Biome

TOOLCHAIN OF THE WEB

Victorien Elvinger

@Conaclos

Biome lead maintainer

What is Biome?

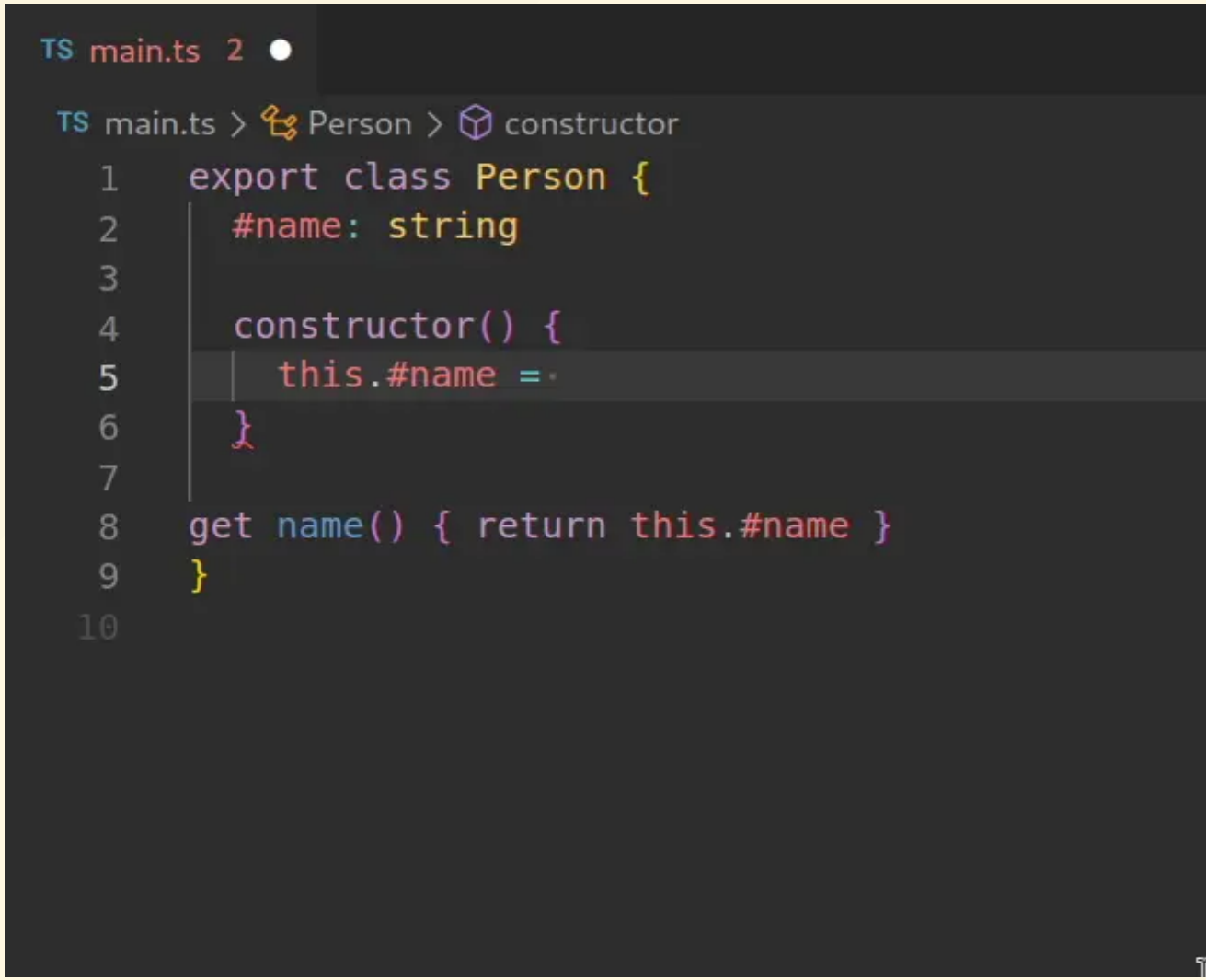
A code linter

- JavaScript, **TypeScript, JSX**, TSX
  - no extra dependencies
- **Helpful diagnostics**
- **200 lint rules**
  - some unique to Biome
  - ESLint, ESLint plugins
  - 🚧 Tailwind class sorting

```
❯ npx @biomejs/biome lint main.ts

main.ts:4:18 lint/noAccumulatingSpread

  ✖ Avoid the use of spread ... syntax
    on accumulators.

    3 | array.reduce(
  > 4 |   (acc, v) => [...acc, v],
    |                 ^^^^^^
    5 |   []
    6 | )

  i Spread syntax should be avoided on
    accumulators because it causes
    a time complexity of `O(n^2)`.

  i Consider methods such as
    .splice or .push instead.
```

A code formatter

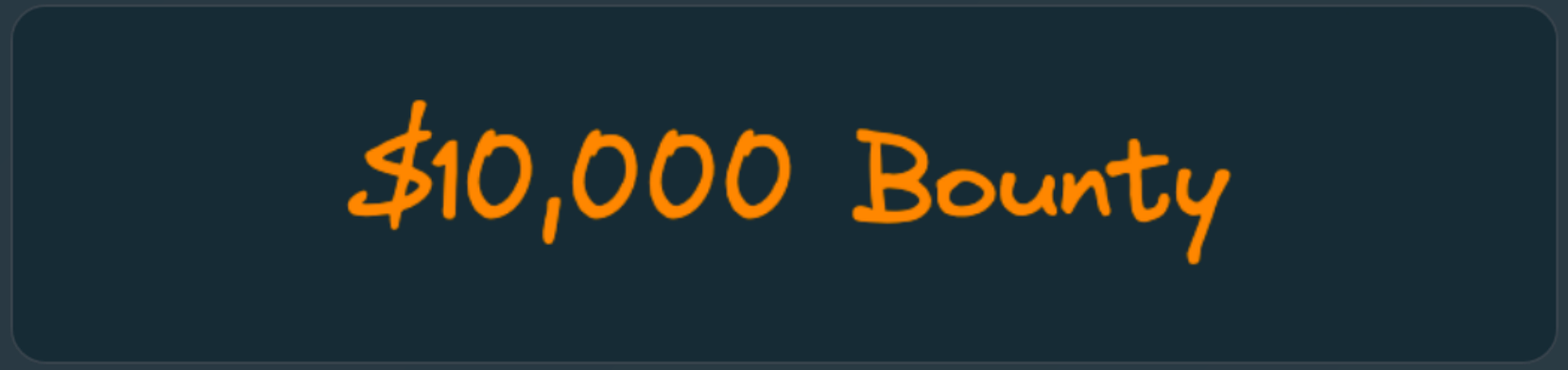
- JavaScript, TypeScript, JSX, TSX
- JSON, JSONC
- 🚧 CSS
- **format invalid code**



vjeux
@Vjeux

There's lot of excitement around faster pretty printers using Rust. The main issue is that none of them match the long tail of formatting logic of prettier.

I'm putting up a $10k bounty for any project written in Rust that passes > 95% of the prettier JavaScript tests.



10:50 PM · Nov 9, 2023

1K Reply

Write a pretty printer in Rust.
Win $25,000

Grand Prize
$22,500

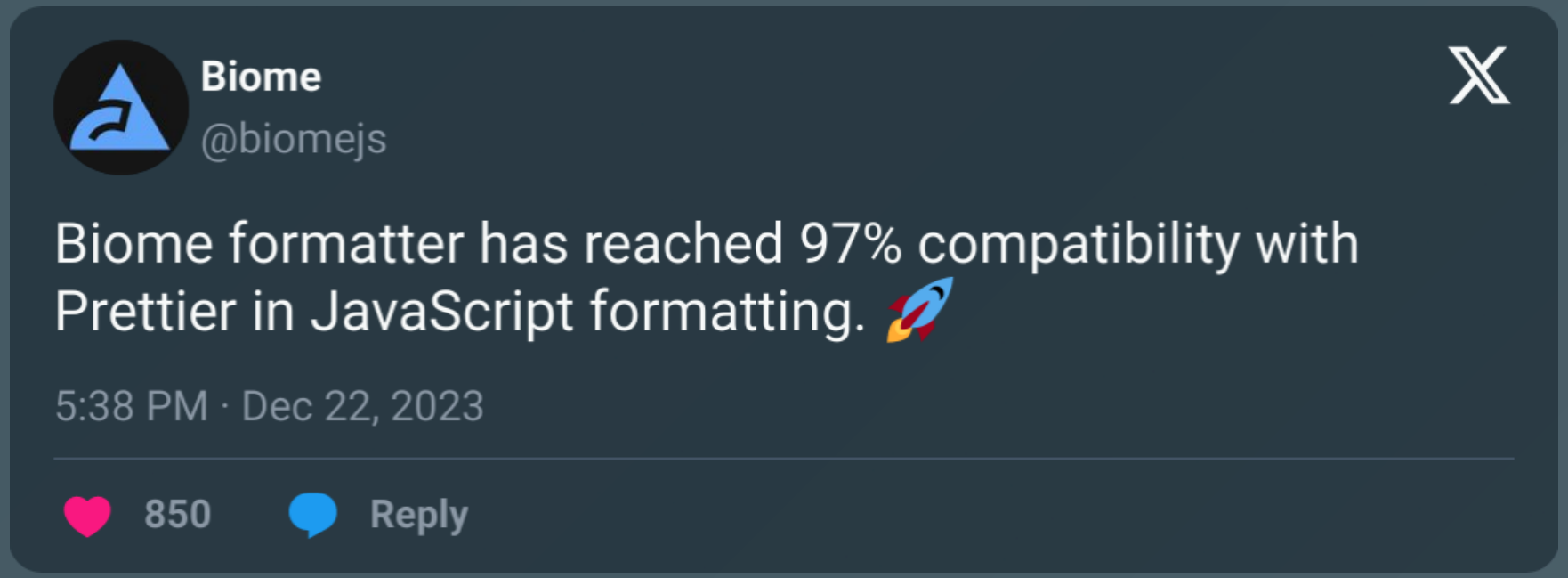
Pass > 95% of the prettier JavaScript tests

READ THE ANNOUNCEMENT →

WASIX Prize
$2,500

Compile to WASIX and publish (via CI) to Wasmer

READ THE ANNOUNCEMENT →


Biome
@biomejs
Biome formatter has reached 97% compatibility with Prettier in JavaScript formatting. 🚀
5:38 PM · Dec 22, 2023
850
Reply

Is Biome fast?

fforres
@fforres

Holy shit.

Just did a quick parsing test of @biomejs in one of our @OpenAI codebases

3️⃣ eslint + prettier = 58.81 s
2️⃣ eslint (w/ cache) + prettier = 12.82 s
1️⃣🤯 biomeJS check (first run) = 1.78 s

(Tried tweaking the config, and was able to get a 95% config parity in ~20m)🤯

9:45 PM · Dec 11, 2023

550 Reply

A community

- ⬇️ 170k weekly downloads
- ⭐ 8.4k GitHub Stars
- 🐦 4.6k followers
- 💬 1.2k Discord members

astro

Rspack

時雨堂

Shiguredo

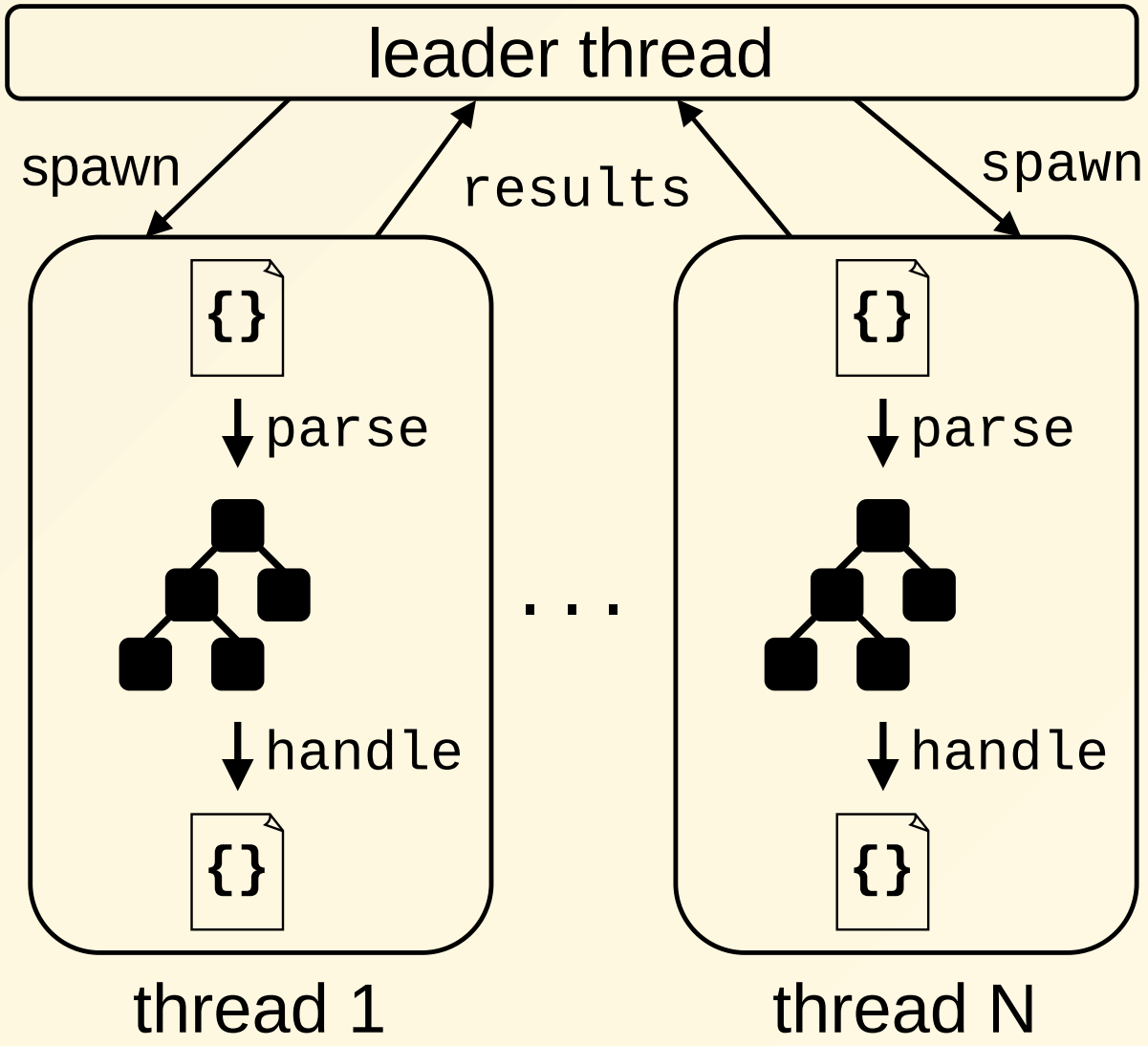
SENTRY

Vercel

How Biome works?

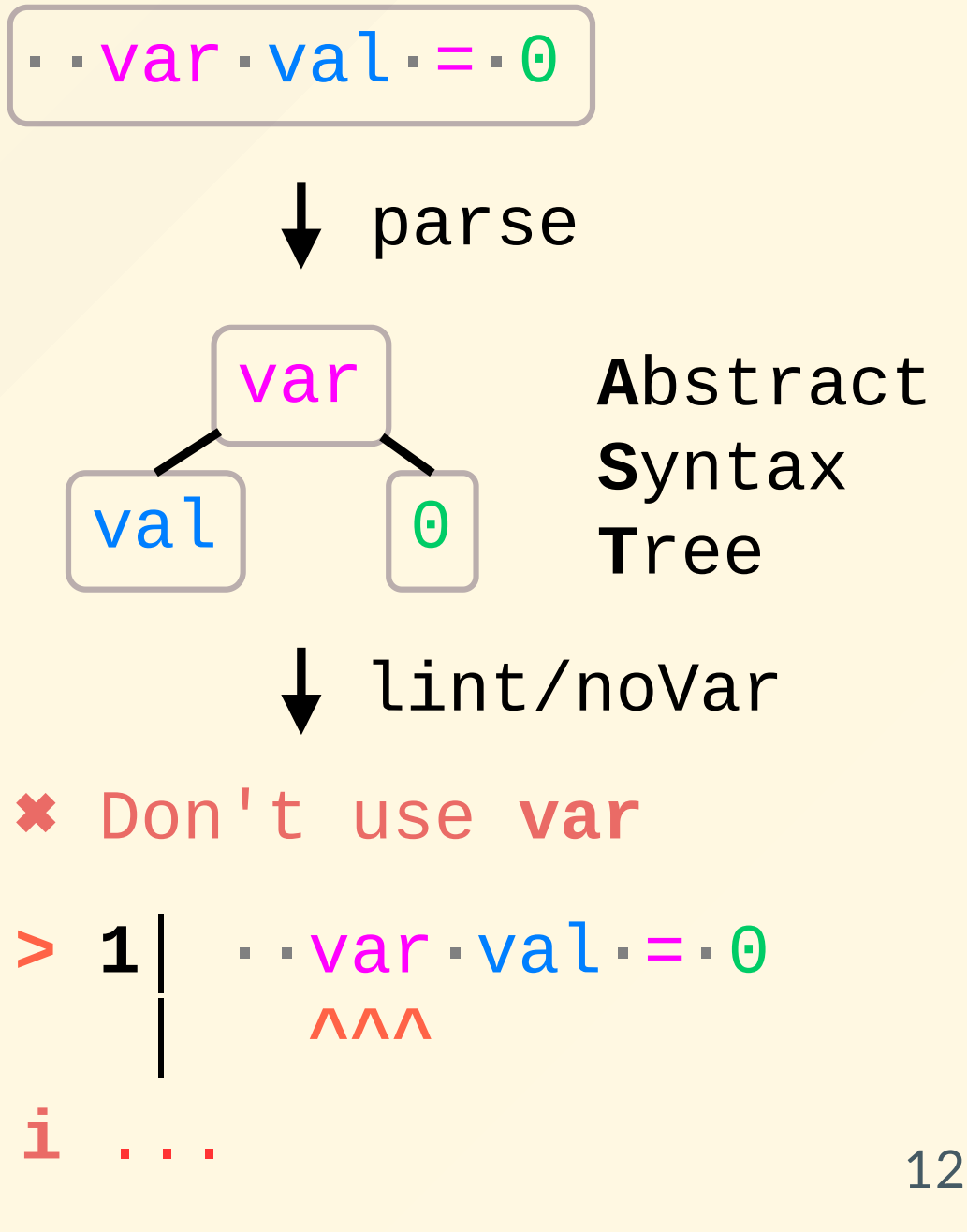
Architecture

- **leader-follower**
- the leader thread
  - spawn a thread per file
  - collect results
- a follower thread
  - parse the given file
  - handle (format, lint)



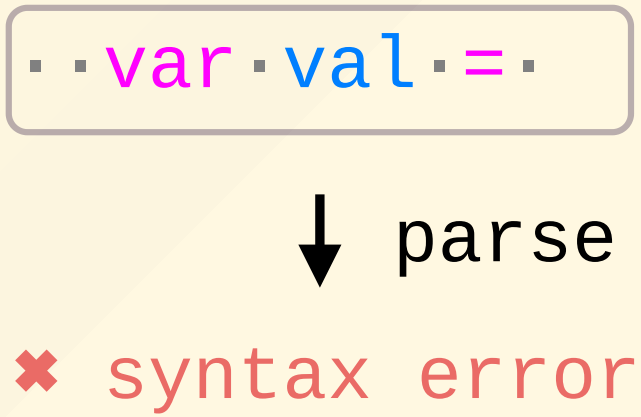
Regular parser

1. parse to an Abstract Syntax Tree
2. handle (format, lint)



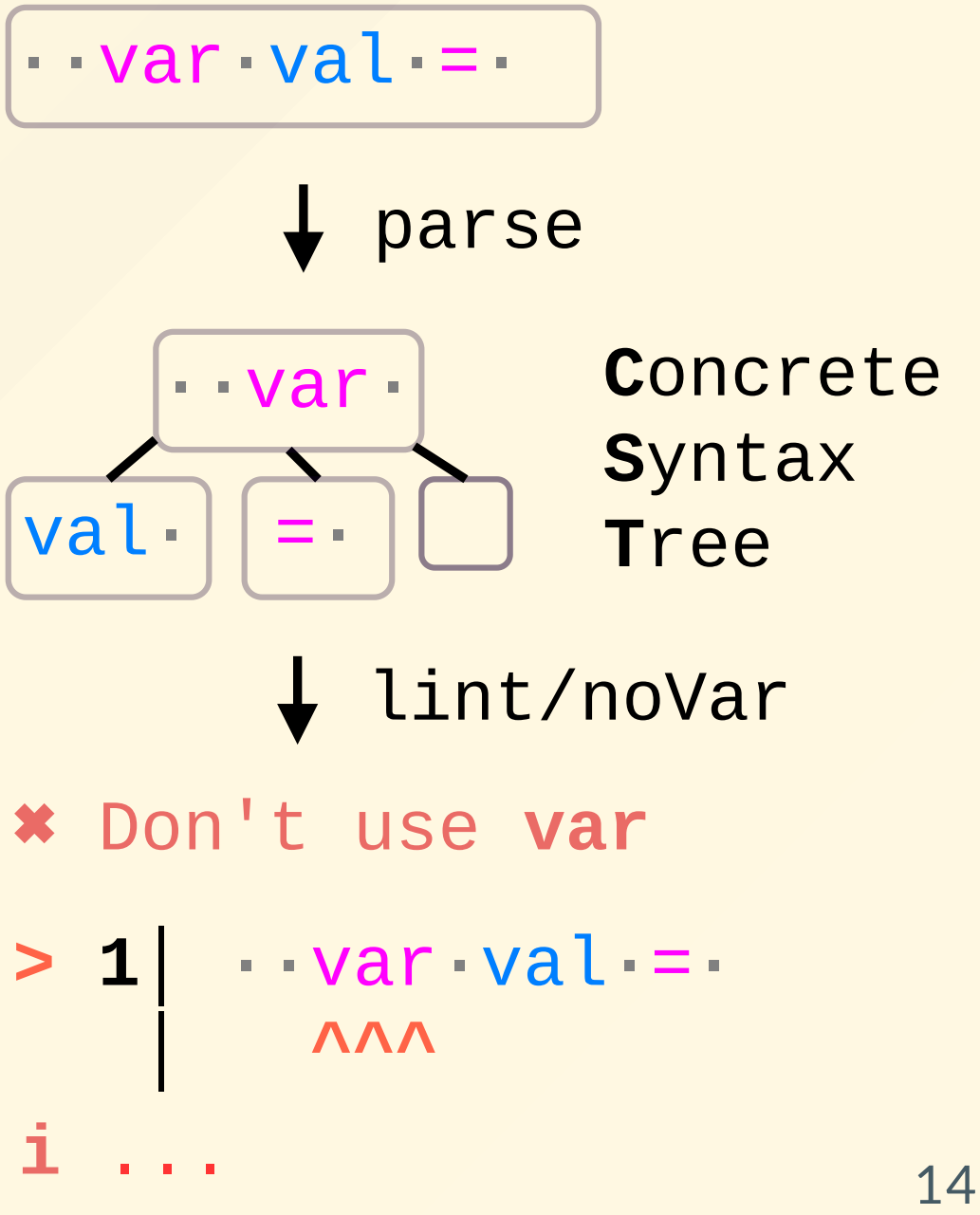
Regular parser

- doesn't handle invalid code
  - **emit syntax error**



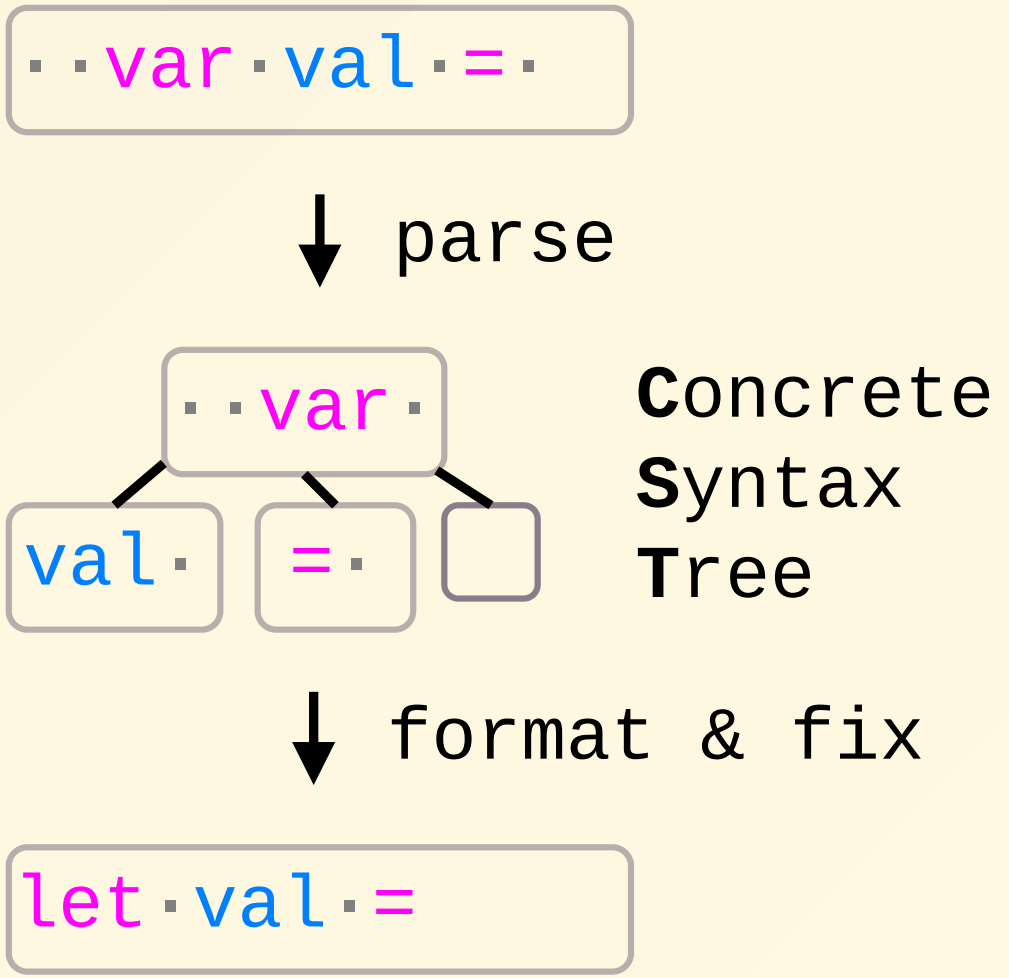
Biome parser

- **accept invalid code**
  - bogus tree nodes
  - holes in the tree
- **lossless** parsing using CST
  - preserve whitespace



Biome parser

- **accept invalid code**
  - bogus tree nodes
  - holes in the tree
- **lossless** parsing using CST
  - preserve whitespace



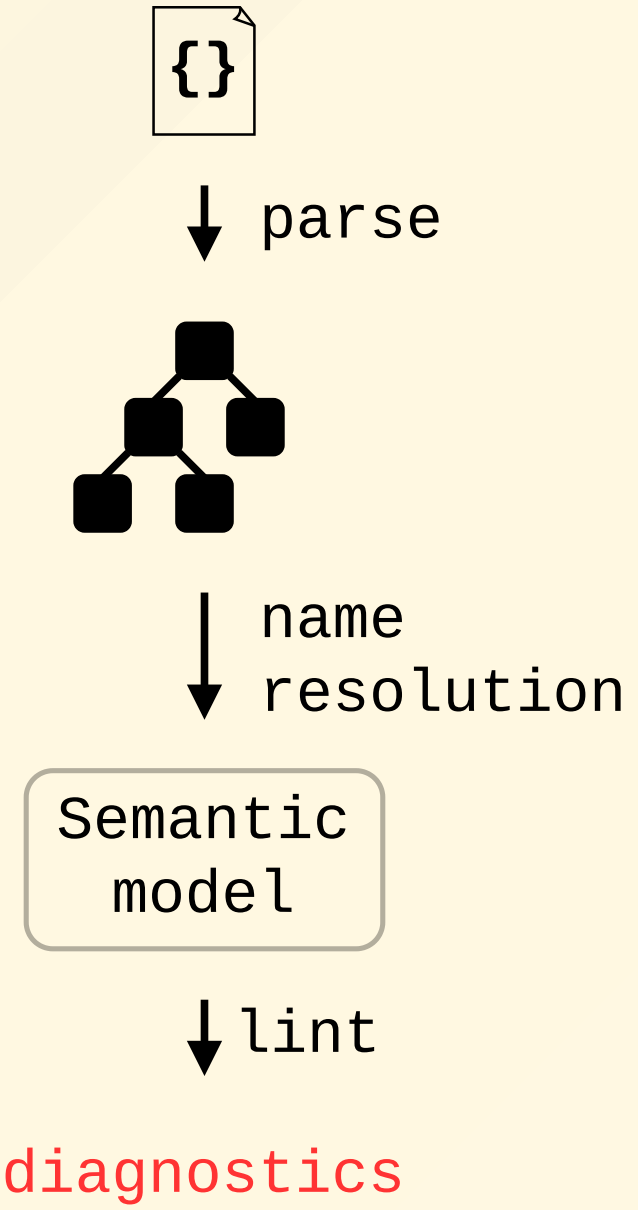
Lint rules

- many rules query the tree
  - noVar
  - noDoubleEquals
  - noAccumulatingSpread
- others need more complex data
  - **noUnusedVariables**
  - noUnusedImports
  - useImportType
  - useExportType

```
function Person (name ) {
  if (name  == "") {
    let name
    name  = "anonymous"
  }
  return { name  }
}

export { Person  }
```

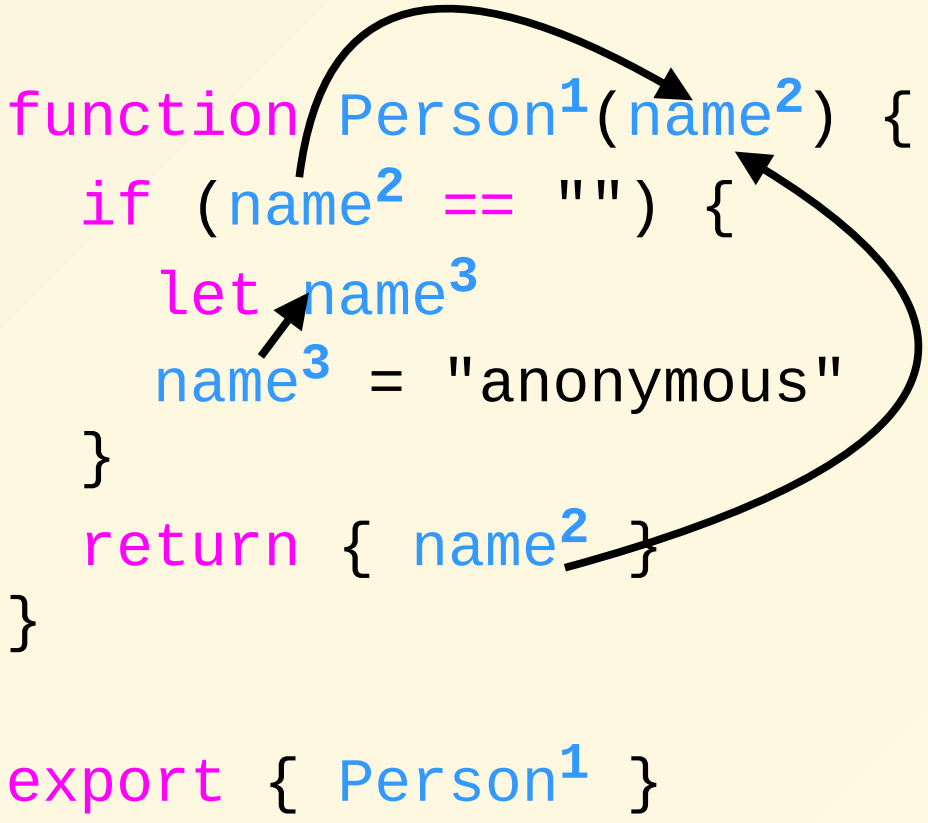
Semantic model

- find references of a declaration
  - write refrences
  - read references



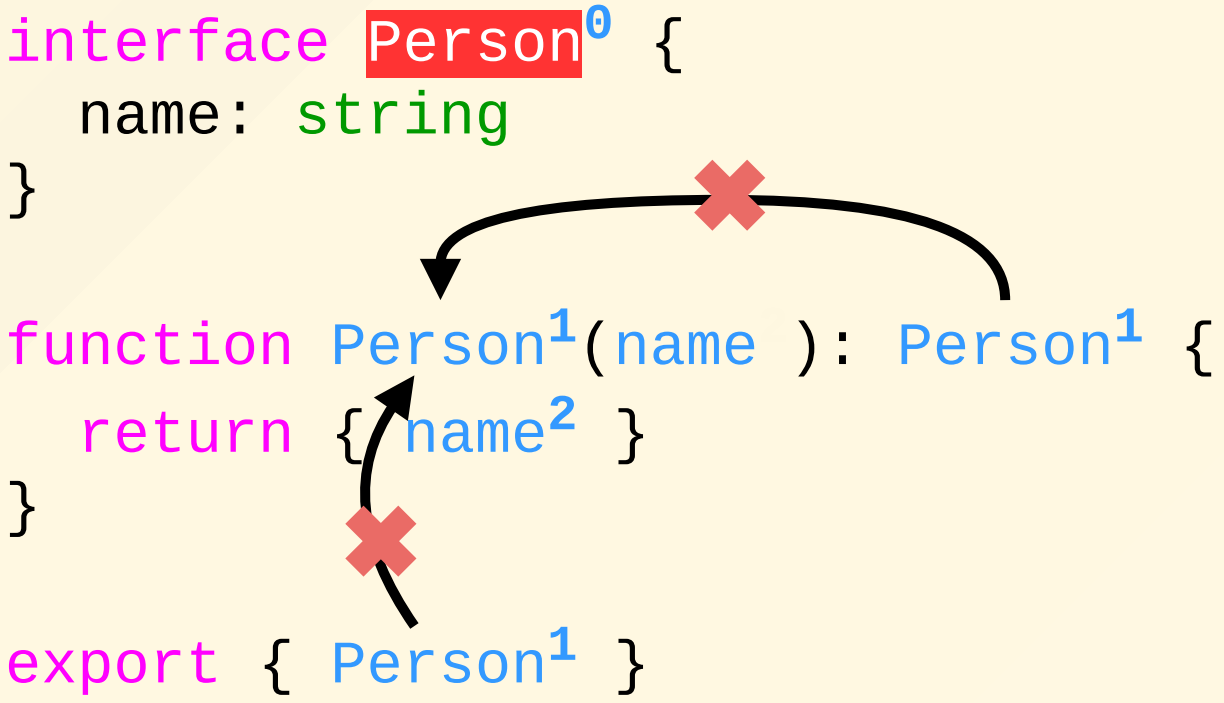
Name resolver v1

- bind declarations to references
  - unique id for each declaration
  - a reference refers to a single declaration
- take scopes into account
  - variable shadowing



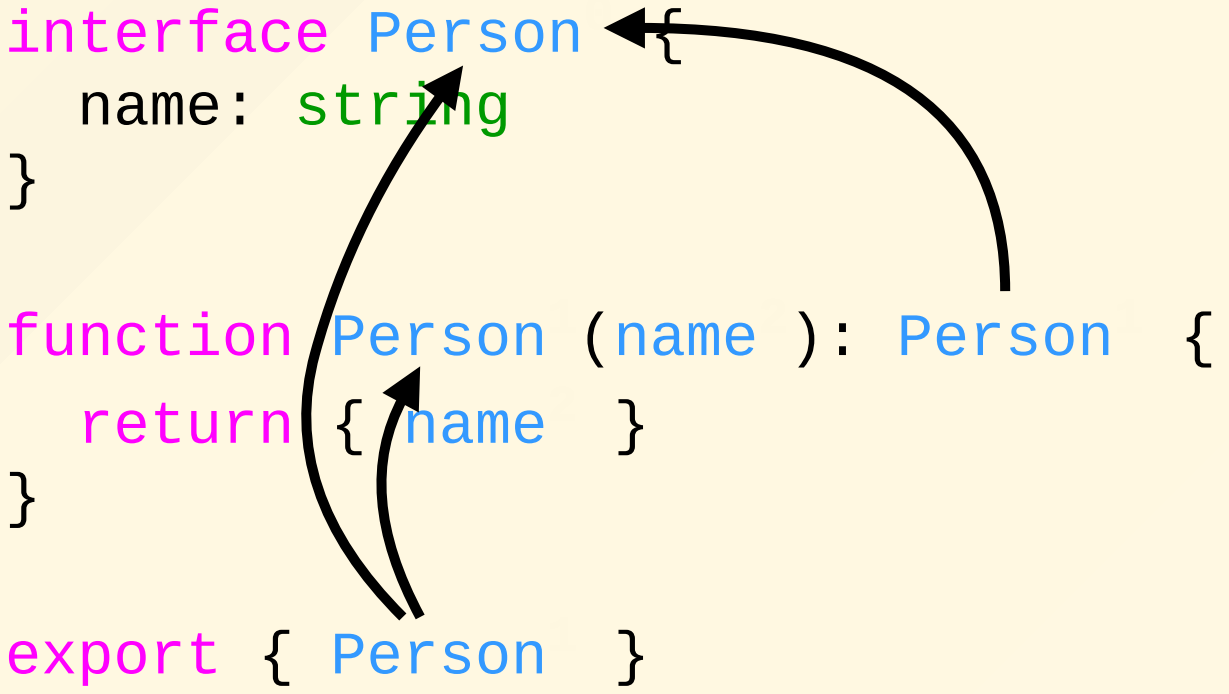
TypeScript?

- bind declarations to references
  - unique id for each declaration
  - a reference refers to a single declaration
- take scopes into account
  - variable shadowing



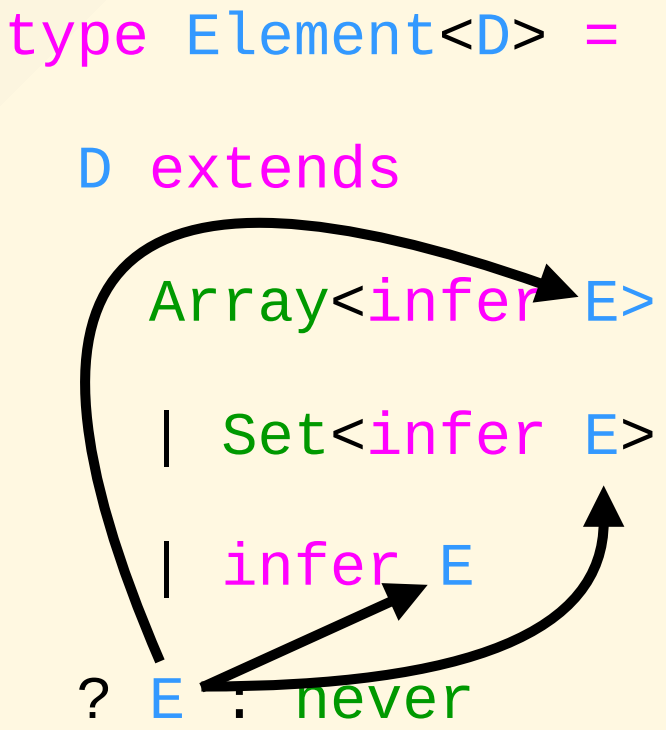
TypeScript 🤯

- **type & variable** with **same name**
- **a reference** refers to **multiple declarations**



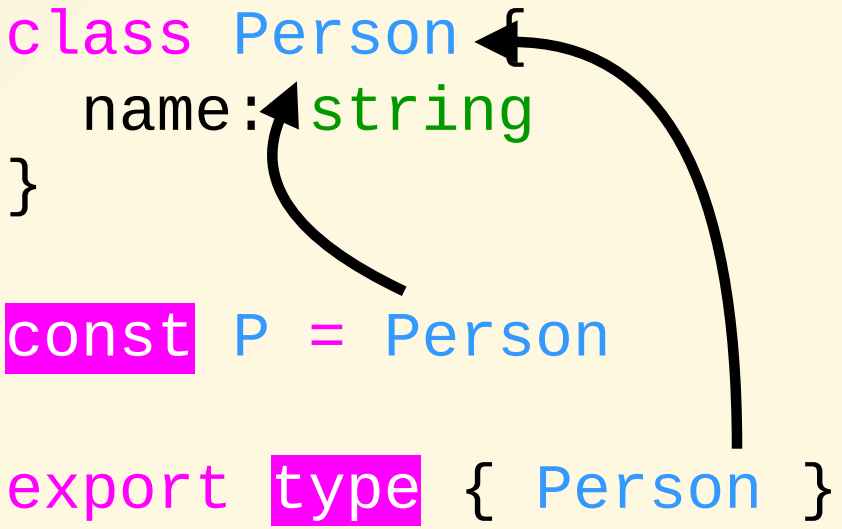
TypeScript 🤯🤯

- type & variable with same name
- **a reference** refers to **multiple declarations**



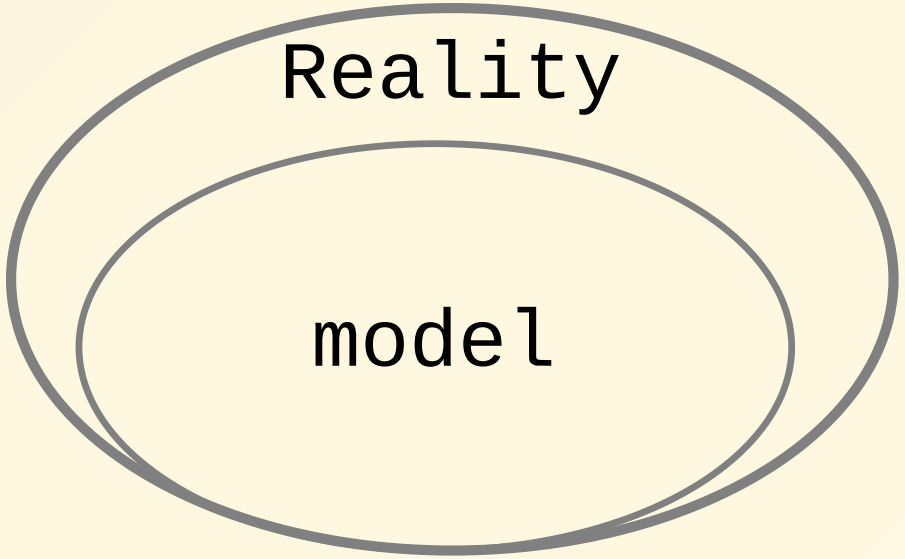
TypeScript 🤯

- type & variable with same name
- a reference refers multiple declarations
- **partially referenced declarations**
  - type / variable duality



Simplification

- a reference refers to a **single declaration**
  - handle differently edge cases (*export*, *infer*)
- type and value with same names
- type / variable duality



Name resolver v2

- A declaration is either
  - a type
  - a variable
  - both
- A reference refers either
  - a type
  - a varaible

```
interface Person[t0] {
  name: string
}

function Person[v1](name[v2]): Person[t0] {
  return { name[v2] }
}

export type { Person[t0] }
```

Name resolver v2

- A declaration is either
  - a type
  - a variable
  - both
- A reference refers either
  - a type
  - a varaible
- type/value duality not exposed

```
interface Person⁰ {
  name: string
}

function Person¹(name²): Person⁰ {
  return { name² }
}

export type { Person⁰ }
```

Conclusion

- Biome is both a **formatter** and a **linter**
  - and more: JavaScript **import sorting**
- Biome is **fast**
- Biome is **editor-ready**
  - error-resilient parsers
  - Concrete Syntax Tree
- Biome **supports TypeScript**
  - type-aware semantic model

2024 and beyond

- extend to **more languages**
  - CSS, HTML, Markdown
  - Vue, Angular, Svelte, Astro
- **improve linter capabilities**
  - multi-file analysis
  - simplified type system
- 🧱 **plugins**

Want to help?

- 🚀 try Biome
  - 🐛 report issues
  - 💬 feedbacks
- contribute to Biome
  - GitHub good first issues
  - 📽️ How to create a lint rule in Biome (youtube.com/@Biomejs)
- 💸 sponsor us!
  - Biome Open Collective

biomejs.dev

format code

npx **@biomejs/biome format --write** src

lint code, apply safe fixes

npx **@biomejs/biome lint --apply** src

all at once

npx **@biomejs/biome check --apply** src

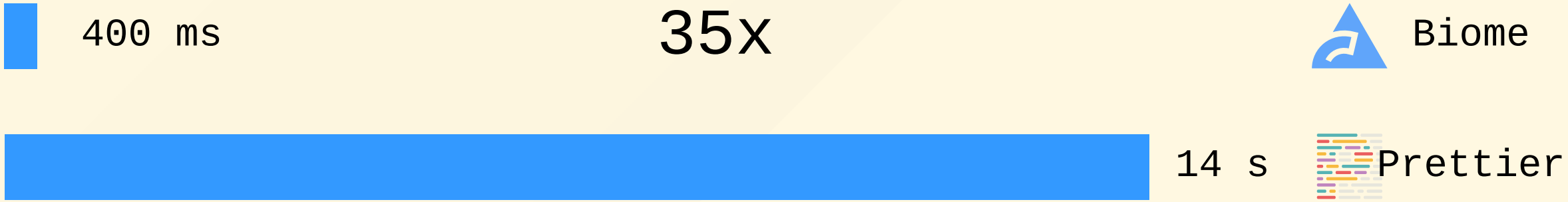
Backup slides

A toolchain

- 🧰 toolchain for **web dev**
  - code formatter
  - code linter
- written in **Rust** 🦀
- supports main web language
  - **JavaScript, TypeScript, JSX, TSX**
  - JSON, JSONC
  - CSS ⏳
- **community successor** of Rome Tools

A fast formatter

- scales with available threads



Formatting **170k lines** of code in **2.1k files** with an **Intel Core i7 1270P**

A governance

- leads (2) 🔑 owners
  - 🛡️ access to sensible data
  - ⚔️ act as tiebreakers
- core contributors (5)
  - 👍 project directions
- maintainers (5)
  - 👍 project decisions
  - ⬆️ write access to the repo