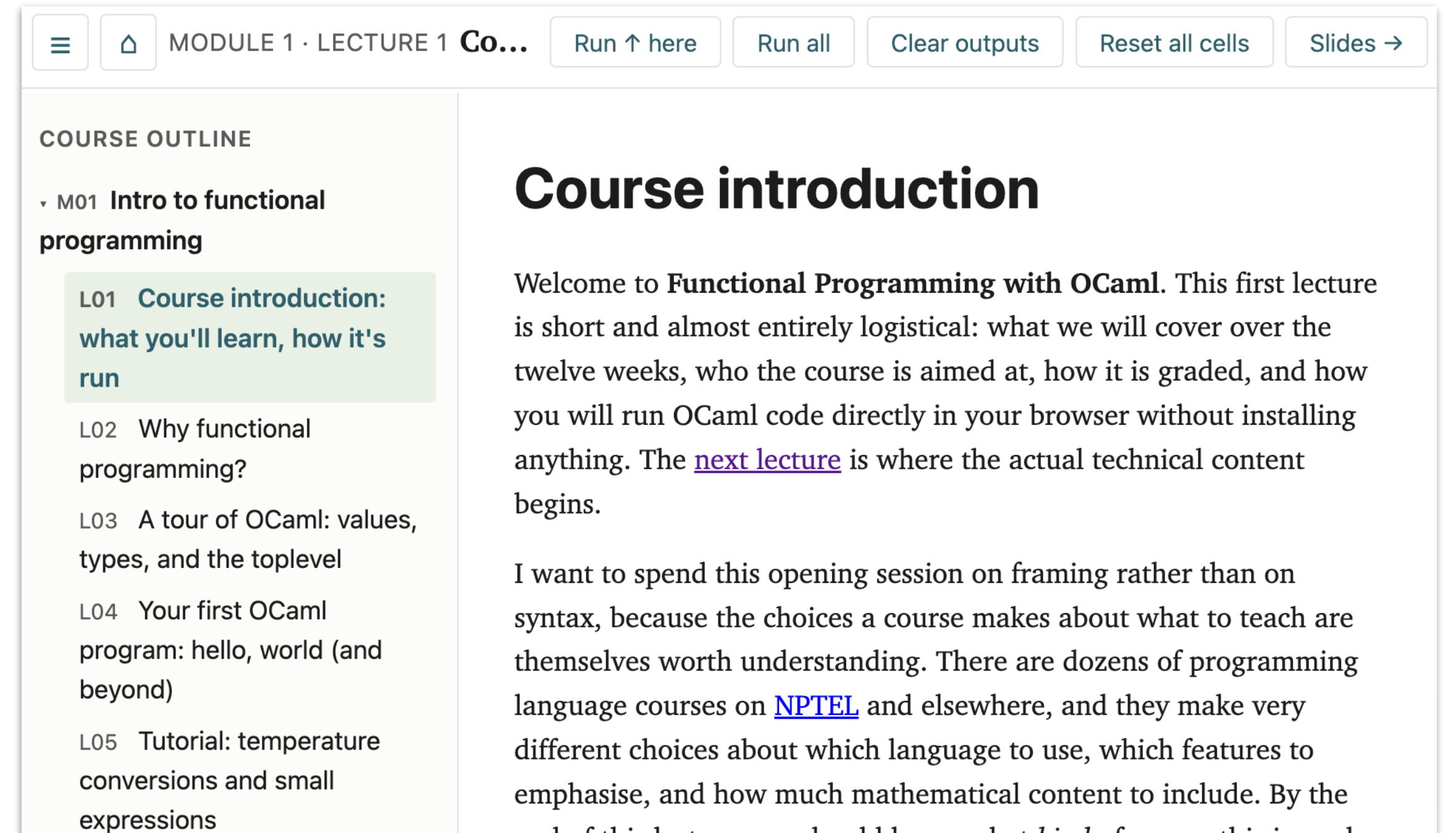
Why OCaml?

- Functional-first, with a serious type system.
- Practical:
 - Jane Street, Bloomberg, Meta (Hack, Flow), Docker, Tarides, Ahrefs, Semgrep.
 - WebAssembly reference interpreter
 - CompCert verified C compiler
 - Rust's first compiler was written in OCaml.
- Fast: native code close to C; GC; no UB in the safe fragment.
- Great teaching language: ideas reappear in Rust, Scala, Haskell, Swift, Kotlin, TypeScript.

Functional Programming with OCaml

Watch on YouTube

<https://nptel.ac.in/courses/106106002>



The screenshot shows the fplaunchpad.org course page for "MODULE 1 · LECTURE 1 Co...". The page includes a header with navigation buttons, a course outline, and a main content area for the "Course introduction".

MODULE 1 · LECTURE 1 Co... Run ↑ here Run all Clear outputs Reset all cells Slides →

COURSE OUTLINE

- M01 Intro to functional programming
 - L01 Course introduction: what you'll learn, how it's run
 - L02 Why functional programming?
 - L03 A tour of OCaml: values, types, and the toplevel
 - L04 Your first OCaml program: hello, world (and beyond)
 - L05 Tutorial: temperature conversions and small expressions

Course introduction

Welcome to **Functional Programming with OCaml**. This first lecture is short and almost entirely logistical: what we will cover over the twelve weeks, who the course is aimed at, how it is graded, and how you will run OCaml code directly in your browser without installing anything. The [next lecture](#) is where the actual technical content begins.

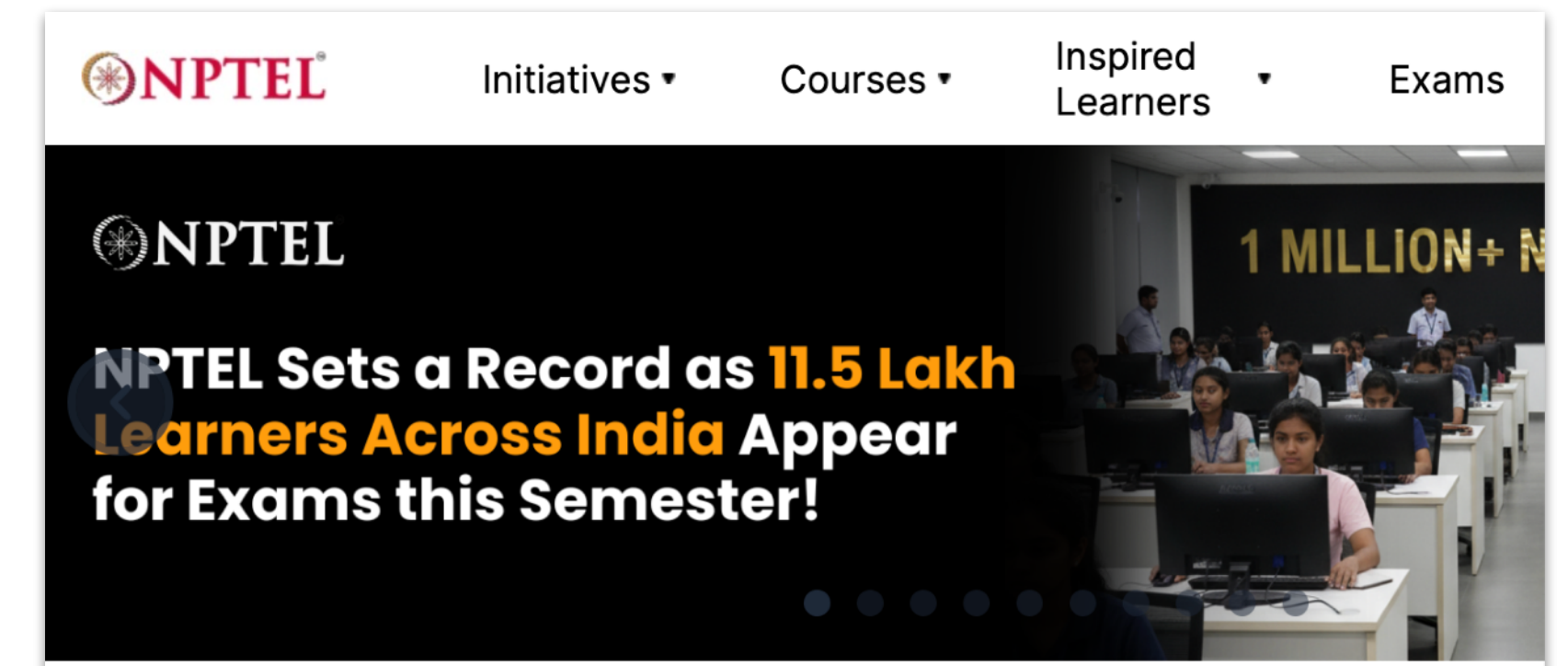
I want to spend this opening session on framing rather than on syntax, because the choices a course makes about what to teach are themselves worth understanding. There are dozens of programming language courses on [NPTEL](#) and elsewhere, and they make very different choices about which language to use, which features to emphasise, and how much mathematical content to include. By the end of this lecture you should know what kind of course this is, and

https://fplaunchpad.org/ocaml_nptel

1227 Enrolled

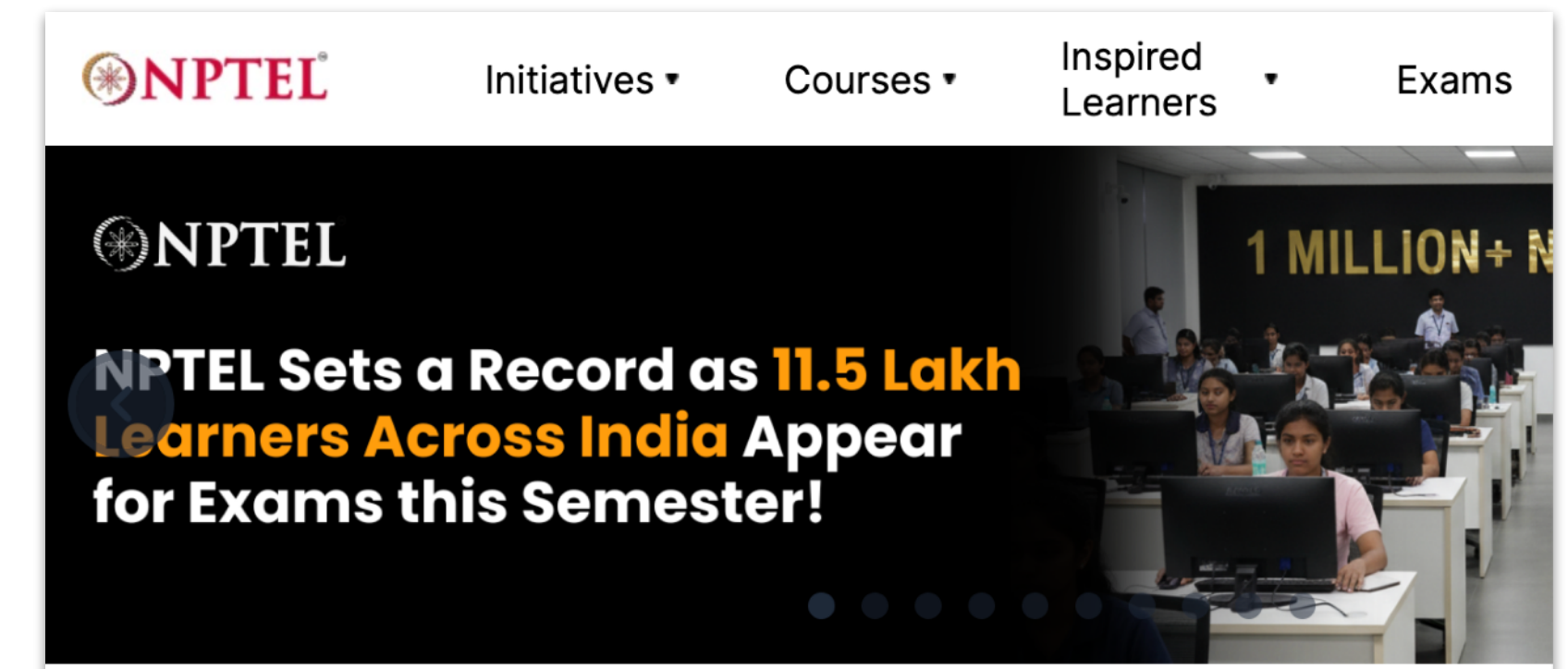
NPTEL

- A free online MOOC platform with courses from IITs and IISc
 - Videos are available on YouTube
 - A student can be “certified” by taking a proctored exam ₹1k per course
 - Credits can satisfy degrees in other colleges and universities



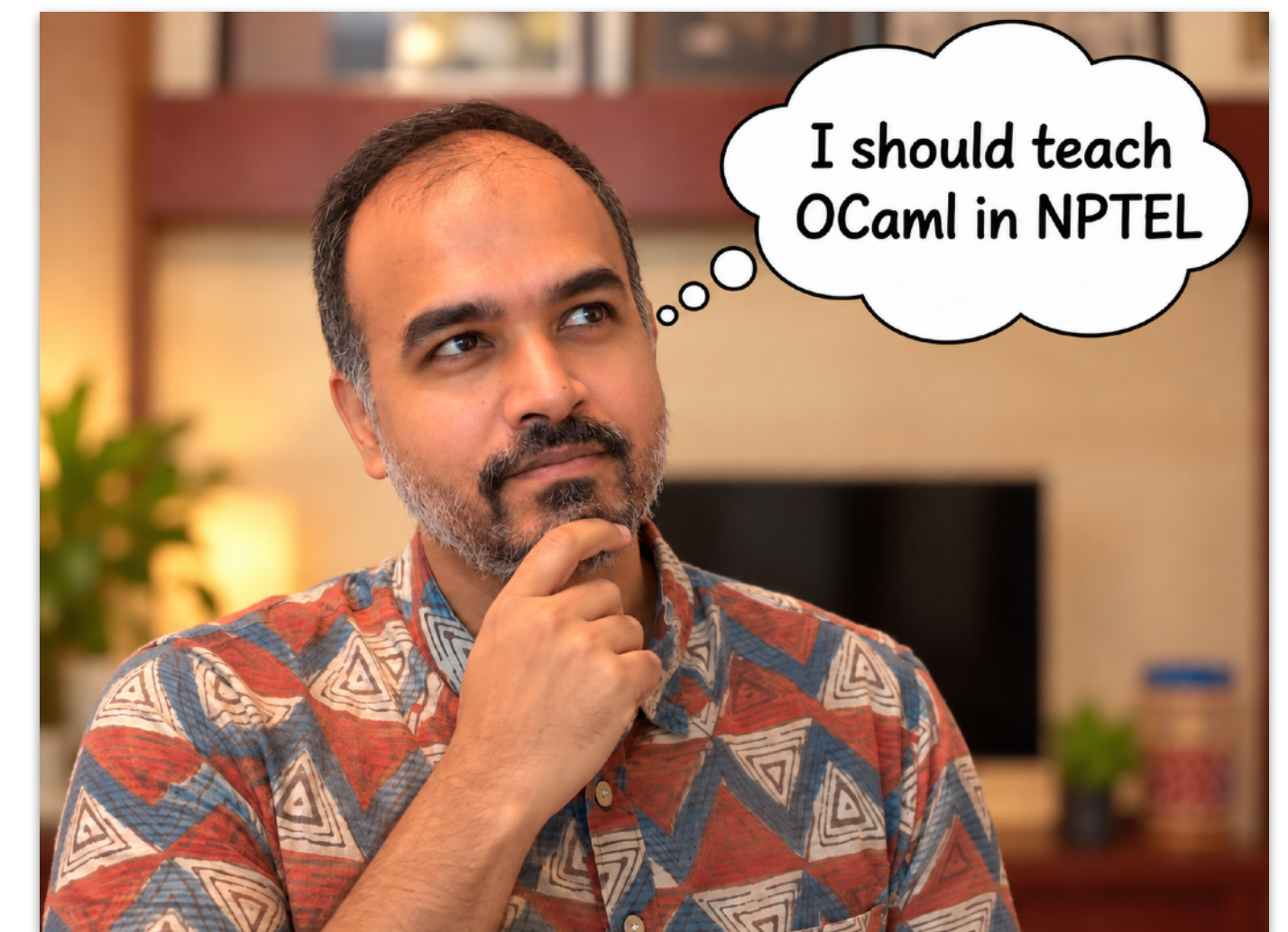
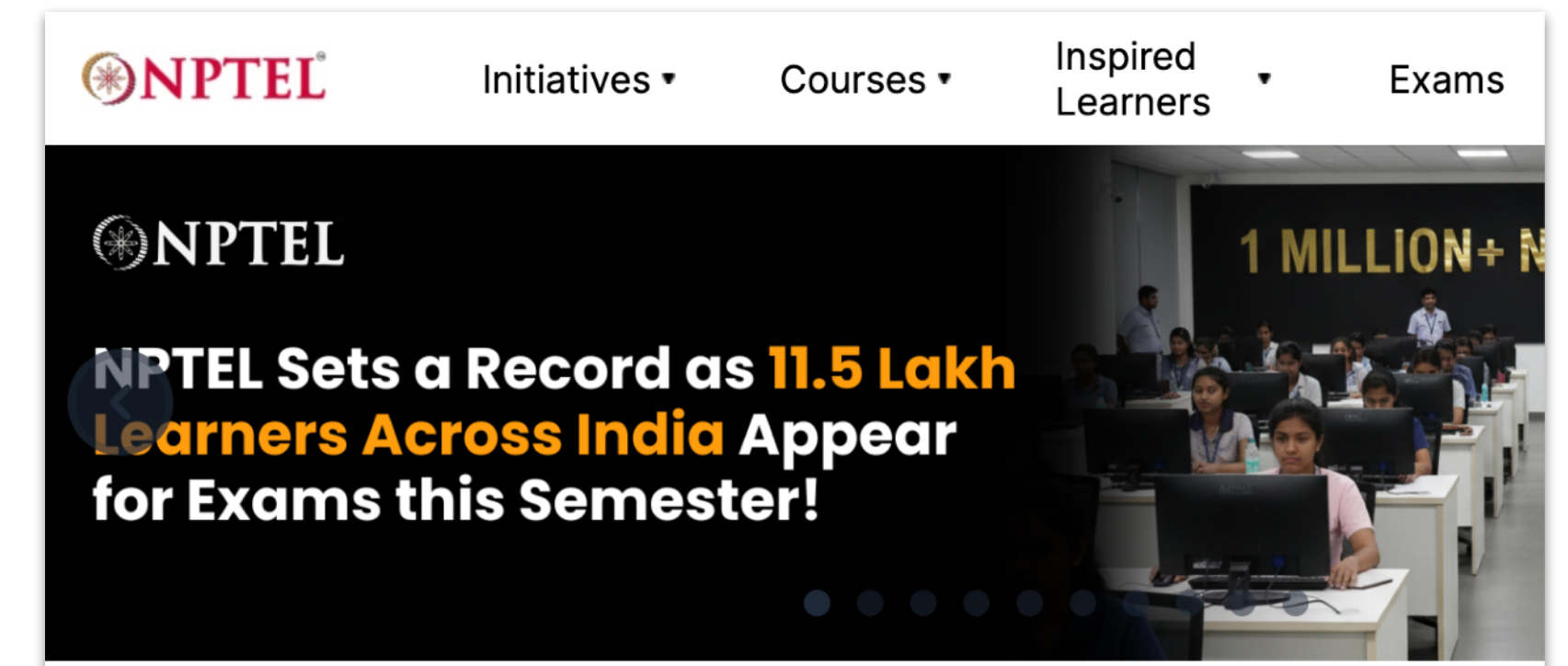
NPTEL

- A free online MOOC platform with courses from IITs and IISc
 - Videos are available on YouTube
 - A student can be “certified” by taking a proctored exam ₹1k per course
 - Credits can satisfy degrees in other colleges and universities
- **One of the greatest collective achievements of Indian academia**



NPTEL

- A free online MOOC platform with courses from IITs and IISc
 - Videos are available on YouTube
 - A student can be “certified” by taking a proctored exam ₹1k per course
 - Credits can satisfy degrees in other colleges and universities
- **One of the greatest collective achievements of Indian academia**



Delivery challenge

- Programming languages are learnt by poking at it

Delivery challenge

- Programming languages are learnt by poking at it
- *How to get OCaml platform to 1227 students?*
 - Shared computers or lack of it
 - platform diversity
 - lack of broader computer skills
 - unreliable Internet

Delivery challenge

- Programming languages are learnt by poking at it
- *How to get OCaml platform to 1227 students?*
 - Shared computers or lack of it
 - platform diversity
 - lack of broader computer skills
 - unreliable Internet
- And the millions (👉) who will watch the videos later

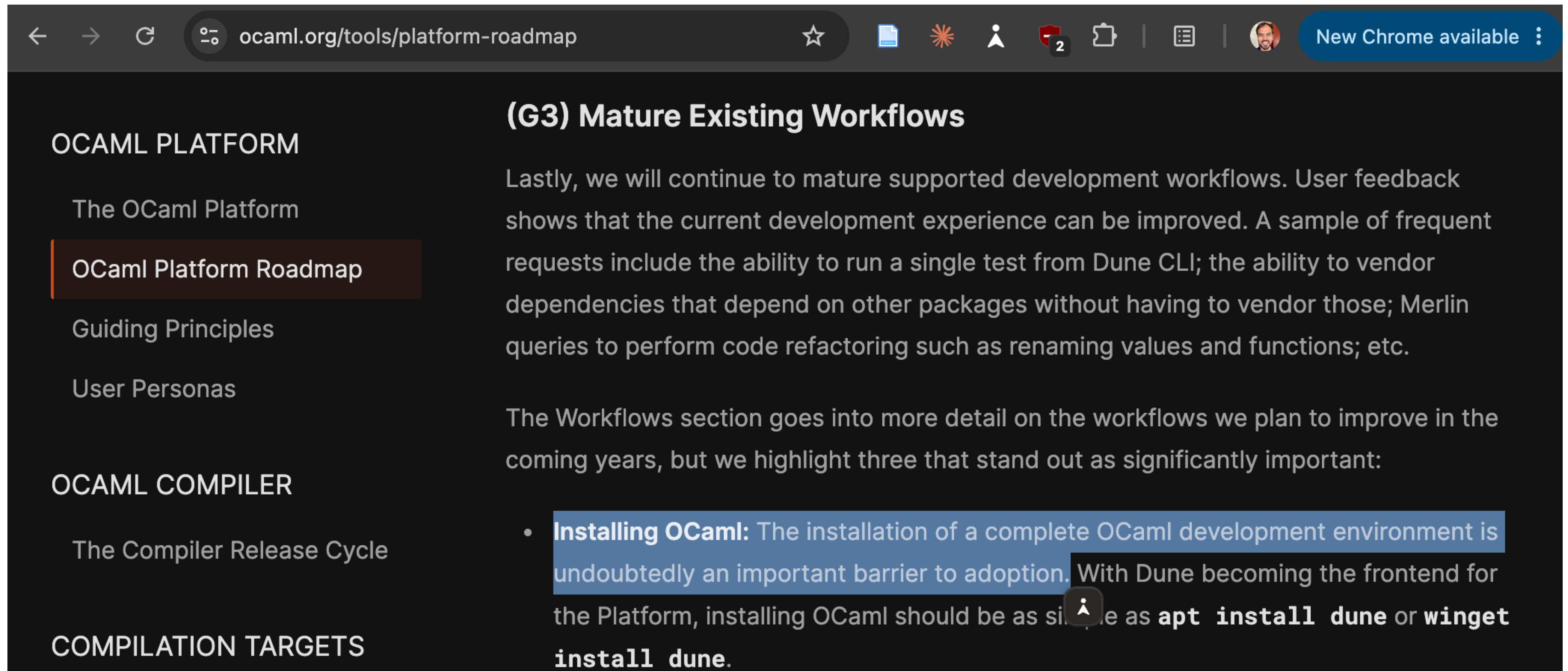
Delivery challenge

- Programming languages are learnt by poking at it
- *How to get OCaml platform to 1227 students?*
 - Shared computers or lack of it
 - platform diversity
 - lack of broader computer skills
 - unreliable Internet
- And the millions (👉) who will watch the videos later
- Videos recorded in a studio
 - 2 TAs to run the course
 - Personalised attention is impossible



Installation challenge

- A big challenge for *learning* and *adoption*



The screenshot shows a web browser window with the address bar displaying `ocaml.org/tools/platform-roadmap`. The page has a dark theme. On the left, a sidebar lists navigation items: "OCAML PLATFORM", "The OCaml Platform", "OCaml Platform Roadmap" (highlighted with an orange bar), "Guiding Principles", "User Personas", "OCAML COMPILER", "The Compiler Release Cycle", and "COMPILATION TARGETS". The main content area is titled "(G3) Mature Existing Workflows". It contains a paragraph about maturing development workflows and a list of challenges. One item in the list, "Installing OCaml", is highlighted with a blue background. The browser's top bar shows various icons and a notification for "New Chrome available".

OCAML PLATFORM

The OCaml Platform

OCaml Platform Roadmap

Guiding Principles

User Personas

OCAML COMPILER

The Compiler Release Cycle

COMPILATION TARGETS

(G3) Mature Existing Workflows

Lastly, we will continue to mature supported development workflows. User feedback shows that the current development experience can be improved. A sample of frequent requests include the ability to run a single test from Dune CLI; the ability to vendor dependencies that depend on other packages without having to vendor those; Merlin queries to perform code refactoring such as renaming values and functions; etc.

The Workflows section goes into more detail on the workflows we plan to improve in the coming years, but we highlight three that stand out as significantly important:

- **Installing OCaml:** The installation of a complete OCaml development environment is undoubtedly an important barrier to adoption. With Dune becoming the frontend for the Platform, installing OCaml should be as simple as `apt install dune` or `winget install dune`.

Installation challenge

- A big challenge for *learning* and *adoption*
- Personal experience
 - IITM — many students unfamiliar with terminal
 - Hands-on OCaml workshops — 1/3rd of the time spent on installation

Installation challenge

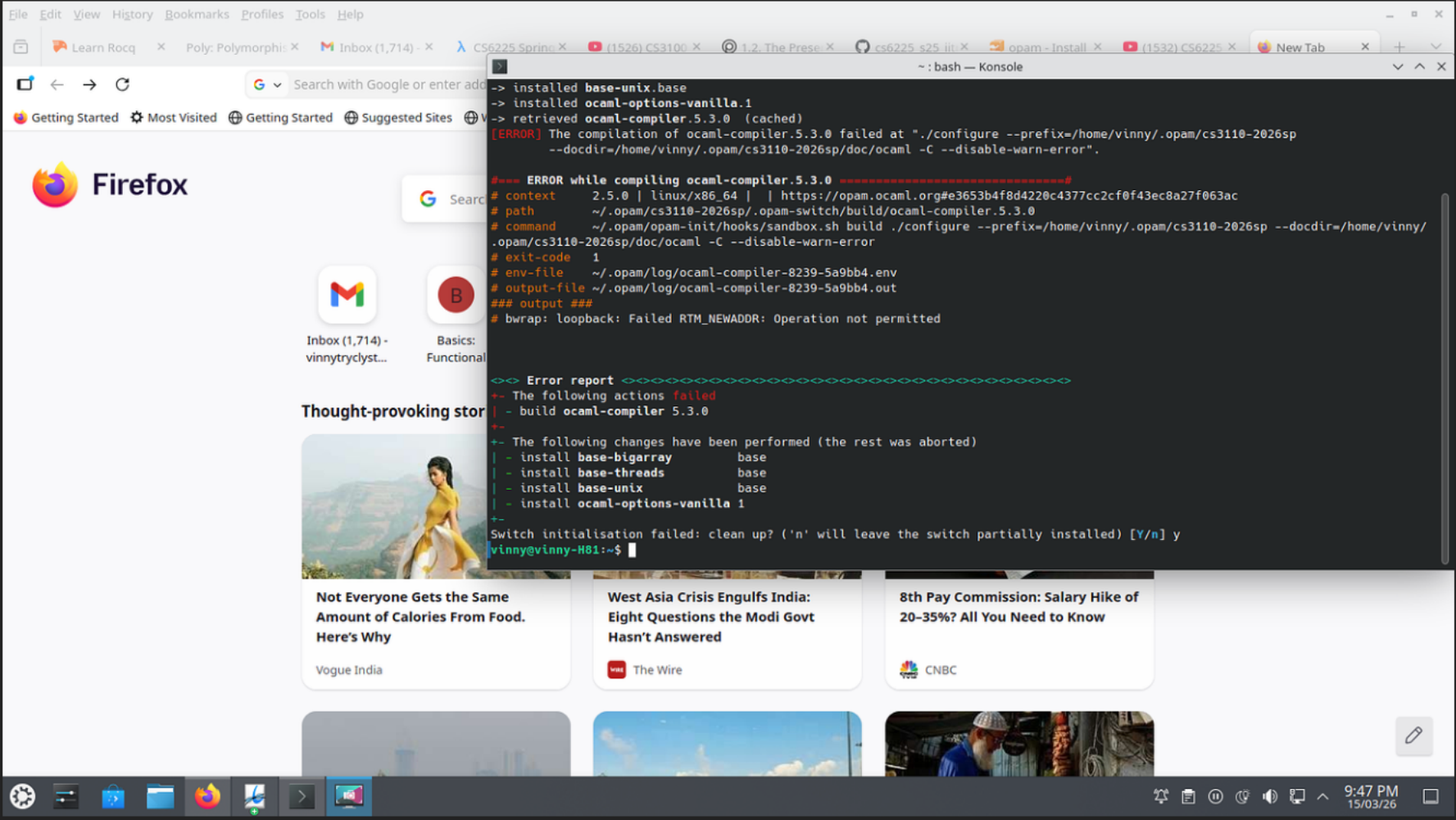
- A big challenge for *learning* and *adoption*
- Personal experience
 - IITM — many students unfamiliar with terminal
 - Hands-on OCaml workshops — 1/3rd of the time spent on installation
- Users and students *disengage* when installation is a challenge
 - Especially challenging when students have little support

 OCaml

OPAM installation error

Learning

 Vinod Mar 15



I am trying to install OPAM to start learning OCaml. I used `sudo apt install opam` to install the package. The version installed was not up to date, so I tried the command `opam update`, which did not work. Then I downloaded and installed the latest binary distribution. Now the command `opam update` is working, but then I am not able to create an opam switch. I get the error that I have attached in the screenshot. Can anyone please help me rectify this error?

Vinod.

 Me too

Who do we not hear from?

Earlier workflow

- CS3100 Paradigms of Programming
 - OCaml + Prolog
- Jupyter Notebooks + Docker
 - RISE for converting them to reveal.js slides
 - nbgrader for assignment evaluation
 - Several offerings at IITM with this set up.

Referential transparency

Consider the function `bar` :

```
[8]: let bar x = x + next()
```

```
[8]:      val bar : int -> int = <fun>
```

```
[14]: let qux = bar 222
```

```
[14]:      val qux : int = 229
```

Can we now optimise `qux` to:

```
[ ]: let qux = 19|
```

NO! Referential transparency breaks under side effects.

Earlier workflow

- CS3100 Paradigms of Programming
 - OCaml + Prolog
- Jupyter Notebooks + Docker
 - RISE for converting them to reveal.js slides
 - nbgrader for assignment evaluation
 - Several offerings at IITM with this set up.
- Downsides
 - Needs Docker — Image is **2.86GB**
 - Editor is not language-aware — no types, live error, code completion
 - Not a book — cannot be read in isolation; meant to go along with lecture videos

Referential transparency

Consider the function `bar` :

```
[8]: let bar x = x + next()
```

```
[8]:      val bar : int -> int = <fun>
```

```
[14]: let qux = bar 222
```

```
[14]:      val qux : int = 229
```

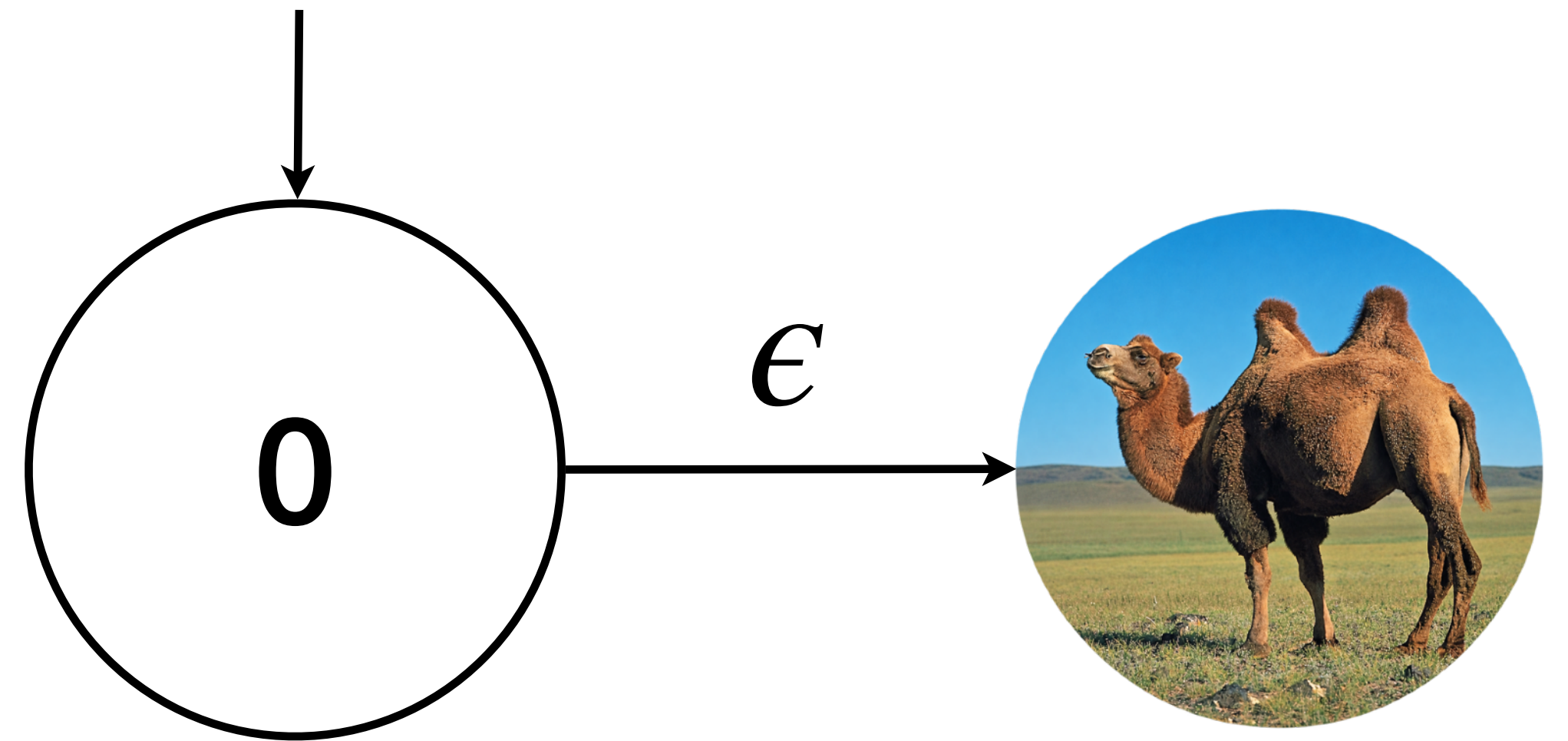
Can we now optimise `qux` to:

```
[ ]: let qux = 19|
```

NO! Referential transparency breaks under side effects.

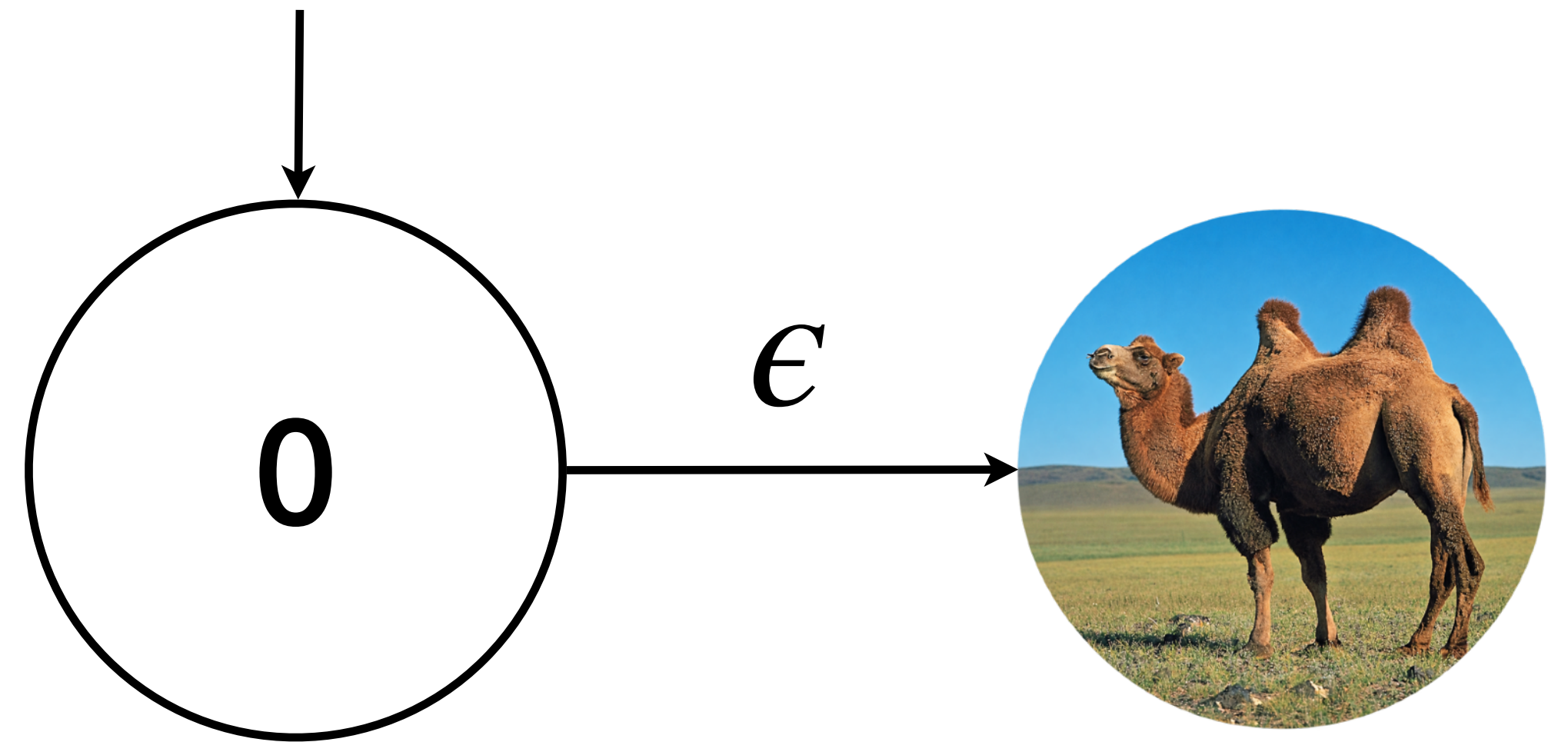
0 to OCaml in 0 steps

- **Start immediately**
 - Zero installation; runs offline in the browser



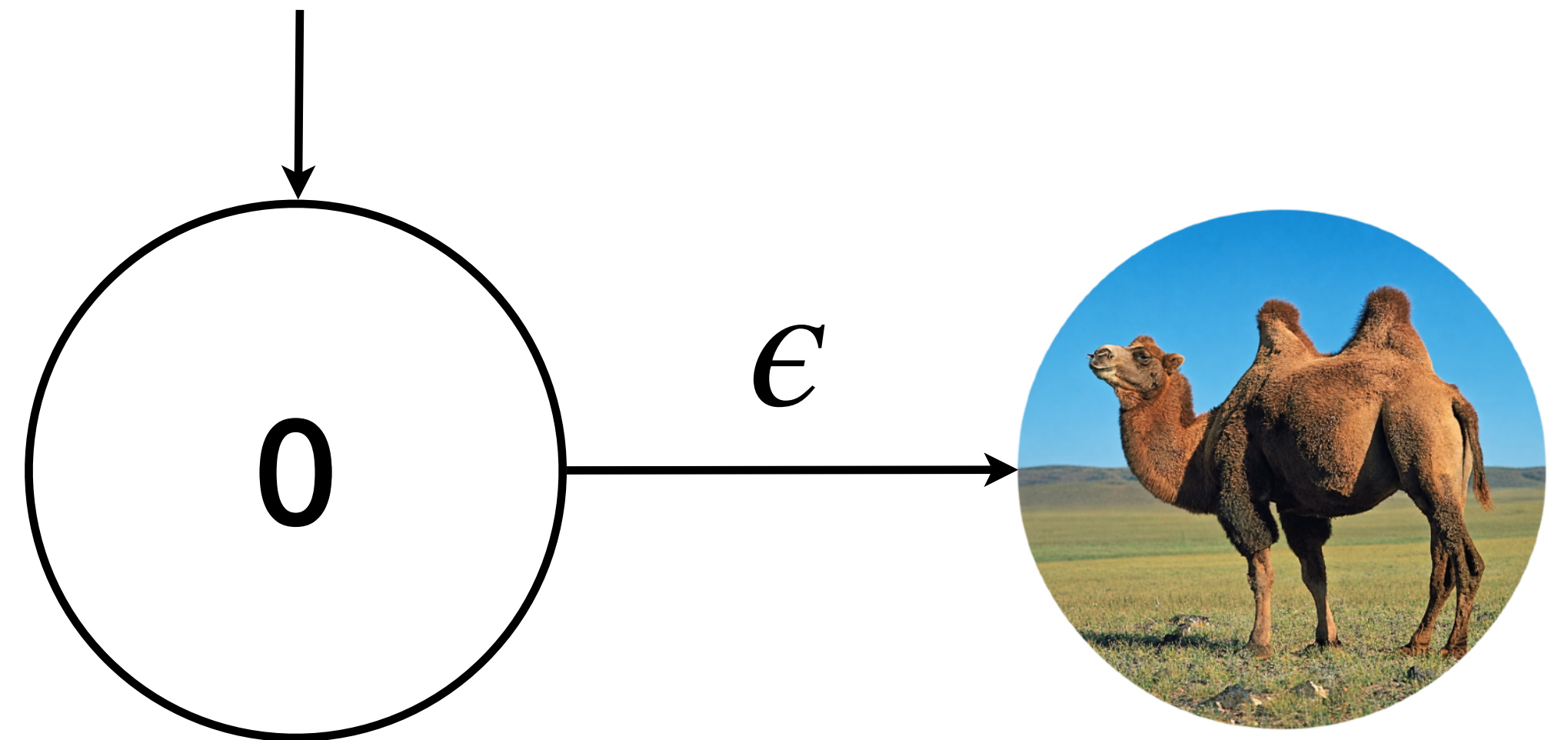
0 to OCaml in 0 steps

- **Start immediately**
 - Zero installation; runs offline in the browser
- **Learn by doing**
 - Types on hover, live errors and autocompletion
 - Immediate quiz feedback



0 to OCaml in 0 steps

- **Start immediately**
 - Zero installation; runs offline in the browser
- **Learn by doing**
 - Types on hover, live errors and autocompletion
 - Immediate quiz feedback
- **Keep the book maintainable**
 - One source for textbook and executable slides
 - Readable Git diffs and merges
 - Automatically type-check and test the code



Demo

- Running the code right in the page
 - changing it, IDE features, presenting as a slide

Demo

- Running the code right in the page
 - changing it, IDE features, presenting as a slide
- OxCaml, a performance-oriented extension of OCaml



Demo

- Running the code right in the page
 - changing it, IDE features, presenting as a slide
- OxCaml, a performance-oriented extension of OCaml
- Full Linux VM, compiled to Wasm
 - Why? Insecurity of C — buffer-overflow, use-after-free, double-free



Demo

- Running the code right in the page
 - changing it, IDE features, presenting as a slide
- OxCaml, a performance-oriented extension of OCaml
- Full Linux VM, compiled to Wasm
 - Why? Insecurity of C — buffer-overflow, use-after-free, double-free
- The joy of DOM interactivity



Tech — OCaml Platform

- **js_of_ocaml (jsoc)** — A compiler from OCaml and OCaml to JavaScript

Tech — OCaml Platform

- **js_of_ocaml (jsoc)** — A compiler from OCaml and OCaml to JavaScript
- OCaml compiler itself is written in OCaml
 - Platform tools too: **Merlin** — OCaml Intelligence, **OCamlFormat** — code formatting
 - *Can compile the OCaml compiler to JavaScript using jsoc!*

Tech — OCaml Platform

- **js_of_ocaml (jsoc)** — A compiler from OCaml and Ocaml to JavaScript
- OCaml compiler itself is written in OCaml
 - Platform tools too: **Merlin** — OCaml Intelligence, **OCamlFormat** — code formatting
 - *Can compile the OCaml compiler to JavaScript using jsoc!*
- **x-ocaml** — OCaml notebook as WebComponents

```
<x-ocaml>let x = 42</x-ocaml>
```

```
15 let x = 42  
   val x : int = 42
```

Run ▶

Tech — OCaml Platform

- **js_of_ocaml (jsoc)** — A compiler from OCaml and OCaml to JavaScript
- OCaml compiler itself is written in OCaml
 - Platform tools too: **Merlin** — OCaml Intelligence, **OCamlFormat** — code formatting
 - *Can compile the OCaml compiler to JavaScript using jsoc!*
- **x-ocaml** — OCaml notebook as WebComponents

```
<x-ocaml>let x = 42</x-ocaml>
```

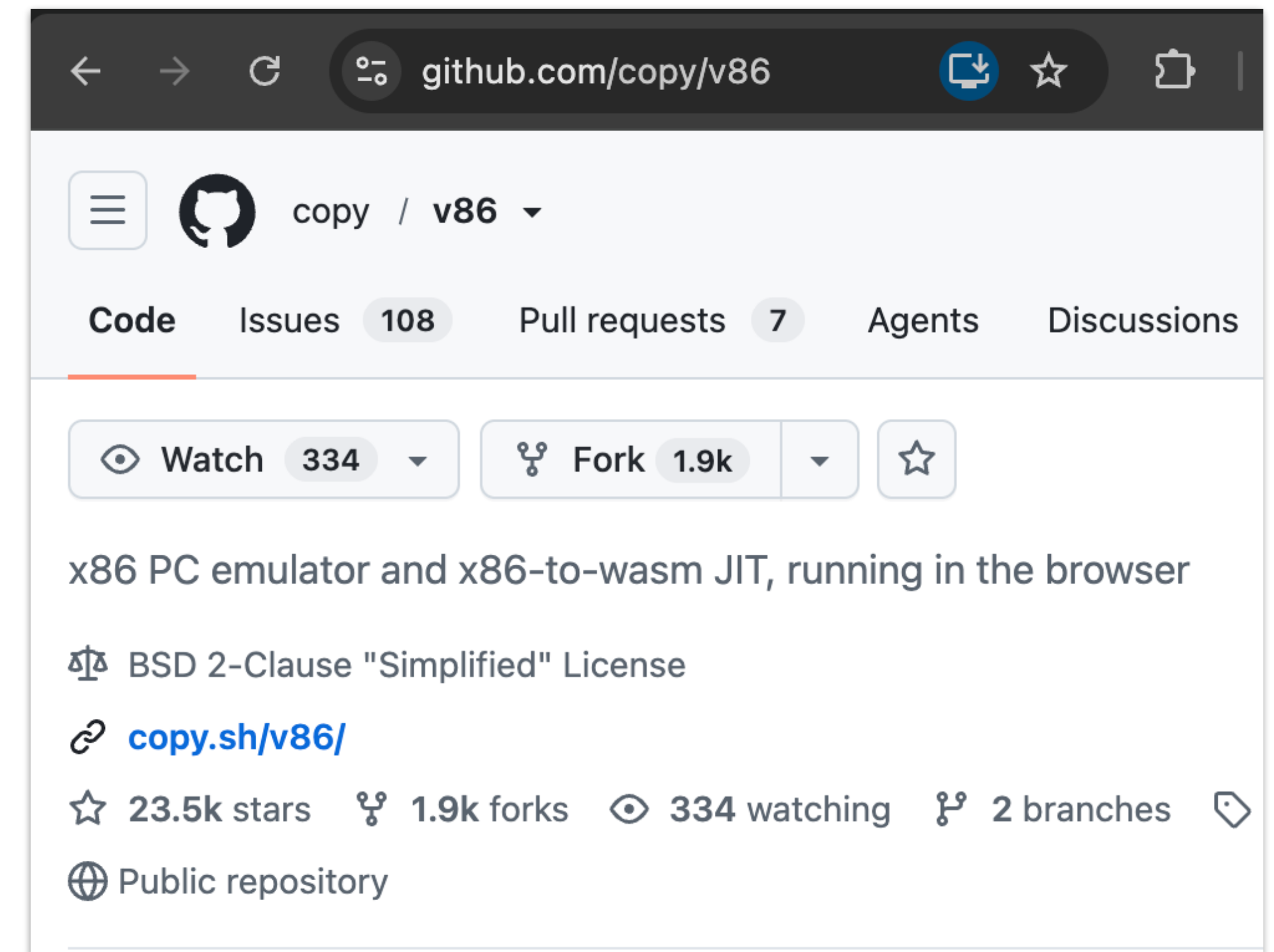
```
15 let x = 42
    val x : int = 42
```

Run ▶

***Liberally licensed
Open Source software***

Tech — Linux VM in the browser

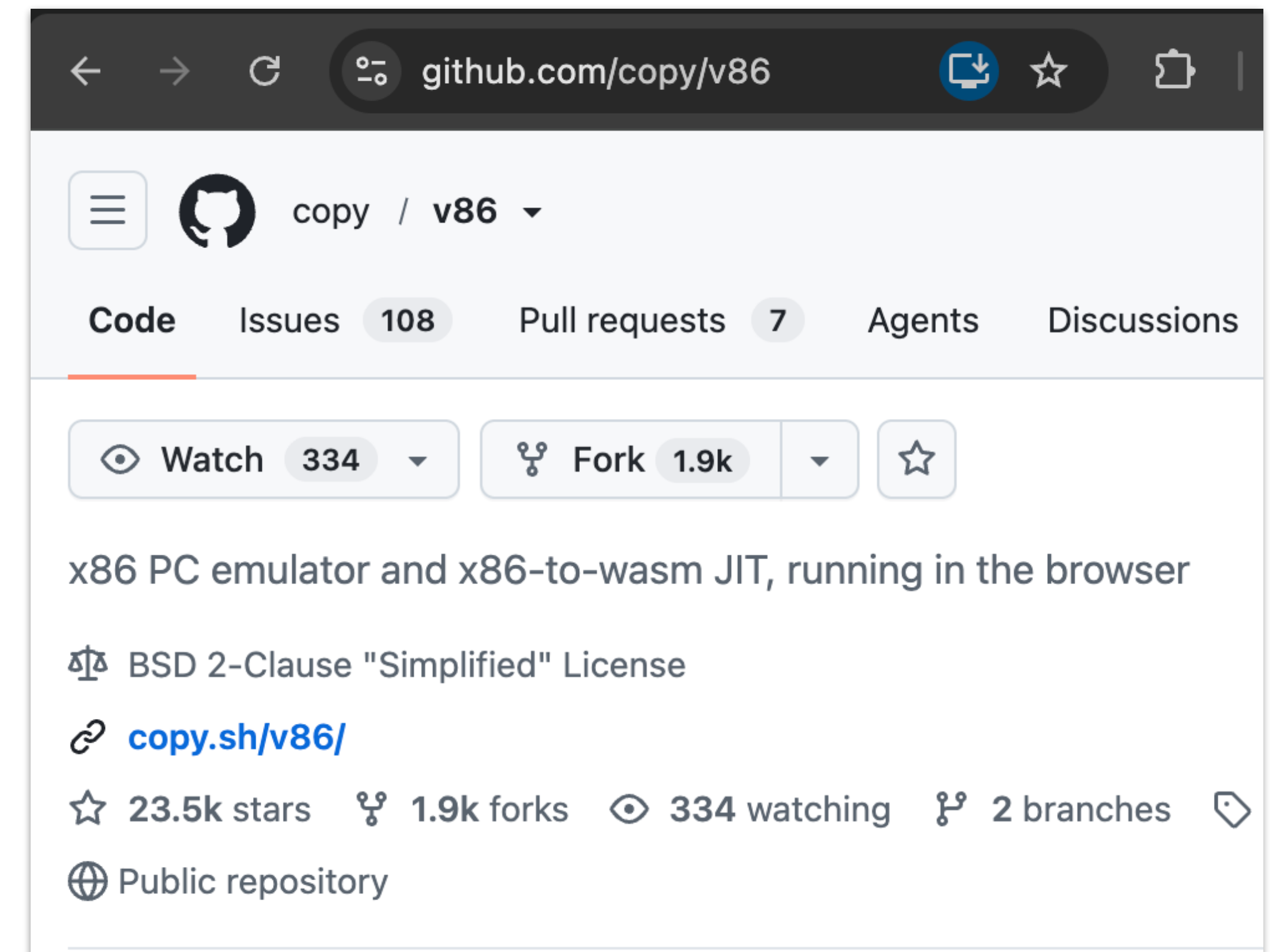
- x86 Alpine Linux image built with Docker
 - With OCaml, dune and course projects preinstalled
- v86 JIT compiles x86 to Wasm



<https://github.com/copy/v86>

Tech — Linux VM in the browser

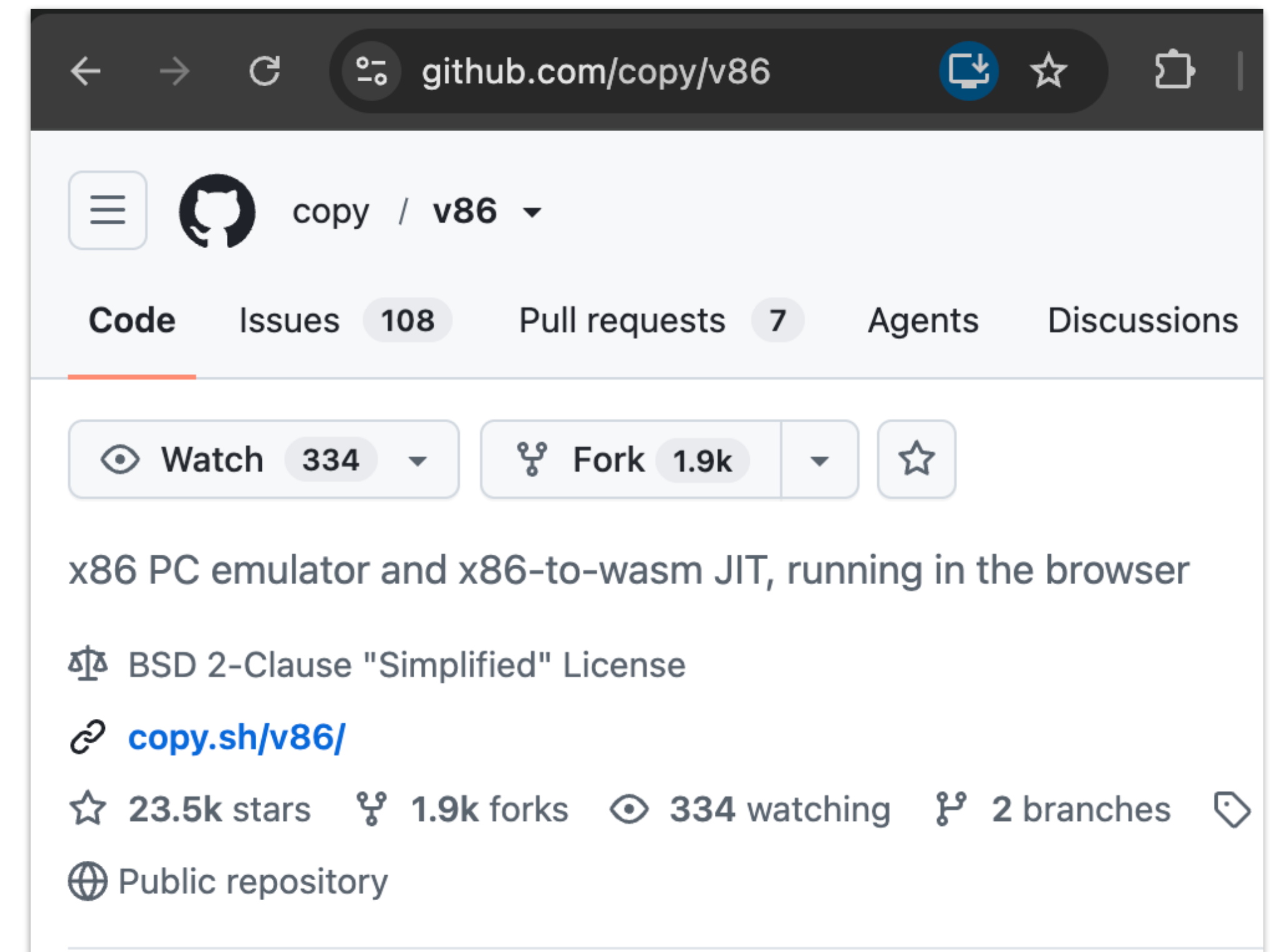
- x86 Alpine Linux image built with Docker
 - With OCaml, dune and course projects preinstalled
- v86 JIT compiles x86 to Wasm
- **~12 MB** to a shell
- No installation or server-side execution



<https://github.com/copy/v86>

Tech — Linux VM in the browser

- x86 Alpine Linux image built with Docker
 - With OCaml, dune and course projects preinstalled
 - v86 JIT compiles x86 to Wasm
- **~12 MB** to a shell
 - No installation or server-side execution
- Resumes a compressed snapshot
 - Not cold-boot
 - Loads filesystem chunks on demand



<https://github.com/copy/v86>

Embedded Quizzes

- Book contains embedded quizzes in lectures
 - Both MCQs and programming puzzles

Embedded Quizzes

- Book contains embedded quizzes in lectures
 - Both MCQs and programming puzzles
- Results are anonymously collected
 - Analytics available publicly

Embedded Quizzes

- Book contains embedded quizzes in lectures
 - Both MCQs and programming puzzles
- Results are anonymously collected
 - Analytics available publicly
- Feedback in both directions
 - Reader about their understanding
 - Author about their explanation
- *Aim to revise the book based on analytics!*

Profiling Programming Language Learning

[WILL CRICHTON](#) and [SHRIRAM KRISHNAMURTHI](#), Brown University, USA

This paper documents a year-long experiment to “profile” the process of learning a programming language: gathering data to understand what makes a language hard to learn, and using that data to improve the learning process. We added interactive quizzes to *The Rust Programming Language*, the official textbook for learning Rust. Over 13 months, 62,526 readers answered questions 1,140,202 times. First, we analyze the trajectories of readers. We find that many readers drop-out of the book early when faced with difficult language concepts like Rust’s ownership types. Second, we use classical test theory and item response theory to analyze the characteristics of quiz questions. We find that better questions are more conceptual in nature, such as asking why a program does not compile vs. whether a program compiles. Third, we performed 12 interventions into the book to help readers with difficult questions. We find that on average, interventions improved quiz scores on the targeted questions by +20%. Fourth, we show that our technique can likely generalize to languages with smaller user bases by simulating our statistical inferences on small N . These results demonstrate that quizzes are a simple and useful technique for understanding language learning at all scales.

Authoring the modules

- Book is written in Markdown with Pandoc-style fenced divs

```
:::slide
```

```
## What's next
```

```
Lecture 5: **local functions and mutual recursion**.
```

- The ``go``-inside-a-function pattern, made explicit.
- Functions that refer to each other.

```
:::
```

What's next

Lecture 5: **local functions and mutual recursion**.

- The `go` -inside-a-function pattern, made explicit.
- Functions that refer to each other.

Authoring the modules

- Book is written in Markdown with Pandoc-style fenced divs

```
 :::quiz code id=M03-L04-q2
Write a tail-recursive `sum_of_squares : int -> int` that returns
`1*1 + 2*2 + ... + n*n` for non-negative `n`. `sum_of_squares 0 = 0`.

```ocaml
let sum_of_squares n =
 failwith "not implemented"
```

```ocaml skip
let check b m = if not b then failwith m
let () =
 check (sum_of_squares 0 = 0) "zero";
 check (sum_of_squares 1 = 1) "one";
 check (sum_of_squares 3 = 14) "small: 1 + 4 + 9";
 check (sum_of_squares 5 = 55) "five: 1 + 4 + 9 + 16 + 25";
 print_endline "all tests passed"
```

```

Write a tail-recursive `sum_of_squares : int -> int` that returns `1*1 + 2*2 + ... + n*n` for non-negative `n`. `sum_of_squares 0 = 0`.

```
55 let sum_of_squares n = failwith "not implemented"
```

 Run 

Check 

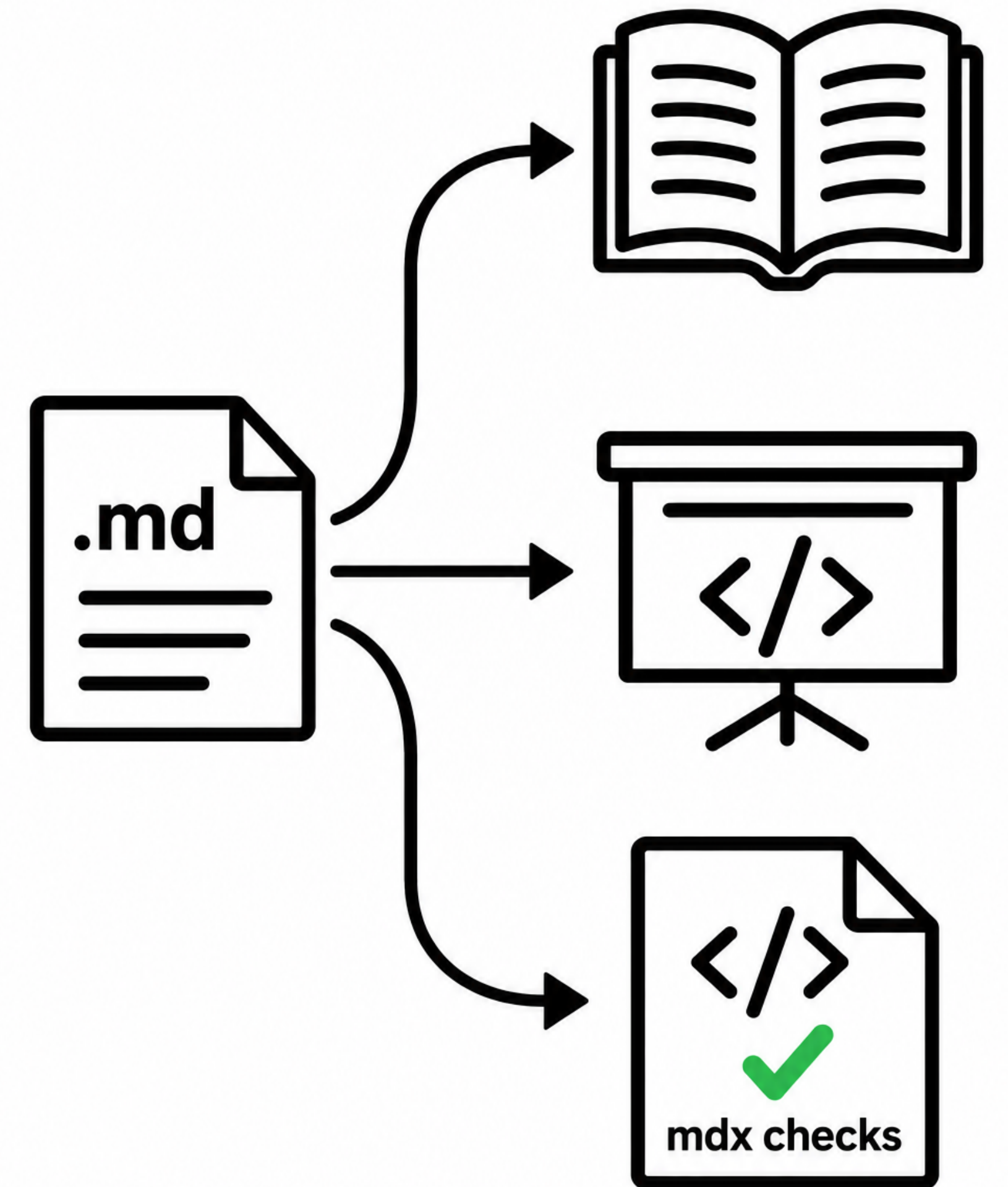
```
56 let check b m = if not b then failwith m
57
58 let () =
59   check (sum_of_squares 0 = 0) "zero";
60   check (sum_of_squares 1 = 1) "one";
61   check (sum_of_squares 3 = 14) "small: 1 + 4 + 9";
62   check (sum_of_squares 5 = 55) "five: 1 + 4 + 9 +
63     16 + 25";
   print_endline "all tests passed"
```

Run 

[► Show reference solution](#)

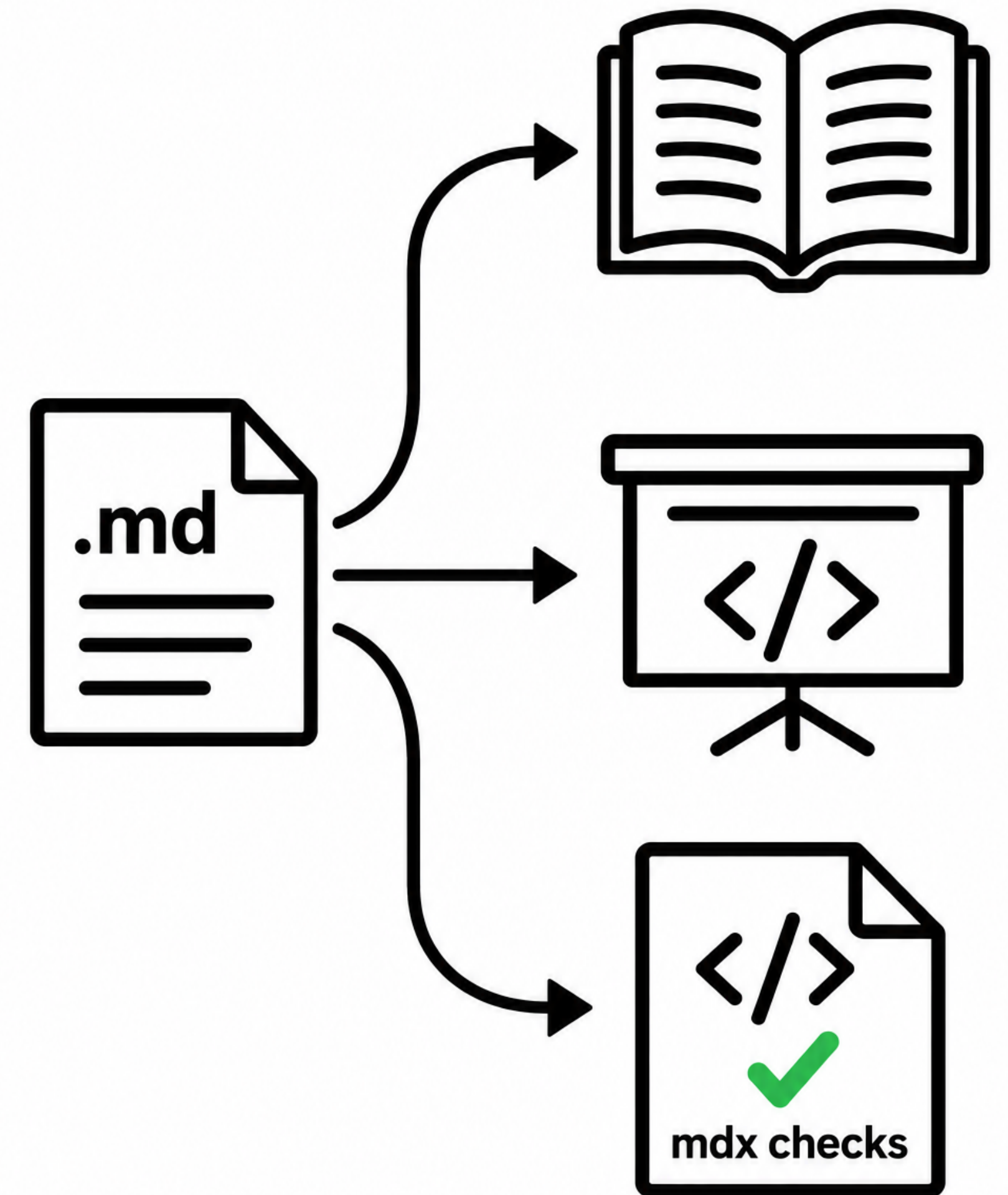
Authoring the modules

- **One markdown source for textbook and slides**
 - Helps keeps explanations and examples consistent
 - Readable Git diffs and merges



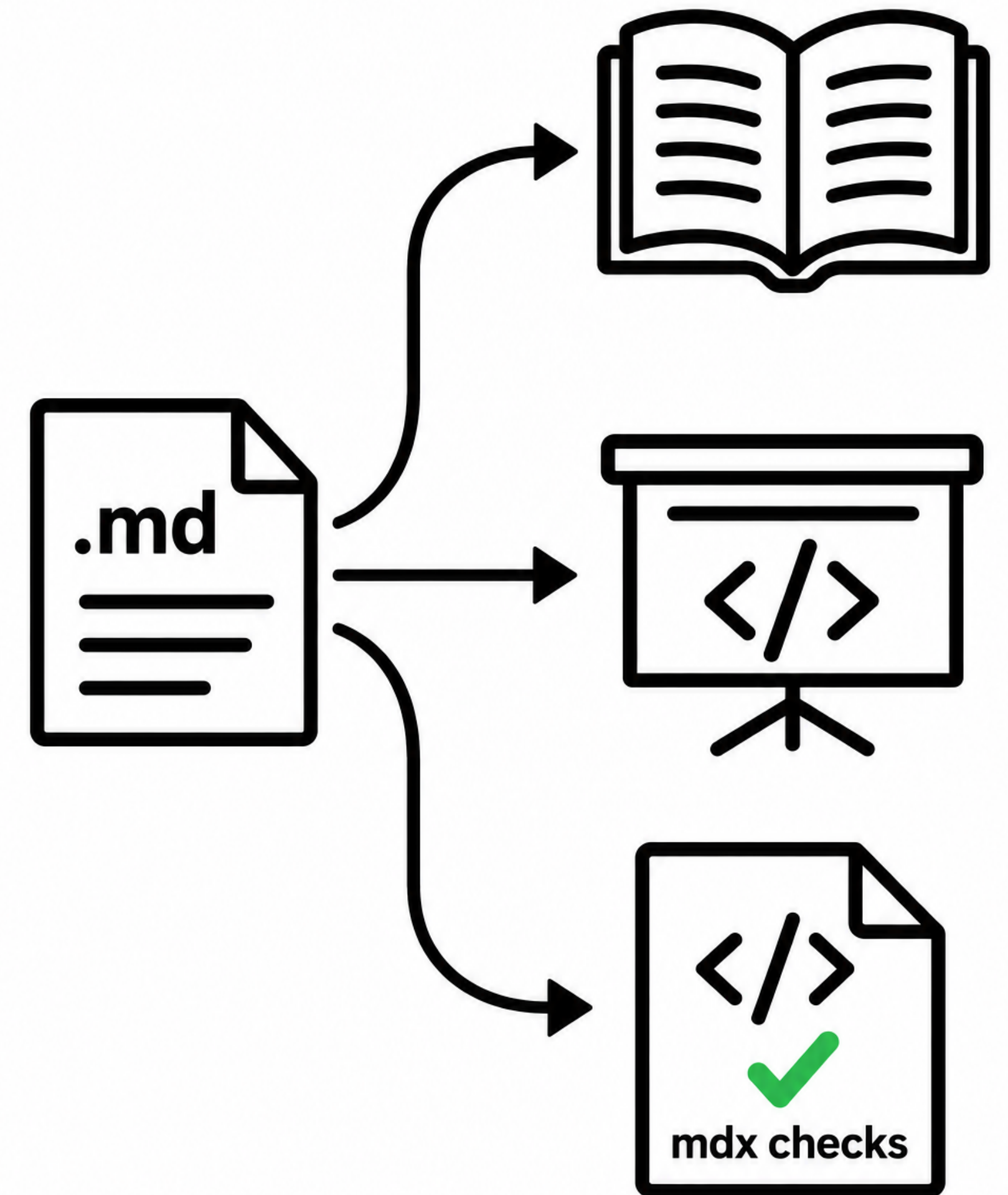
Authoring the modules

- **One markdown source for textbook and slides**
 - Helps keeps explanations and examples consistent
 - Readable Git diffs and merges
- **Supported tags**
 - slides — slide, subslide, fragment, layout — cols, col
 - quizzes — code, mcq, solution
 - vm-terminal



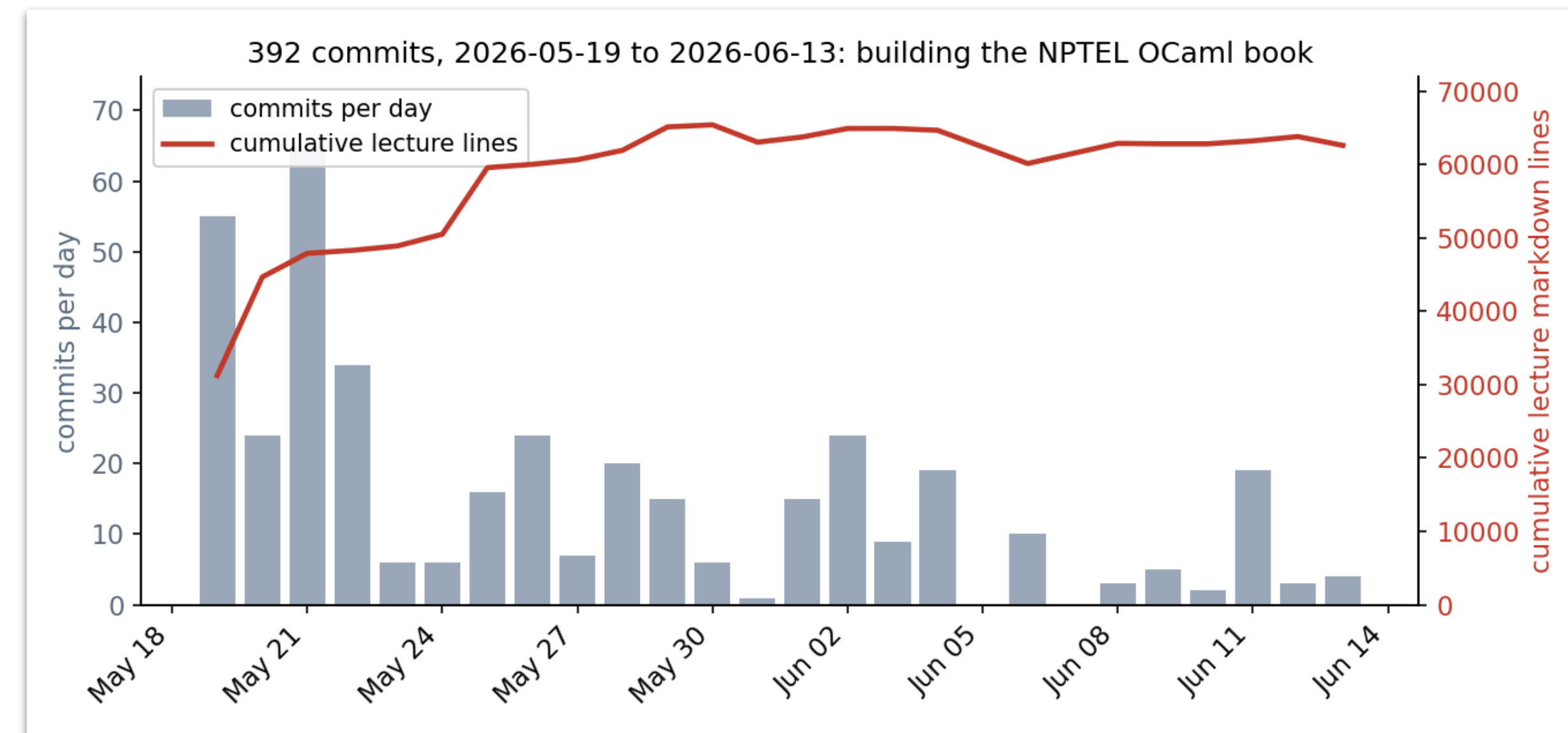
Authoring the modules

- **One markdown source for textbook and slides**
 - Helps keeps explanations and examples consistent
 - Readable Git diffs and merges
- **Supported tags**
 - slides — slide, subslide, fragment, layout — cols, col
 - quizzes — code, mcq, solution
 - vm-terminal
- ***Automated checks support review***
 - **ocaml-mdx** compiles and runs the code examples
 - Additional checks for slide overflow and broken cross-refs



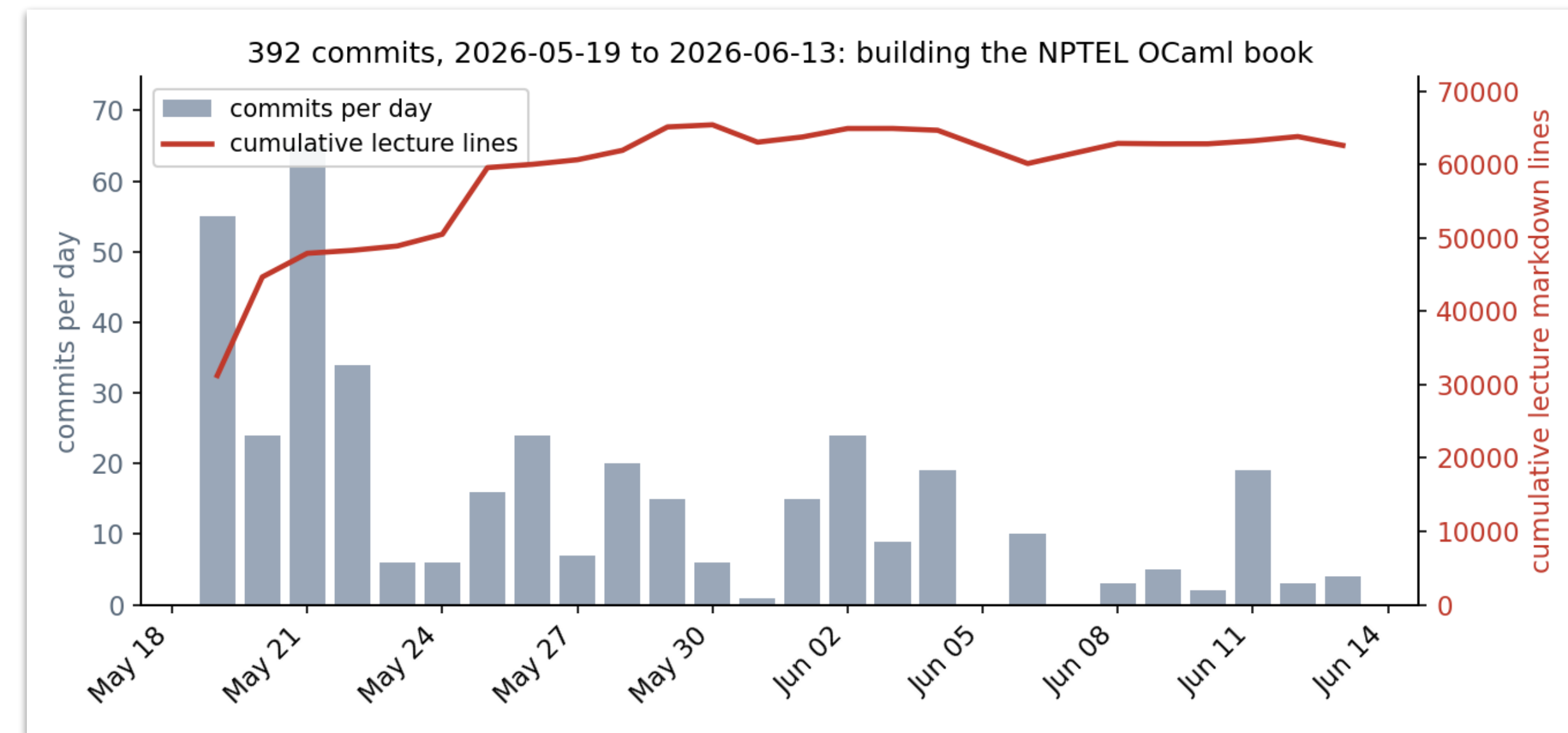
Agentic Programming

- Written in a month
 - Claude Opus 4.7+; 207M non-cached tokens
- Platform side almost entirely vibe coded.
 - ~9k LoC in OCaml, JavaScript, Shell, ...



Agentic Programming

- Written in a month
 - Claude Opus 4.7+; 207M non-cached tokens
- Platform side almost entirely vibe coded.
 - ~9k LoC in OCaml, JavaScript, Shell, ...
- Content is derived from IITM CS3100
 - *60k lines of markdown*
- Agent “learnt” the material
 - ffmpeg to scene (slide) detect
 - Local Whisper model for transcribing audio
 - A small script aligns slide & transcript



CS3100 Paradigms of Programming @ IITM ...
by KC Sivaramakrishnan
Playlist • Public • 47 videos • 20,129 views

Lectures from the CS3100 Paradigms of Programming (OCaml + Prolog) course taught at IIT Madras, India. ...more

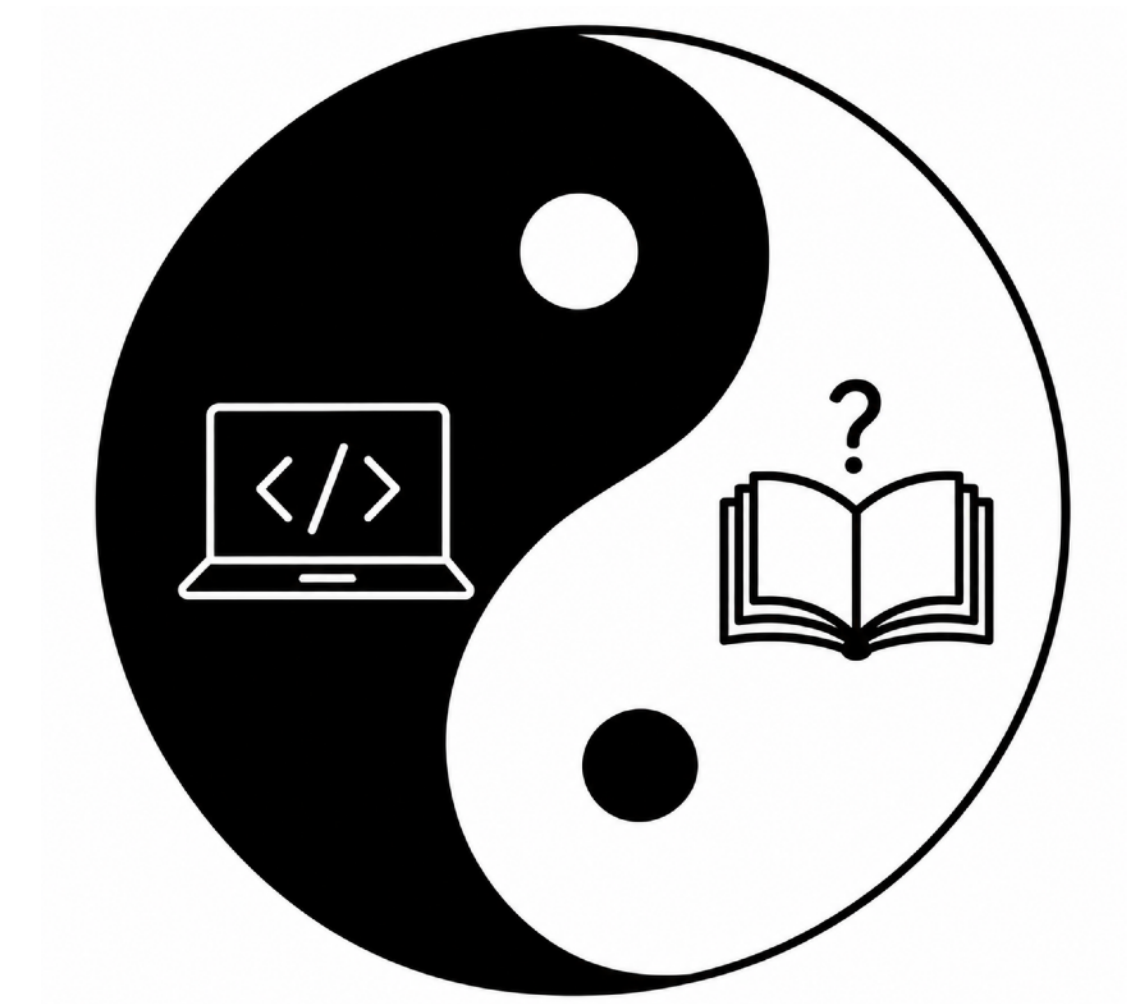
CS3100 POP - Lec 01 - Why PL?
KC Sivaramakrishnan • 8k views • 5 years ago • 51:02

CS3100 POP - Lec 02 - Expressions + Functions
KC Sivaramakrishnan • 2.2k views • 5 years ago • 53:21

CS3100 POP - Lec 03 - Setting up the notebooks
KC Sivaramakrishnan • 1.2k views • 5 years ago • 55:43

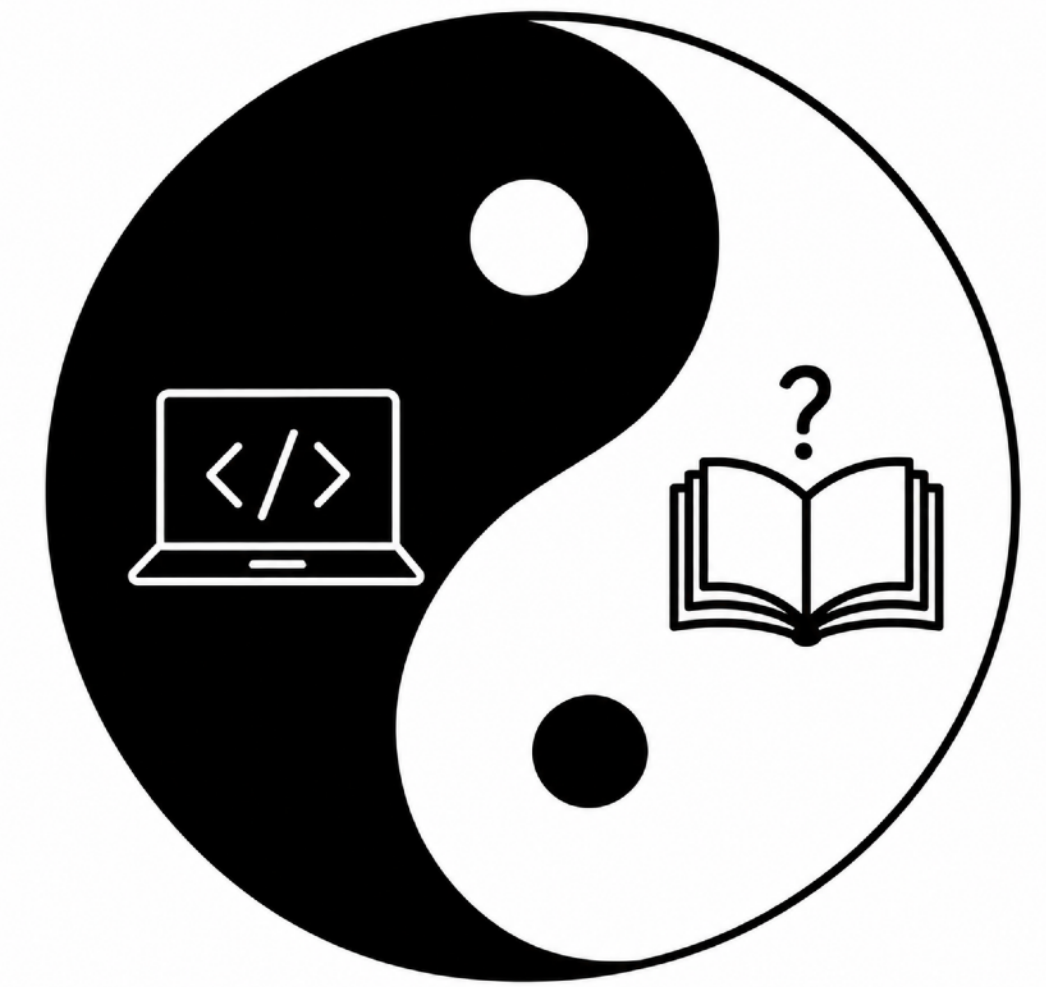
Reflections on Agentic Development

- Platform development was *mostly good*
 - Frontend work, quizzes and analytics, etc..
 - OxCaml JS bundle size reduction from 285MB to 4MB



Reflections on Agentic Development

- Platform development was *mostly good*
 - Frontend work, quizzes and analytics, etc..
 - OxCaml JS bundle size reduction from 285MB to 4MB
- Content writing was *mostly bad*
 - Assumed too much about what the learner already knew
 - Used concepts before introduction; presented ideas without motivation
 - Great oracle but bad teacher
 - Curse of Knowledge — an expert assumes others have their specialised knowledge
 - Pedagogical Content Knowledge — knowing how to teach is separate from knowing



A Skill for writing chapters

- **No forward concepts**
 - Use only concepts that are currently in reader's toolbox
- **No jumps**
 - Motivate each idea, introduce using small examples
- **Slides carry the content**
 - Viewers don't read chapters first $\implies$ slides alone must carry the narrative
- **Fresh activities**
 - Programming quizzes shouldn't repeat existing content
- **Reviews in fresh agent**
 - Prompt to assume basics and ask the agent to read preceding content

Summary

Summary

- AI is fast changing the nature of education
 - But also enables broader access to specialised knowledge
 - Gaps need educators to innovate

Summary

- AI is fast changing the nature of education
 - But also enables broader access to specialised knowledge
 - Gaps need educators to innovate
- The course seems to be going well
 - The course content is distributed under CC-BY-NC-SA



**Functional
Programming
with OCaml**
online book

Summary

- AI is fast changing the nature of education
 - But also enables broader access to specialised knowledge
 - Gaps need educators to innovate
- The course seems to be going well
 - The course content is distributed under CC-BY-NC-SA



**Functional
Programming
with OCaml**
online book



FP Launchpad

Post-bacc fellowship — Cohort 2

Deadline: Dec 10th 2026