INFINI Labs

极限科技·让搜索更简单

2025

Coco AI 技术选型——Tauri V2 的分析

主讲人：Rain9

时间：2025.4

Catalogue

目录

1. Coco AI App 项目背景
2. 市面上桌面端框架对比
3. 选择 Tauri V2 作为开发框架
4. Tauri V2 开发中遇到的问题

PART.01

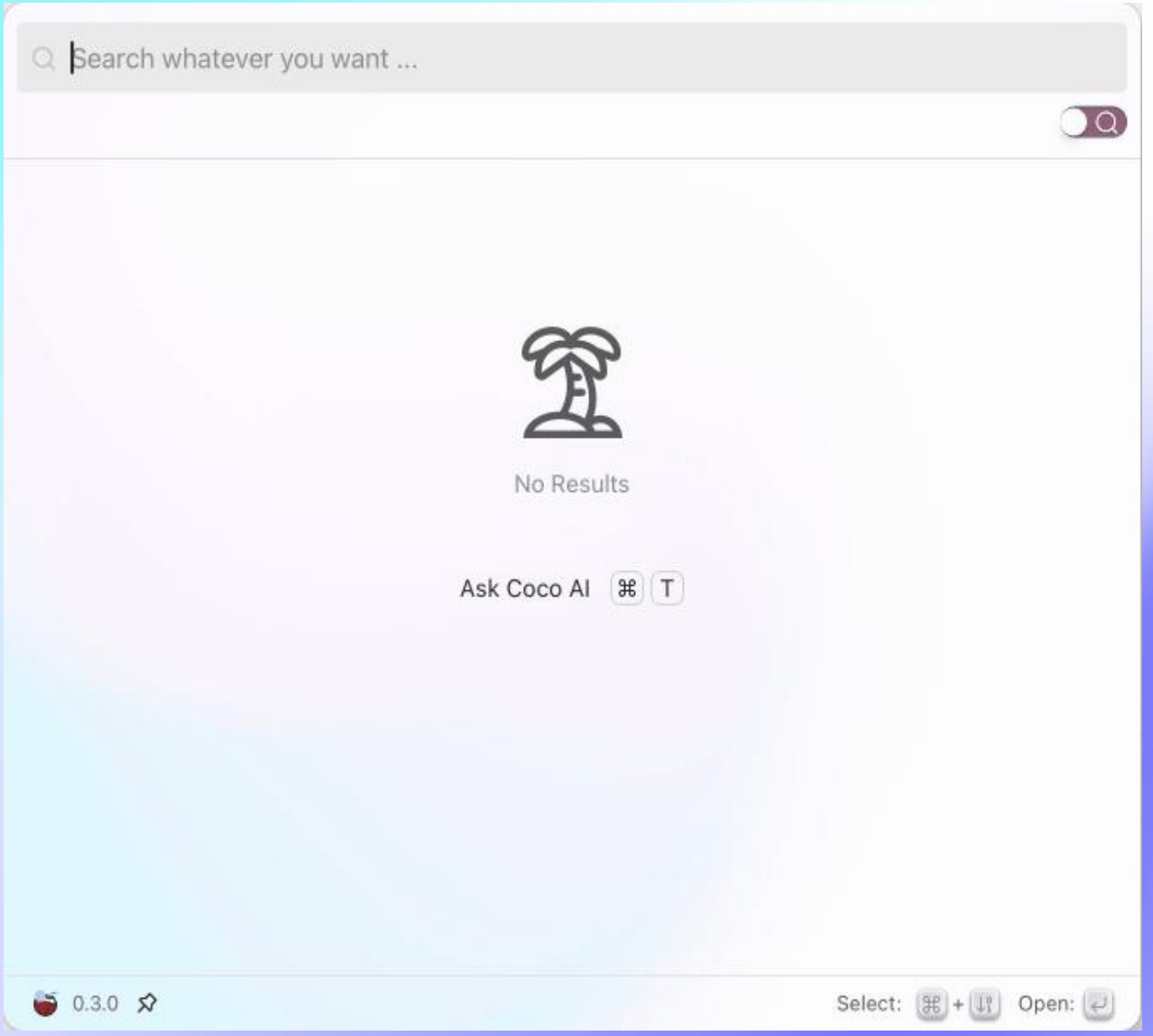
Coco AI App 背景

Coco AI App 简介

Coco AI –一站式搜索、连接与协作平台

Coco官网：https://coco.rs

源码：https://gitcode.com/infinilabs/coco-app



桌面端框架需求分析

01

轻量级与高性能

作为企业级应用，需快速启动、低内存占用，以适应不同性能设备，保障流畅体验，提升用户满意度。

02

强安全性

涉及企业敏感数据，框架需具备高安全性，防止数据泄露，保护企业信息安全，符合合规要求。

03

良好的开发体验

开发团队熟悉 Web 技术栈，希望框架能无缝对接，降低学习成本，同时支持热重载等功能，提高开发效率。

PART.02

市面上桌面端框架对比

NW.js 框架分析

优势

基于 Node.js 和 Webkit，开发门槛低，与 Web 技术栈契合度高，易于上手。

劣势

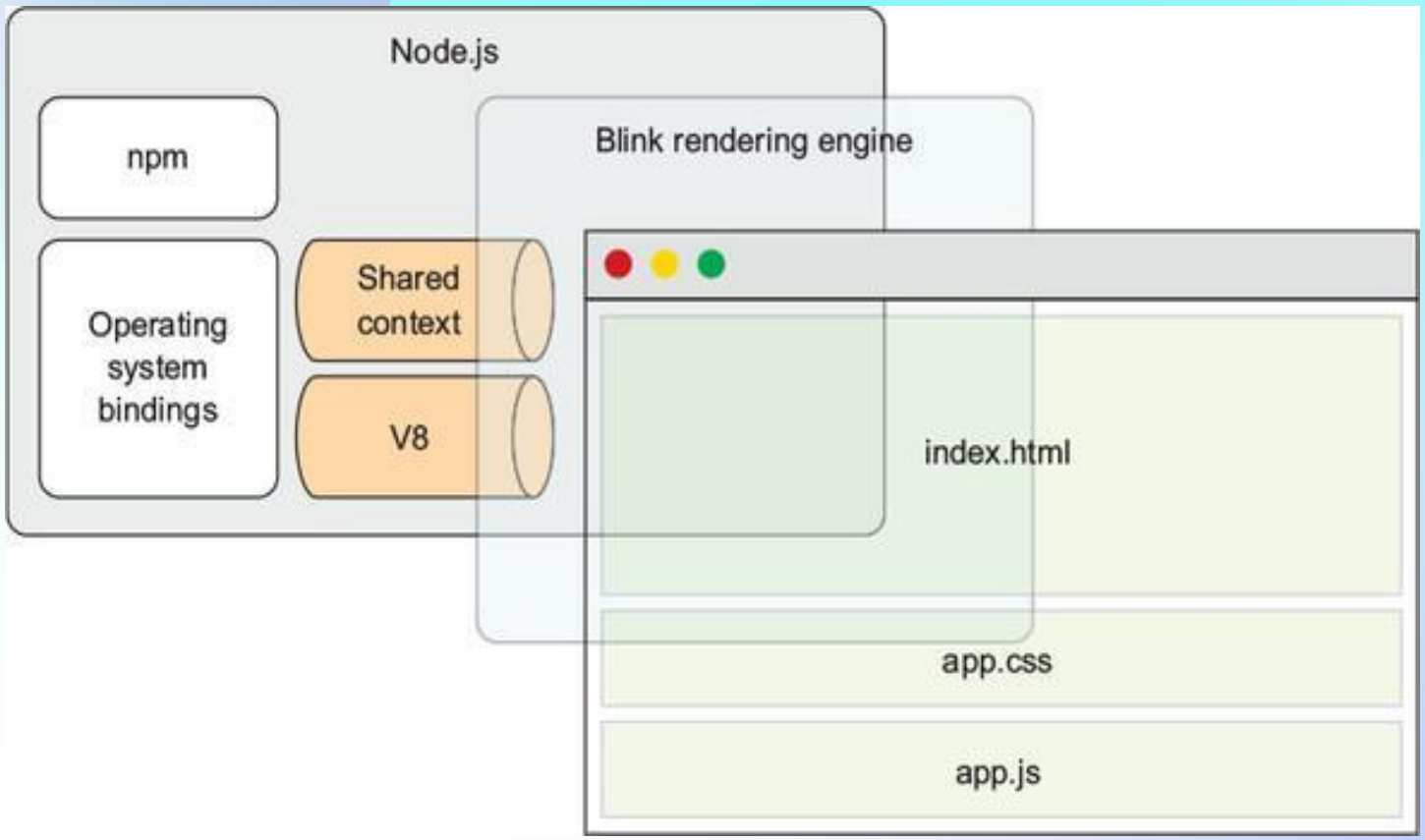
应用体积较大，性能表现一般，安全性相对不足，难以满足企业级应用对轻量级与安全性的需求。

适用场景

适用于小型、非企业级桌面应用开发，对性能与安全性要求不高，注重开发便捷性与快速迭代的项目。

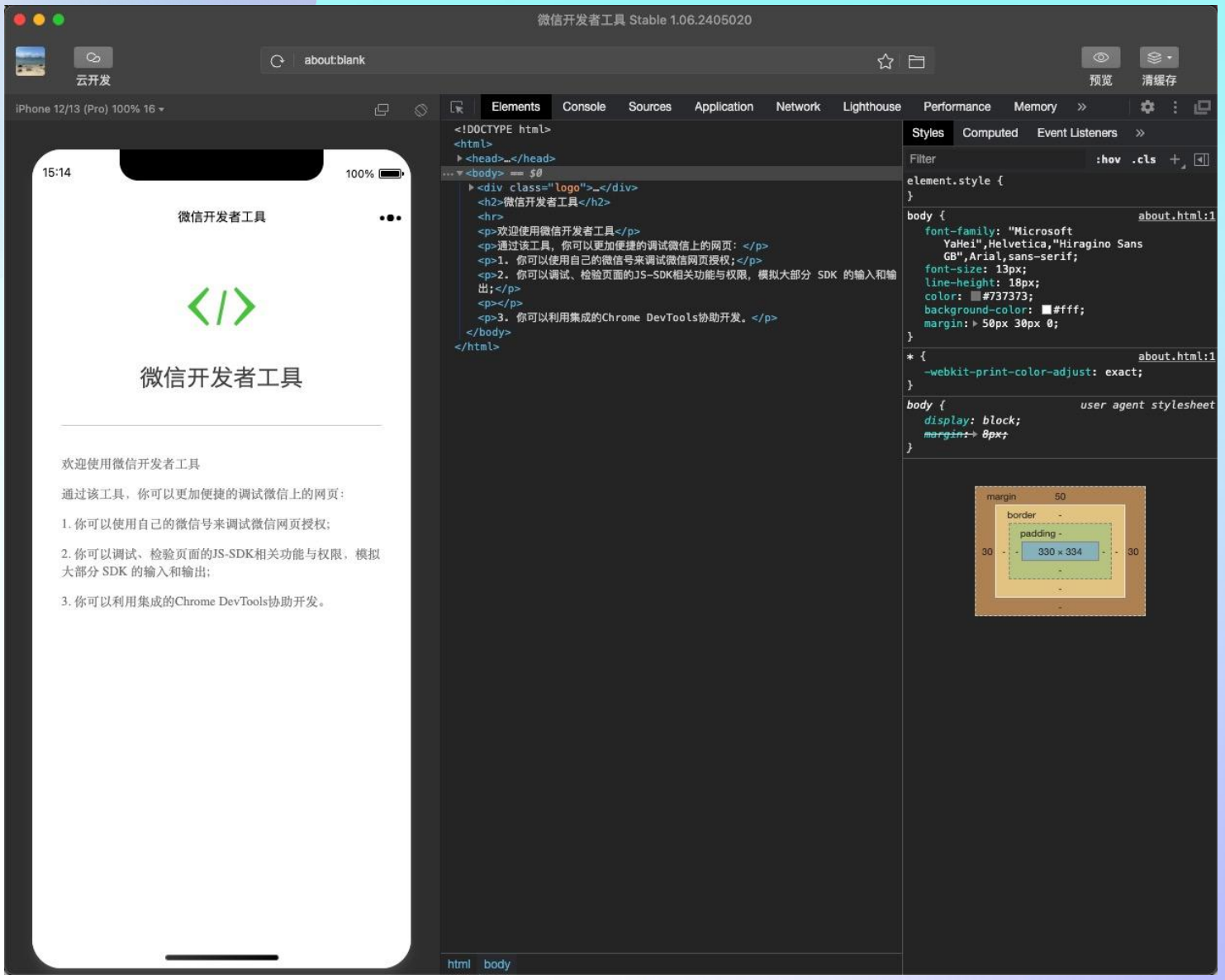
- NW.js 官网：https://nwjs.io/
- 源码：https://gitcode.com/gh_mirrors/nw/nw.js（star 40.7k）

NW.js 框架原理



1. Node 与 Chromium 的结合
2. 模块加载与打包
3. 调试与热重载

NW.js 框架应用



官方统计的一些 app 列表：
https://github.com/nwjs/nw.js/wiki/List-of-apps-and-companies-using-nw.js

Electron 框架分析

01

优势

生态系统丰富，插件众多，开发资源充足，能快速实现复杂功能，社区支持强大。

02

劣势

应用体积庞大，启动慢，内存占用高，在性能敏感场景下表现不佳，且存在安全漏洞风险。

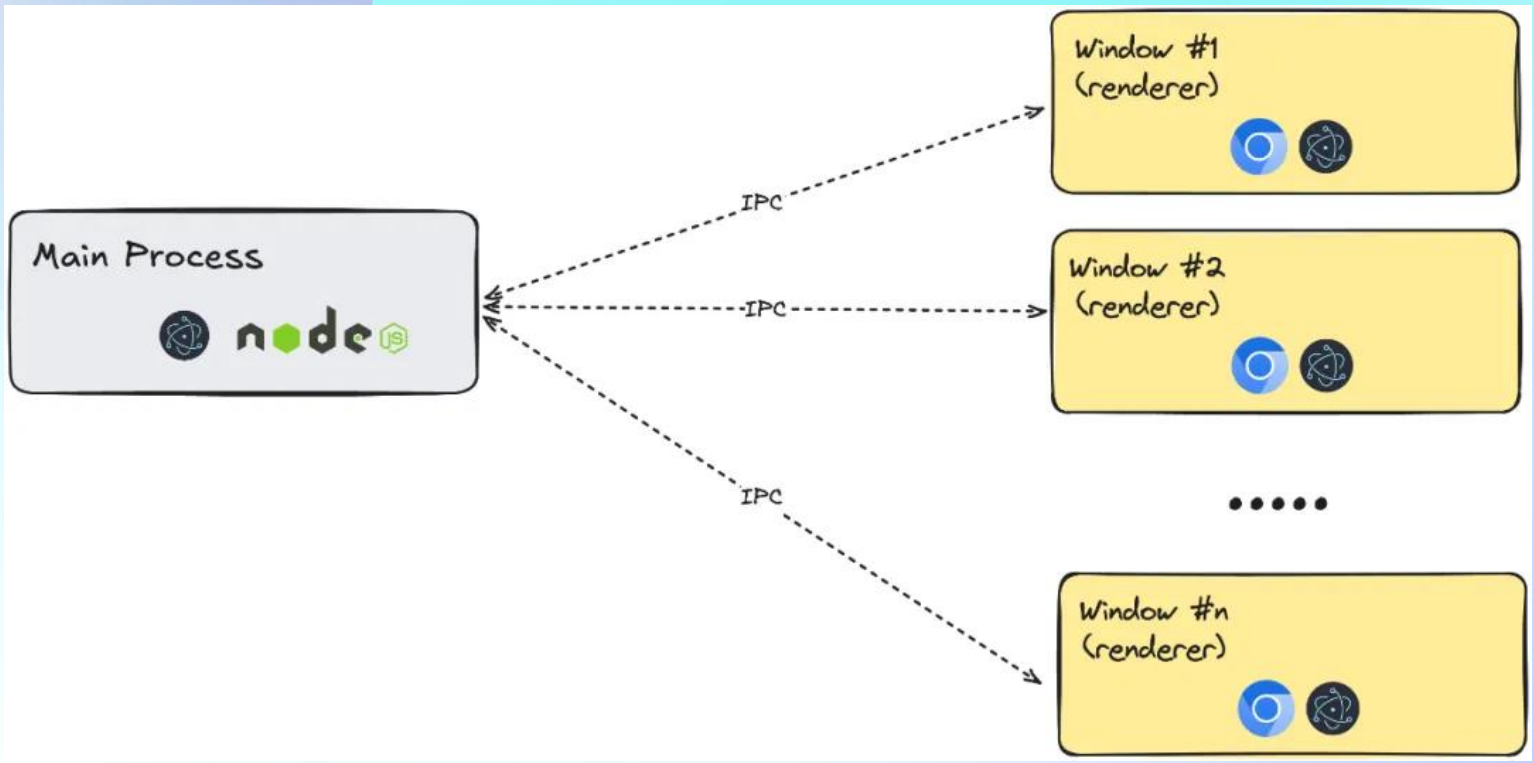
03

适用场景

适合企业级复杂应用开发，需快速开发、功能丰富，对应用体积与性能要求不苛刻，有足够资源优化性能。

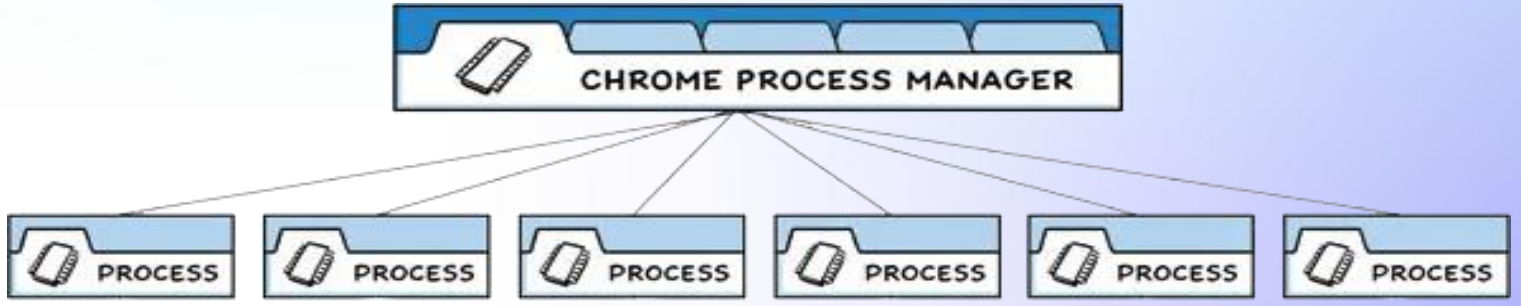
- Electron 官网：https://www.electronjs.org（v33.0）
- 源码：https://gitcode.com/gh_mirrors/el/electron （Star 114k）

Electron 框架原理



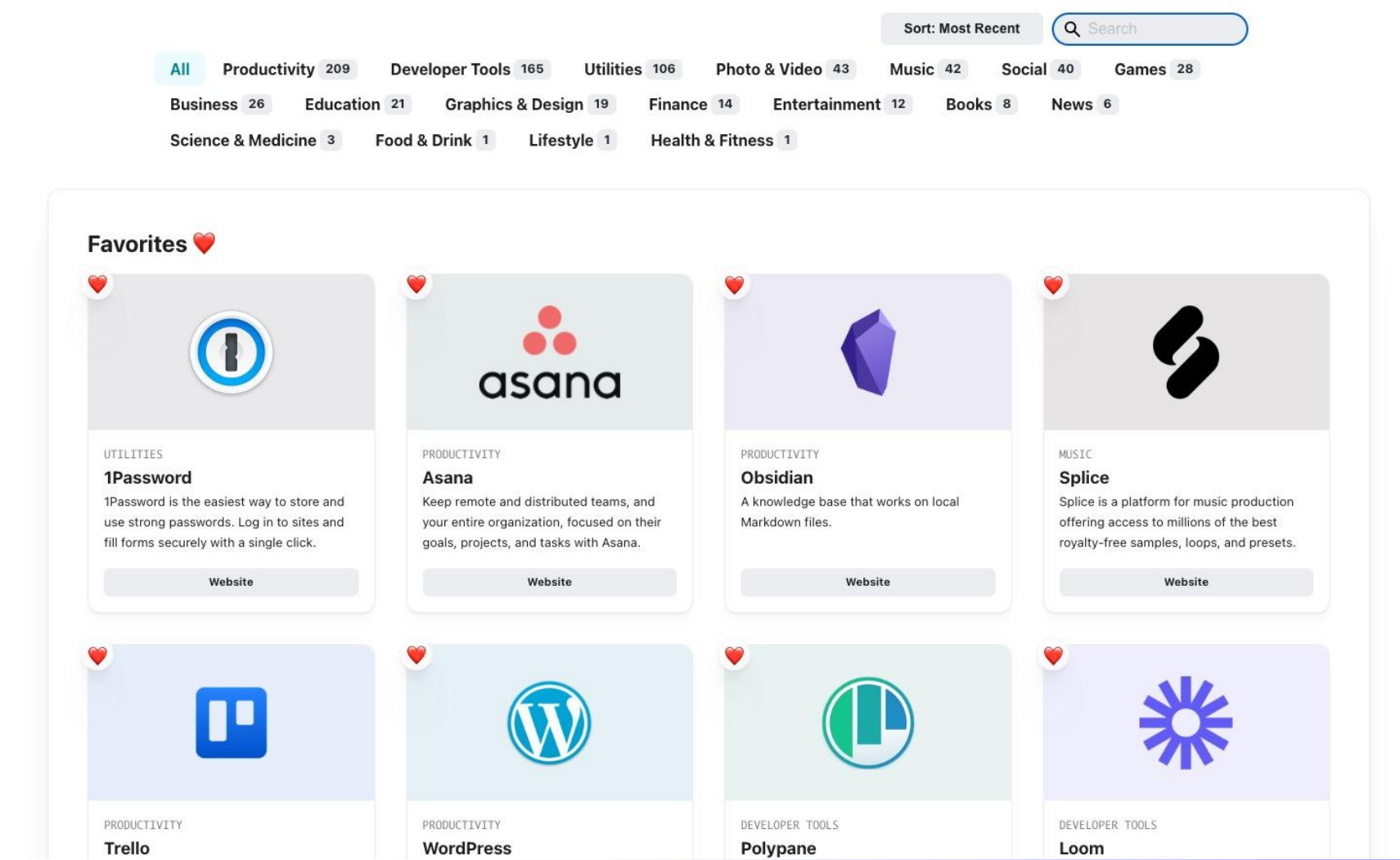
1. Electron 的主进程是一个 Node.js 进程。

2. 可以将渲染进程视为由主进程生成的单个浏览器标签页。



一个拥有多个窗口的应用会同时运行多个渲染进程。

Electron 框架应用



官方统计的一些 app 列表：
https://www.electronjs.org/apps

Tauri 框架分析

01 优势

应用体积小，启动快，内存占用低，性能卓越，基于 Rust 开发，安全性高，契合企业级应用需求。

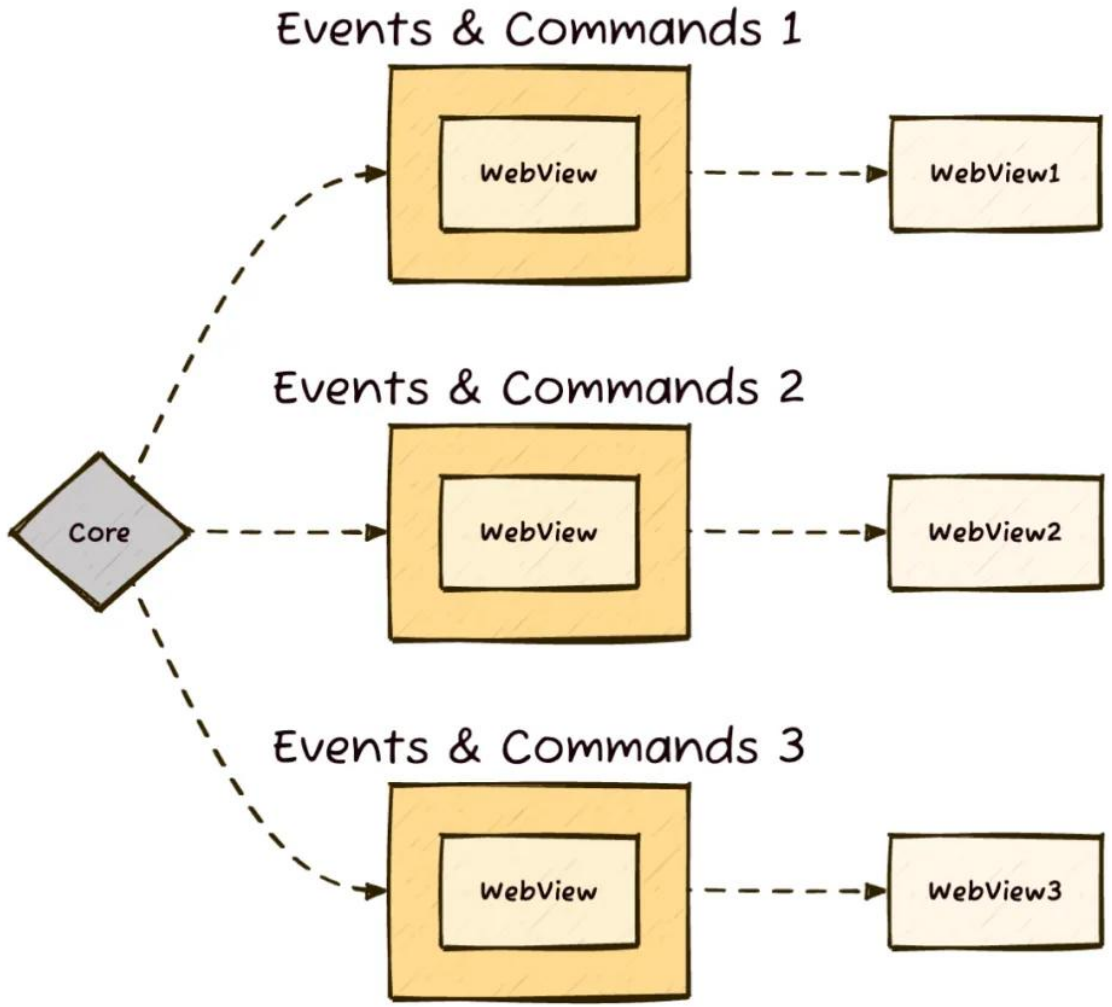
02 劣势

Rust 学习曲线陡峭，开发初期上手难度大，插件生态相对薄弱，需自行开发部分功能。

03 适用场景

理想选择为轻量级、高性能、安全性要求高的企业级桌面应用，开发团队愿投入时间学习 Rust 语言。

Tauri 框架原理



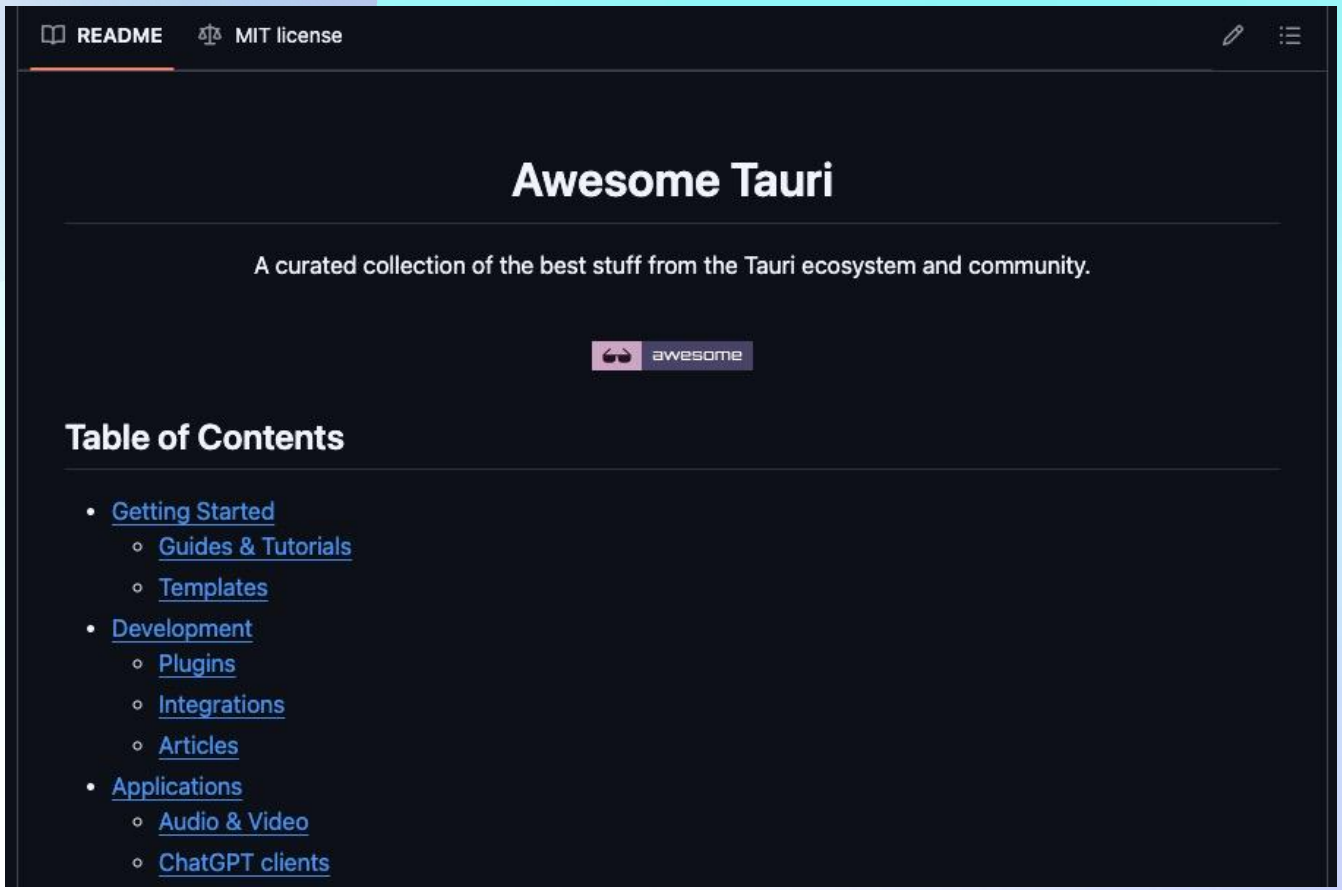
主进程（Rust后端）

- ✅ **优势：**
  - 编译为原生二进制，无Node.js运行时依赖
  - 极简体积（无需捆绑运行时）
- ⚠️ **权衡：**
  - Rust学习曲线陡峭
  - 原生API调用需额外开发

WebView进程（系统内核驱动）

- **各平台实现：**
  - Windows：WebView2（Edge/Chromium内核）
  - macOS：WKWebView（Safari/WebKit内核）
  - Linux：WebKitGTK
- ⚠️ **跨平台代价：**
  - UI表现差异（如Safari与Chrome的CSS兼容性问题）
  - 需针对性适配不同内核特性

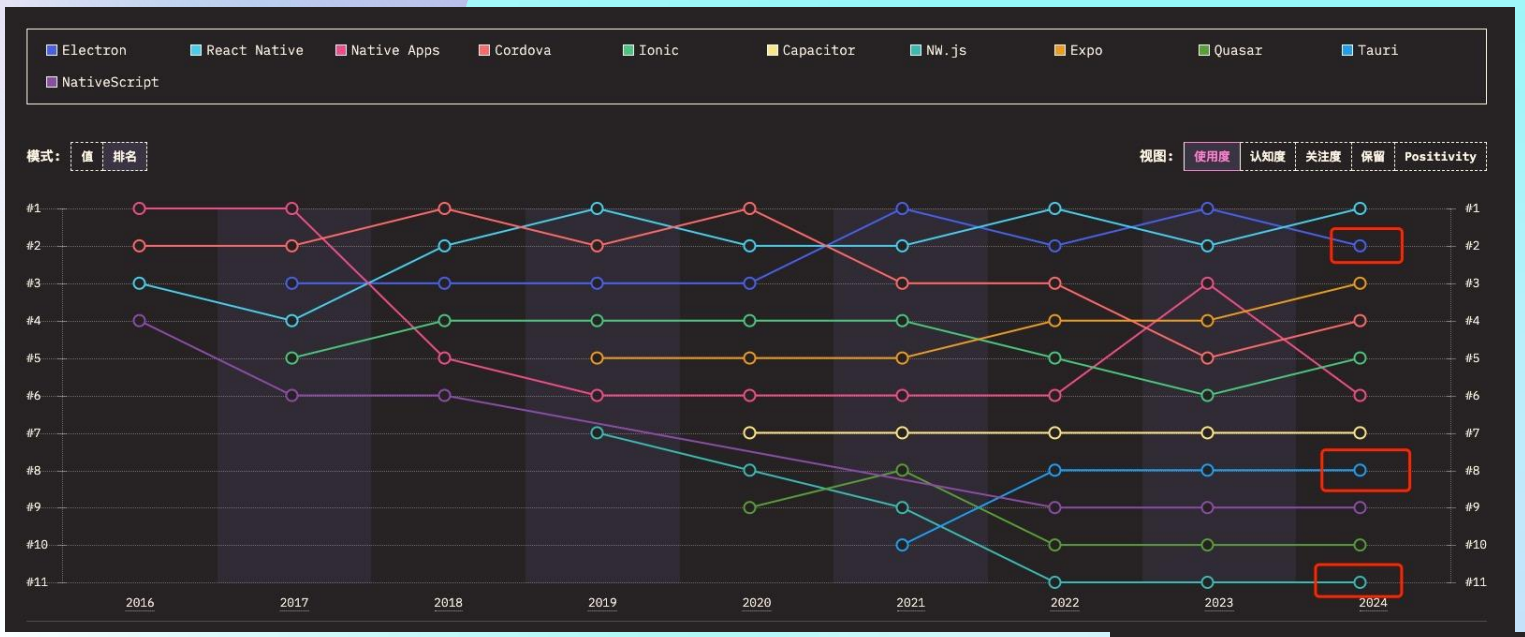
Tauri 框架应用



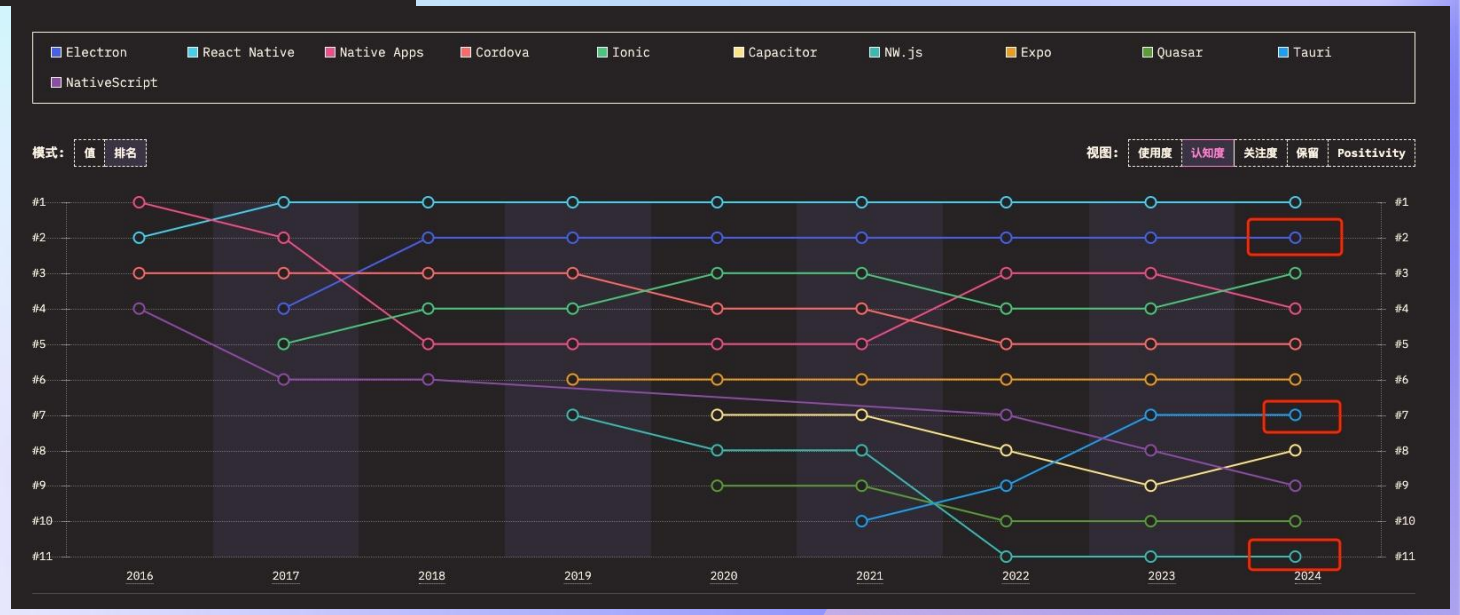
官方统计的一些 app 列表：
https://gitcode.com/gh_mirrors/aw/awesome-tauri

- 官网：https://tauri.app （v2.0）
- 源码：https://gitcode.com/gh_mirrors/ta/tauri （Star 84k）

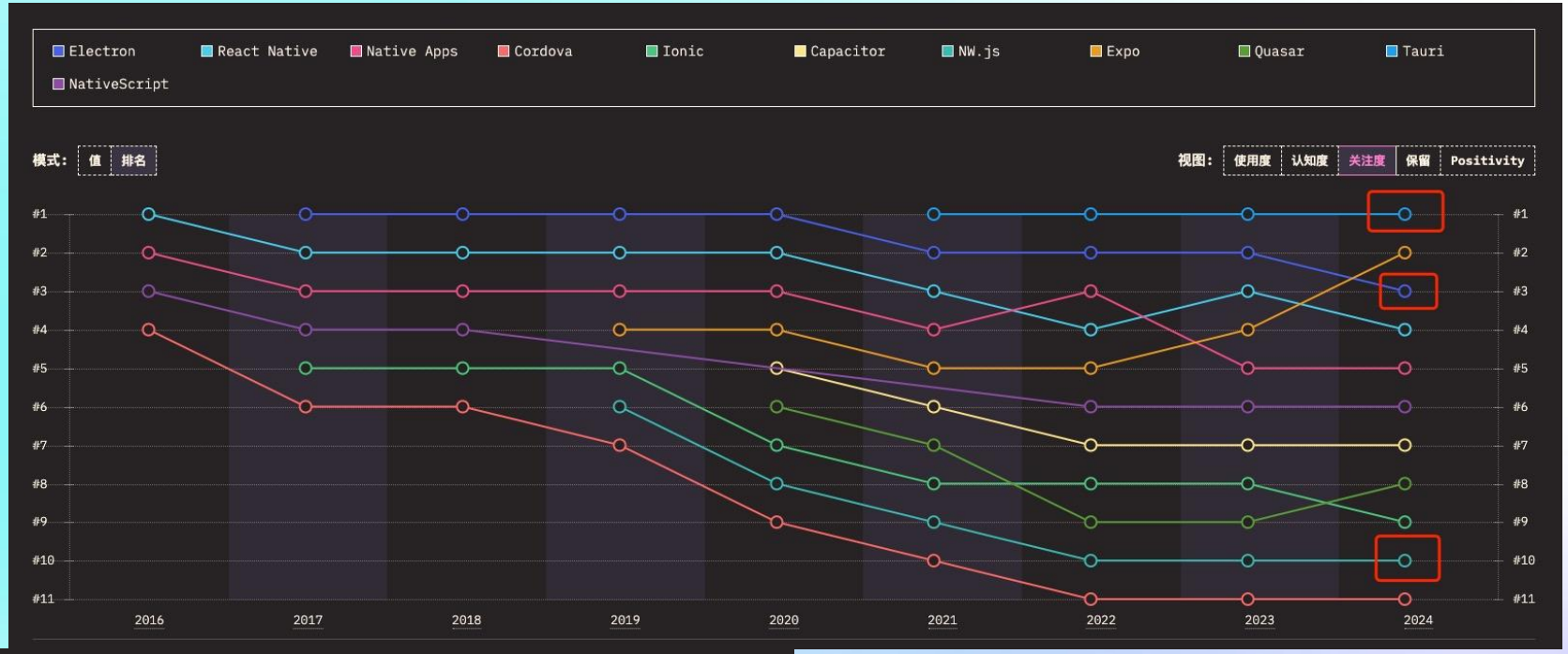
Tauri VS 其它框架

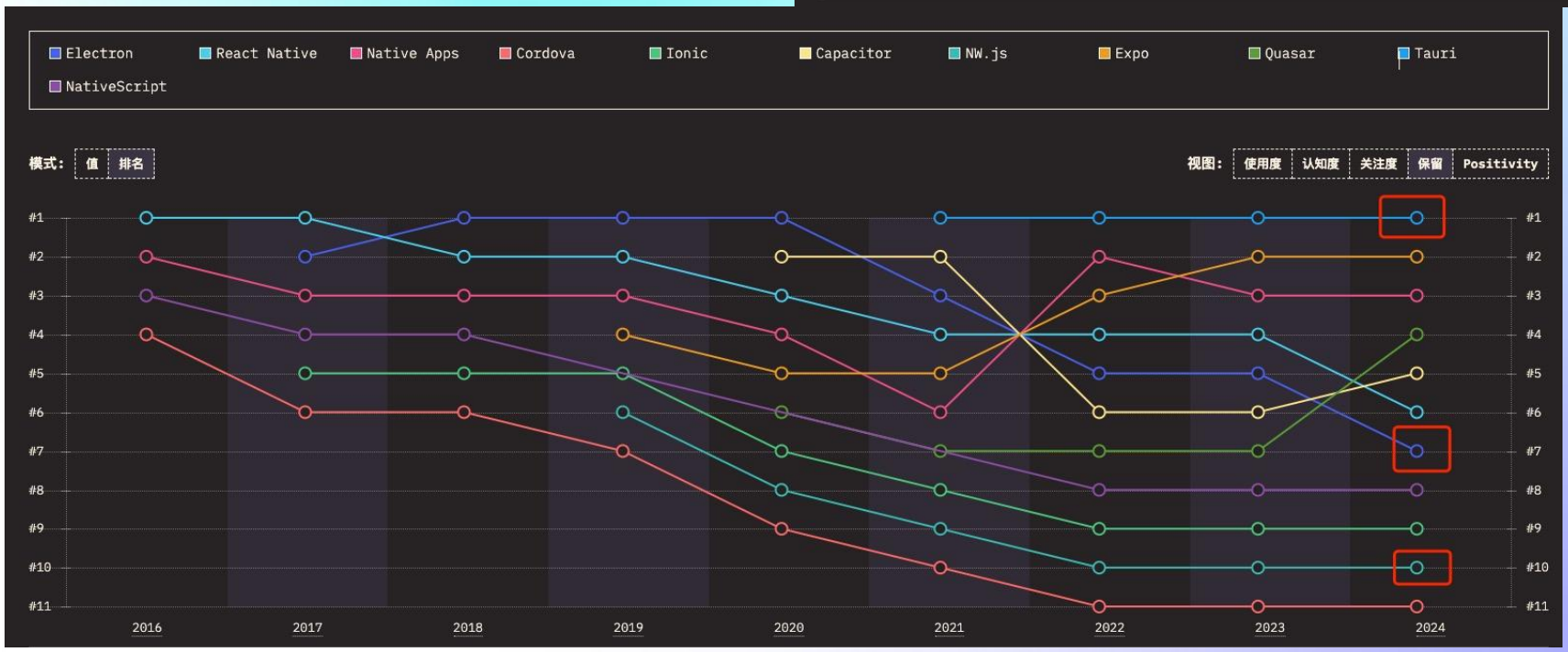


数据来源：https://2024.stateofjs.com/zh-Hans/libraries/mobile_desktop/



Tauri VS 其它框架





PART.03

选择Tauri V2作为开发框架

Tauri V2 版本特性

独立于前端

将你现有的网络技术栈带到 Tauri 或开始新的项目。 Tauri 支持任何前端框架，所以你不需要改变你的技术栈。

进程间通讯

用 JavaScript 编写前端，用 Rust 编写应用程序逻辑，并使用 Swift 和 Kotlin 在系统中深入集成。

最小化体积

使用操作系统的本地网页渲染器，Tauri 应用的体积可以达到最小 600KB。

跨平台

使用单个代码库为 Linux、macOS、Windows、Android 和 iOS 构建你的应用程序。

高安全性

Tauri 团队的首要目标，推动我们的首要任务和最大的创新。

由 Rust 驱动

以性能和安全性为中心，Rust 是下一代应用程序的语言。

Tauri V2 的优劣势分析

01

优势

在体积、性能、安全性方面表现优异，满足 Coco AI App 轻量级、高性能、强安全的核心需求，助力打造优质企业级应用。

02

劣势

尽管生态逐渐完善，但与 Electron 等相比仍有差距，部分复杂功能开发需投入更多精力，开发团队需持续学习与探索。

选择Tauri V2的原因

符合项目核心需求

Tauri V2 的特性与 Coco AI App 追求的高效、安全、跨平台体验高度契合，为项目成功奠定基础。

长期发展潜力

Tauri 社区活跃，版本更新迅速，未来发展可期，选择其作为框架利于项目长期稳定发展，紧跟技术前沿。

团队技术适应性

经过评估，开发团队认为投入学习 Rust 的收益大于成本，且 Tauri 的开发模式与团队现有技术栈兼容性良好，可平稳过渡。

PART.04

Tauri V2开发中遇到的问题与解决方案

插件生态不足

问题描述

生态局限性：复杂功能需自研，个别领域插件缺失

开发影响：依赖自主研发 → 周期延长 + 技术验证风险上升

解决方案

开发团队基于项目需求自行开发关键插件，并将其开源贡献至社区，既满足自身需求，又推动Tauri生态发展。

tauri-plugin-macos-permissions
tauri-plugin-screenshots
tauri-plugin-fs-pro
tauri-plugin-mic-recorder
tauri-plugin-windows-version
tauri-plugin-clipboard-x
tauri-plugin-locale

跨平台兼容性挑战

01

问题描述

不同操作系统上 WebView 表现不一致，导致 UI 渲染效果、功能实现存在差异，影响用户体验。

02

解决方案

针对各平台特性进行适配开发，制定详细的跨平台测试流程，确保应用在不同系统上表现一致，提升应用品质。

性能优化瓶颈

01

问题描述

随着应用功能不断增加，部分复杂操作如大规模数据处理、实时交互时出现性能瓶颈，影响流畅度。

02

解决方案

深入分析性能瓶颈，优化 Rust 代码，采用多线程、异步编程等技术，同时借助 Tauri 社区资源寻求优化方案，持续提升性能。

INFINI Labs

极限科技·让搜索更简单

2025

谢谢大家

https://gitcode.com/infinilabs

主讲人：Rain9

时间：2025.4