

Servo

Developing an independent, light-weight, modular and parallel web-engine
独立的，轻量级，模块化与并行处理架构的Web引擎开发

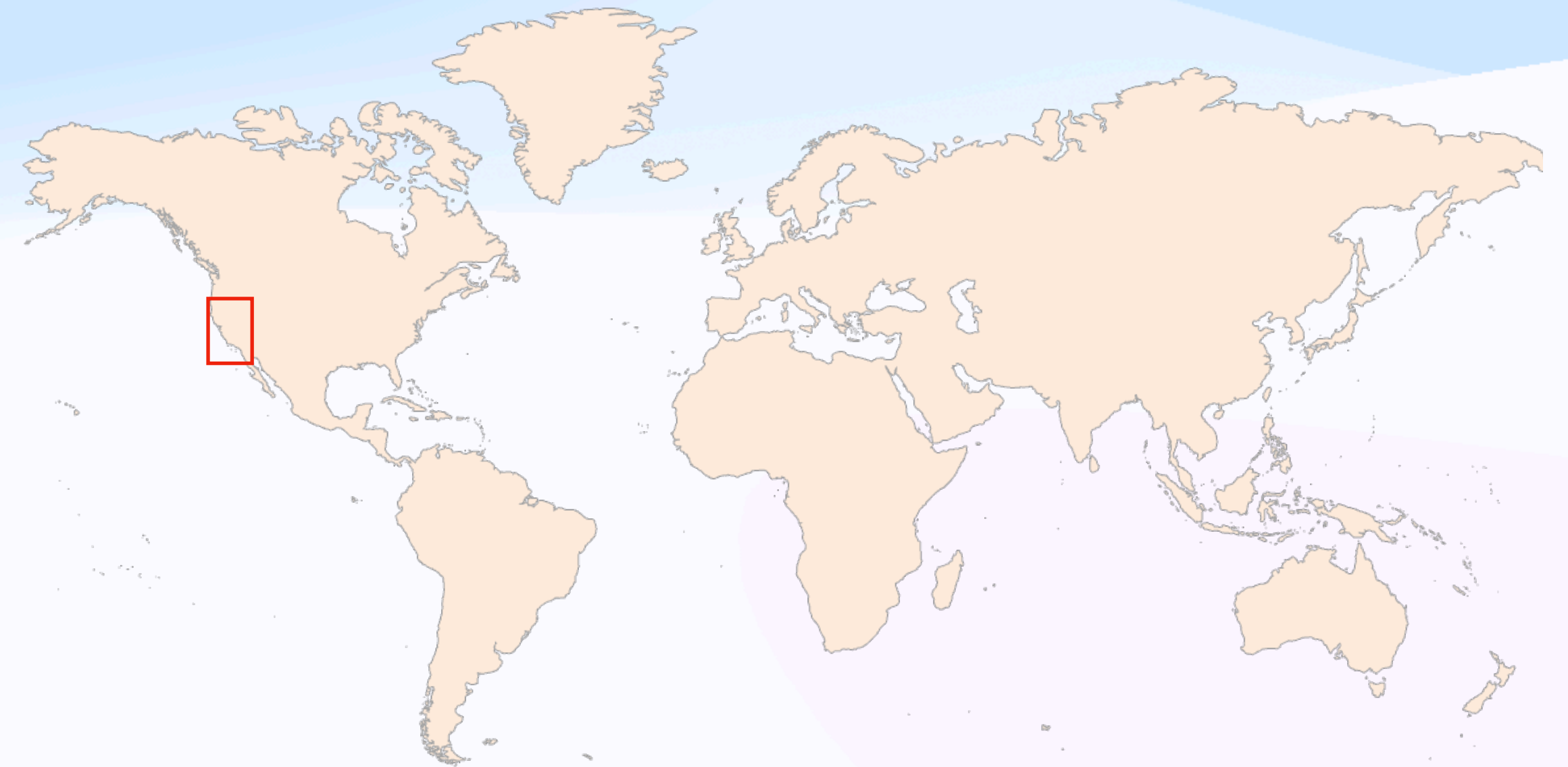
Jonathan Schwender, Senior Engineer @ 华为希爾伯特研究所（德国），2025-09-05

Agenda

- What is servo?
- Goals and status
- Why do we need a new Web-engine?

Browser Engines

- Production ready Engines:
 - WebKit (Apple)
 - Blink (Google)
 - Gecko (Mozilla)
- Upcoming:
 - Servo (Independent, Linux Foundation EU)
 - LibWeb (Ladybird foundation US)



Servo

A quick overview

- **Embeddable:** Provides a Webview API for applications to embed content
- **Memory safe:** Written in Rust to reduce chance of vulnerabilities
- **Parallel:** Use concurrency for faster and more energy-efficient usage of multi-core devices
- **Cross-platform:** Windows, Linux, MacOS, Android and OpenHarmony
- **Independent:** Open governance under the Linux Foundation EU

Servo

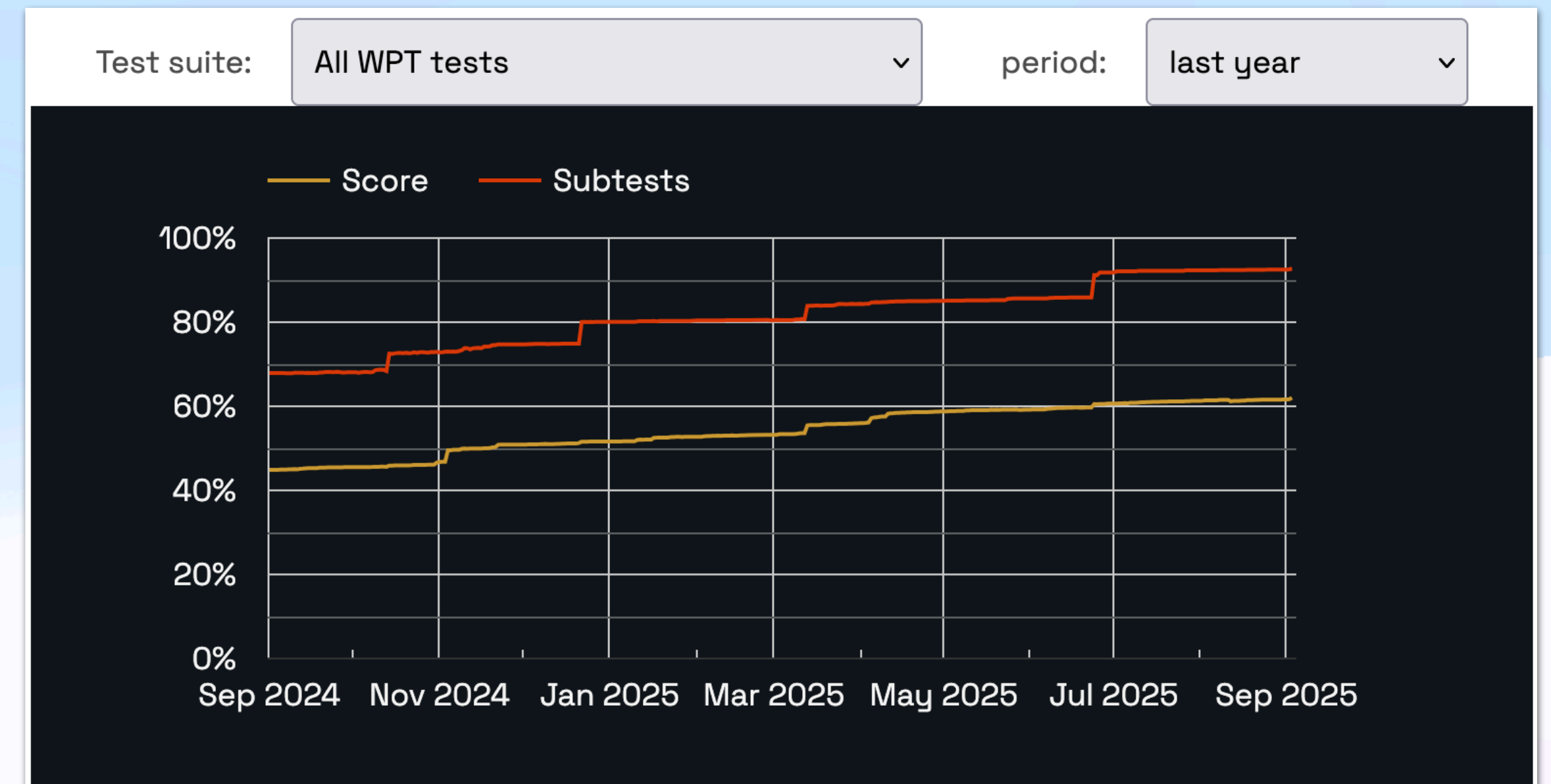
A brief overview

- Rust and servo were co-developed at Mozilla, due to frustration with C++
- 2012-2020: Mozilla: Servo as a parallel web-engine for browser research
 - Modular design, some components are part of Firefox
- Donated to Linux Foundation and 2023 transferred to Linux Foundation EU
 - Has gathered an active community

Status: Web compatibility

WPT: Web platform tests

- WPT is an extensive test suite to test browsers
- 92% subtest pass rate
 - 68% 1 year ago
 - This metric is deceptive
- 62% weighted score
 - 45% 1 year ago
 - Chromium: 89%



Scores are computed as follows: a passing test with no subtests gets a score of 1 while a test with subtests gets a score between 0 and 1 representing the fraction of passing subtests within that test.

62%?

That sounds pretty bad ...

Servo as a Browser?

- <https://servo.org/download/>
- Download prebuilt binaries and try it out yourself
- Live Demo



What about other Scenarios?

An embedded Web-engine for applications

Embedding servo is a first-class use case

- Content is known by the application author in advance
 - Only uses a **subset** of web standards
- Can we tailor the engine towards the applications needs?
 - **Modular** design
 - **Strip** unneeded features
 - **Lazy** load rarely used features
 - **Flexible** Isolation: Let the application decide

Seamless AI-Agent integration

- Servo aims to provide a seamless embedding experience
- Agents are just another embedder!
 - Navigate pages, investigate the DOM, ...
- AI, Webdriver, Devtools and Accessibility APIs could all be based on the rich embedder API.
- Flip it around: Embedder could use Webdriver to abstract over the engine powering a web view across different platforms!

Why Servo

Technical Goals for Servo

- Rust: Write concurrent code with confidence
- Modular: Use great crates from the Rust ecosystem
- Configurable: Adapt servo to the applications needs
- Light-Weight: Reduce Memory usage and binary size

**What about non-technical
reasons?**

Independant

An independent web engine under the Linux Foundation Europe

Independant

A neutral and independent web engine

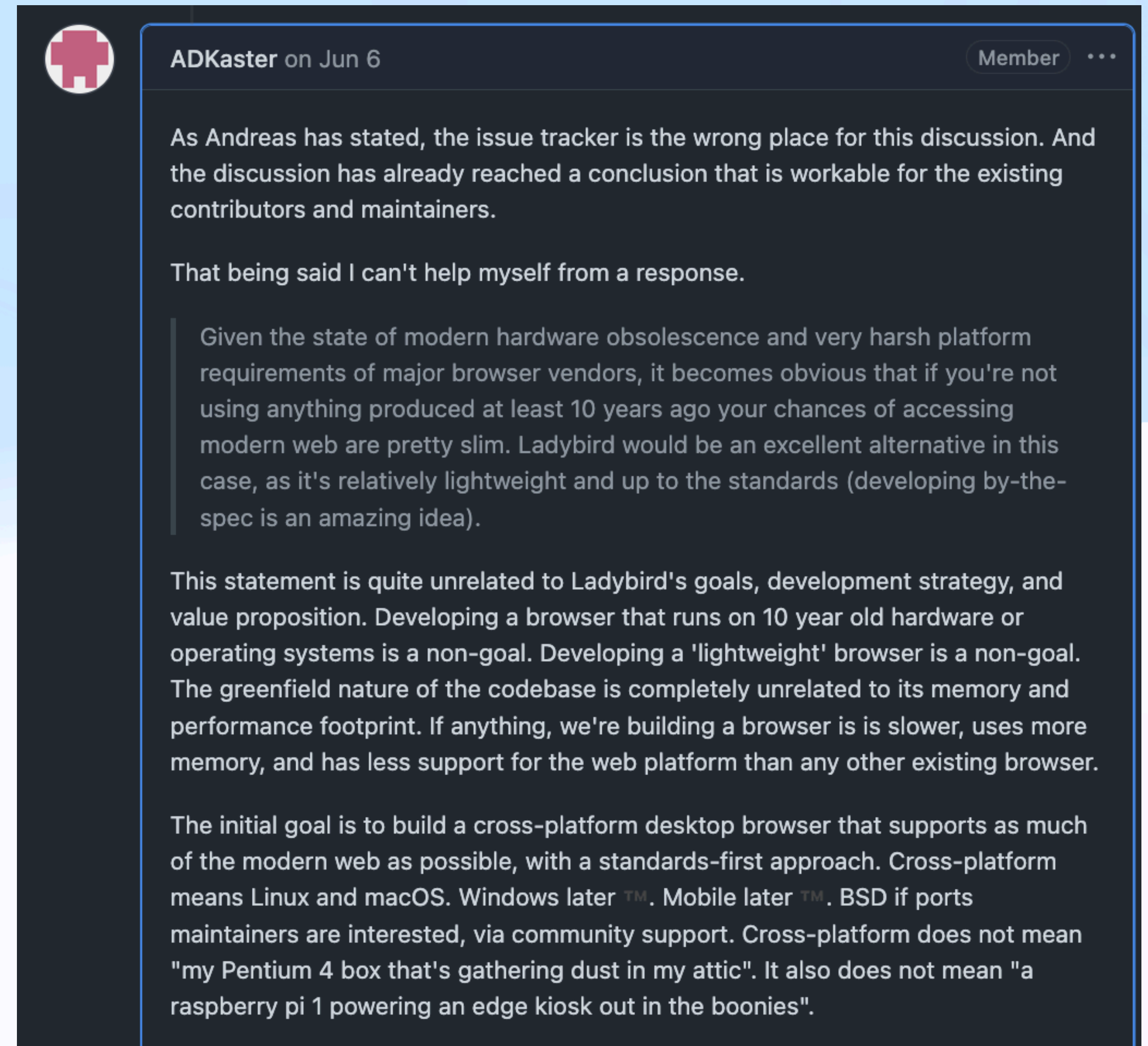
- Open Governance (Linux Foundation EU)
- Governed by a neutral TSC (Technical steering committee)
 - Influence of a single organization is limited to **1/3** of the votes
 - Public meetings and decision process
- Worldwide contributors (mainly from EU and Asia)

Servo Contributors

- Contributors from all over the world
- Funded contributors from companies: Huawei, Igalia, ...
- Contributors funded by public grants (e.g. Next Generation Internet Zero Core)
- Volunteer contributors
- > 50% of the Servo TSC are unaffiliated contributors (5x Igalia, 1x Huawei, 11x unaffiliated)

What about Ladybird?

- It's a great project! But ...
- Browser-first strategy
 - No Mobile support
- Light-weight and performance are not immediate goals
- Swift
 - Hard to port to other platforms



ADKaster on Jun 6 Member

As Andreas has stated, the issue tracker is the wrong place for this discussion. And the discussion has already reached a conclusion that is workable for the existing contributors and maintainers.

That being said I can't help myself from a response.

Given the state of modern hardware obsolescence and very harsh platform requirements of major browser vendors, it becomes obvious that if you're not using anything produced at least 10 years ago your chances of accessing modern web are pretty slim. Ladybird would be an excellent alternative in this case, as it's relatively lightweight and up to the standards (developing by-the-spec is an amazing idea).

This statement is quite unrelated to Ladybird's goals, development strategy, and value proposition. Developing a browser that runs on 10 year old hardware or operating systems is a non-goal. Developing a 'lightweight' browser is a non-goal. The greenfield nature of the codebase is completely unrelated to its memory and performance footprint. If anything, we're building a browser is is slower, uses more memory, and has less support for the web platform than any other existing browser.

The initial goal is to build a cross-platform desktop browser that supports as much of the modern web as possible, with a standards-first approach. Cross-platform means Linux and macOS. Windows later [™]. Mobile later [™]. BSD if ports maintainers are interested, via community support. Cross-platform does not mean "my Pentium 4 box that's gathering dust in my attic". It also does not mean "a raspberry pi 1 powering an edge kiosk out in the boonies".

Reach out to us

- Discuss your use case of an embedded web engine with us!
 - Tell us what you care about!
- Future Web Forum @ GOSIM 杭州 09月13 上午
- 开源鸿蒙技术大会 Web 分论坛 @ 长沙 09月27
- Join the development on Github: <https://github.com/servo/servo>
- Discuss on Zulip: <https://servo.zulipchat.com/>



Thank you