WebAssembly

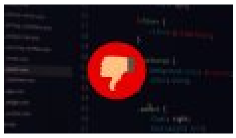
WebAssembly

WebAssembly .WASM

WebAssembly **WebAssembly** **WebAssembly** Emscripten **WebAssembly** asm.js C/C++ **WebAssembly** **WebAssembly** **WebAssembly** **WebAssembly** **WebAssembly** **WebAssembly** CAD

در ادامه ما مراحلی را که شما برای به کارگیری اجزای **WebAssembly** در یک اپلیکیشن وب به آن نیاز دارید، بررسی خواهیم کرد . از آنجا که **WebAssembly** یک پروژه ناتمام است که هنوز به توسعه و تکمیل نیاز دارد، این مراحل کاملا به زبانی که شما استفاده میکنید وابسته است . اما در حال حاضر، این امکان وجود دارد که اپلیکیشنهای وب اسمبلی را در اندازه کوچک نوشته و گسترش داد .

مطلب پیشنهادی



تقابل علاقه و اجبار : بر اساس یافتههای سایت استکاورفلو

10 زبانی که برنامهنویسان علاقه دارند و _ **15** زبانی که بیزارند

موارد کاربرد WebAssembly

اصلیترین مورد استفاده از وب اسمبلی به عنوان یک هدف برای نوشتن نرمافزارهای درون مرورگر است که از زبانی غیر از جاوااسکریپت استفاده میکند . **WebAssembly** برای موارد استفاده مبتنی بر مرورگر که عملکرد بالایی نیاز دارد طراحی شده است . کاربردهایی مثل بازی، استریم موسیقی، تدوین ویدیو، CAD، کدگذاری، تشخیص تصویر و مواردی شبیه به این .
به طور کلی، هنگام تعیین مورد خاص استفاده تمرکز روی این سه حوزه کمککننده است :

- کدهایی با عملکرد بالا که از قبل در یک زبان هدف موجود بوده است . برای نمونه، اگر شما یک تابع ریاضی پر سرعت را در اختیار دارید که قبلا به زبان C نوشته شده و میخواهید آن را در یک اپلیکیشن وب به کار بگیرید، میتوانید آن را به عنوان یک ماژول **WebAssembly** اجرا شده از طریق جاوا اسکریپت به کار بگیرید . سایر بخشهای این اپلیکیشن را که به توان عملیاتی بالایی نیاز ندارد میتوان به صورت جاوا اسکریپت باقی نگه داشت .
- کدهایی با عملکرد بالا که از ابتدا به نوشتن نیاز دارند و جاوا اسکریپت برای این منظور ایدهال نیست. قبلا ممکن بود از asm.js برای نوشتن چنین کدهایی استفاده شود . شما هنوز هم میتوانید چنین کاری انجام دهید اما WebAssembly به عنوان یک راهکار بلند مدت بهتر عمل میکند .
- انتقال اپلیکیشنهای دسکتاپ به محیط وب . بخش عمدهای از موارد استفاده از این فناوری برای asm.js و **WebAssembly** در همین رابطه است. وب اسمبلی میتواند زیرساختی برای اپلیکیشنهایی فراهم کند که بلندپروازانهتر از صرفا یک رابط گرافیکی HTML هستند.

اگر یک اپلیکیشن جاوا اسکریپت آماده دارید که مشکل عمدهای برای اجرا و وضعیت عملکرد ندارد، در این شرایط نیازی به توسعه **WebAssembly** نخواهید داشت. اما اگر میخواهید این اپلیکیشن سریعتر اجرا شود، **WebAssembly** میتواند کمککننده باشد .

زبانهای پشتیبانی شده توسط WebAssembly

گام بعدی برای اجرای یک اپلیکیشن WebAssembly انتخاب زبانی است که در نهایت میتواند به WebAssembly کامپایل شود . به احتمال زیاد حداقل یکی از زبانهای اصلی که شما در تولیدات نرمافزاری خود از آنها استفاده میکنید یک کامپایلر را که میتواند به WebAssembly تبدیل شود، خواهد داشت. در ادامه با اصلیترین آنها آشنا خواهید شد :

- **C:** بدون تردید متداولترین روش انجام آن از طریق Emscripten است، زیرا C به Emscripten – به WebAssembly اولین زنجیره ابزار WebAssembly بود که معرفی شد. بعد از آن ابزارهای دیگری نیز ارائه شد . یک کامپایلر کامل به نام Cheerp به طور خاص برای تولید اپلیکیشنهای WebAssembly از کدهای C/C++ تدارک دیده شده است. این کامپایلر را میتوان برای جاوا اسکریپت، asm.js یا ترکیبی از همه آنها نیز به کار برد .

- **Rust:** [illegible]
- **TypeScript:** [illegible]
- **Java:** [illegible]
- **Lua:** [illegible]
- **Kotlin/Native:** [illegible]
- **Net:** [illegible]
- **LLVM.[illegible]:** [illegible]

[illegible]

[illegible]

[illegible]

WebAssembly [illegible]

[illegible]

ویژگیهای WebAssembly

WebAssembly هنوز مراحل اولیه توسعه خود را سپری میکند، بنابراین زنجیره ابزار و اجرای آن هنوز به امکان سنجی و تایید وابسته است . این به این معنا است که متولیان WebAssembly قصد دارند با یک سری از ابتکارات مزایای WebAssembly را افزایش دهند :

بازیافت حافظه

WebAssembly مستقیما از زبانهایی که از مدلهای بازیافت حافظه یا زباله روبی استفاده میکنند، پشتیبانی نمیکند . از زبانهایی مثل لوا و پایتون تنها با مجموعه ویژگیهای محدود یا با الحاق کل چرخه اجرا به عنوان فایل اجرایی WebAssembly پشتیبانی میشود . پشتیبانی از مدلهای بازیافت حافظه صرف نظر از زبان قابلیتی است که هنوز مراحل ابتدایی خود را پشت سر میگذارد .

Threading

پشتیبانی ذاتی از Threading در زبانهایی مثل Rust یا C/C++ رایج است. فقدان پشتیبانی از Threading در WebAssembly به این معنا است که در چنین زبانهایی نمیتوان از تمام کلاسهای نرمافزار استفاده کرد . روش پیشنهادی برای اضافه کردن Threading به WebAssembly استفاده از مدل Threading زبان C++ است.

عملیات حافظه انبوه و SIMD

همسانی عملیات حافظه انبوه و SIMD (یک دستورالعمل، چندین داده) از ضروریات هر اپلیکیشنی (مثل یادگیری ماشین یا اپلیکیشنهای علمی) است که از طریق جریان داده و شتاب پردازنده مرکزی از کمبود منابع جلوگیری میکند . پیشنهاداتی برای اضافه کردن این نوع از قابلیتها به WebAssembly از طریق عملگرهای جدید ارائه شده است .

ابزار دیباگ کردن و نمایه سازی

به دلیل ناتوانی در ارتباط بین کد ترجمه شده و منبع، یکی از بزرگترین چالشها با جاوا اسکریپت انتقال داده شده سخت شدن دیباگ و نمایه سازی است. با WebAssembly نیز ما با چالش مشابهای روبرو هستیم. با استفاده از روش به کار گرفته شده در مورد جاوا اسکریپت (پشتیبانی نقشه منبع) میتوان راهکاری برای آن پیدا کرد .

نویسنده :
محسن آقاجانی
منبع :
infoworld
دسته بندی:
کارگاه
برنامه نویسی
تاریخ انتشار :
00:10 - 21/03/1397
برچسب:
WebAssembly - اپلیکیشن پاینری - SIMD - کاربرد asm.js - Emscripten - wasm - WebAssembly - جاوااسکریپت

نشانی
□□□□:https://www.shabakeh-mag.com/workshop/13003/%DA%86%DA%AF%D9%88%D9%86%D9%87-%D8%A8%D8%A7-webassembly-%D8%A7%D9%BE%D9%84%DB%8C%DA%A9%DB%8C%D8%B4%D9%86%E2%80%8E%D9%87%D

8%A7%DB%8C-%D8%A8%D8%A7%DB%8C%D9%86%D8%B1%DB%8C-%D8%A8%D8%A7-%D8%B9%D9%85%D9%84%DA%A9%D8%B1%D8%AF-%D8%A8%D8%A7%D9%84%D8%A7-%D8%B1%D8%A7-%D8%A8%D9%87-%D9%85%D8%B1%D9%88%D8%B1%DA%AF%D8%B1-%D9%88%D8%A7%D8%B1%D8%AF-%DA%A9%D9%86%DB%8C%D9%85