Rust & WebAssembly

结合

为什么选择 Rust?

性能强

系统级的编程语言
没有GC暂停、性能抖动

体积小

标准库很小高效性设计
结合工具几乎无额外代码

互操作

Rust 可和 JS 很好互调用
三者之间可以很好协同工作

工具链

官方工具链 Cargo 支持
wasm-pack & wasm-opt

使用场景

Mozilla 用 Rust 编译成 WASM
提升 Source Maps 的性能

Cloudflare 将 Rust 编译成 WASM
在 Cloudflare Workers 中调用它

Shopify 用 Rust 实现模板引擎编译成 WASM
提高运行效率，减轻服务端压力

Dropbox 用 Rust 编译成 WASM
轻松地 DivANS 编解码器嵌入到网页中

Rust & 前端工具链

结合

🐢 JS 编译工具慢是原罪?

单线程语言 | 抽象 AST | 众多插件 | 源代码映射 | 兼容性转换 | I / O 操作

🐆 Turbopack、SWC 为什么这么快?

1. Rust 语言本身：设计用于处理大量的并发和高性能的工作负载，编译器优化后性能接近 C 的水平
2. 并行编译提速：可以并行处理任务，充分利用多核心处理器的性能，尤其在处理大型项目时
3. 优化的算法和数据结构：高度优化的机器代码和低层级增量计算引擎，可以缓存到单个函数的级别
4. 少量的 I/O 操作：尽量减少了 I/O 操作，因为 I/O 操作通常是造成编译器慢的一个重要原因