LAB 3SA04 : Intro to Web Assembly

ตอนที่ 1: สรุปความรู้ที่ได้จากการทดลอง

จากการทดลองครั้งนี้ ได้เรียนรู้ว่า Web Assembly (WASM) เป็นเทคโนโลยีที่ถูกพัฒนาขึ้นเพื่อแก้ปัญหาข้อจำกัดด้านประสิทธิภาพของ JavaScript ในการพัฒนา Application บนเว็บเบราว์เซอร์ โดย WASM เป็นรูปแบบไบนารี (binary format, ให้นึกภาพภาษา Assembly) ที่สามารถทำงานได้เร็วกว่า JavaScript แบบเดิม ๆอย่างมาก เพราะใช้สารตั้งต้นที่มี performance ดีระดับต้น ๆ ของ programming language ทั้งหมด

กระบวนการคอมไพล์และการทำงาน

การนำ WASM ไปใช้งานเริ่มต้นจากการเขียนโค้ดในภาษาโปรแกรมระดับสูง (high-level language) เช่น C, C++, หรือ Rust จากนั้นจึงคอมไพล์ให้เป็นรหัส WASM โดยใช้คอมไพเลอร์เฉพาะ เช่น Clang หรือที่เป็นที่นิยมอันอันก็คือ emscripten สำหรับภาษา C ในการทดลองนี้ได้เรียนรู้การใช้คำสั่ง clang --target=wasm32 --no-standard-libraries -Wl,--export-all -Wl,--no-entry -o findsquare.wasm findsquare.c

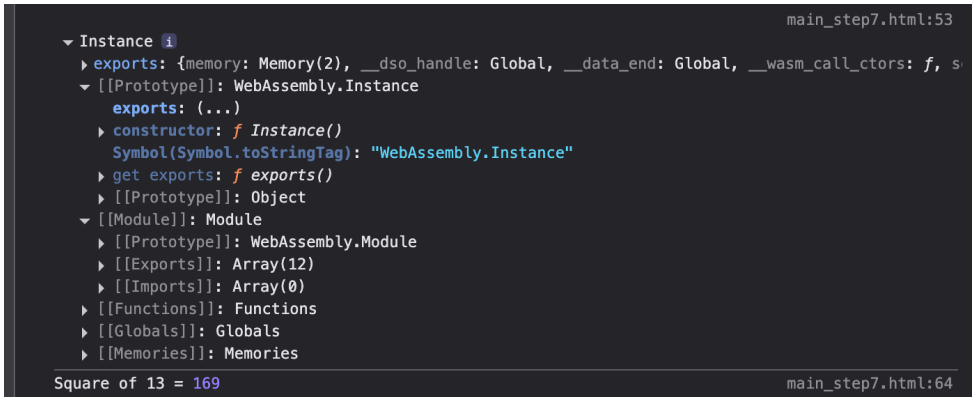
เพื่อคอมไพล์ฟังก์ชัน square() ที่เขียนด้วยภาษา C ให้เป็นไฟล์ WASM

การโหลดและใช้งาน WASM ใน JavaScript

จากการทดลอง ได้เรียนรู้ขั้นตอนการโหลด WASM ในเบราว์เซอร์ผ่าน JavaScript API ที่ประกอบด้วย

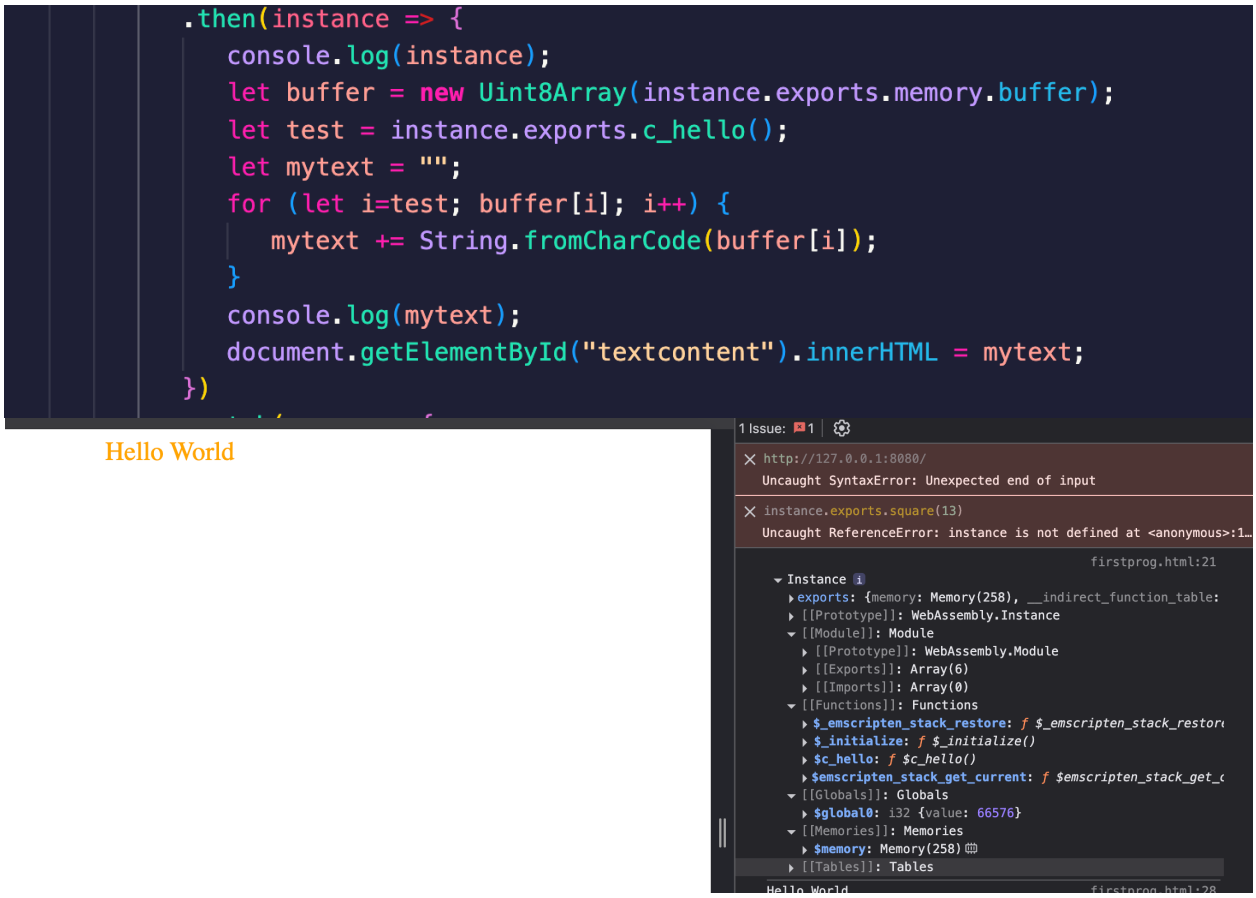
1. **การดึงไฟล์ WASM**: ใช้ fetch() เพื่อโหลดไฟล์ .wasm
2. **การคอมไพล์**: ใช้ WebAssembly.compile() เพื่อแปลง binary เป็น module
3. **การสร้าง Instance**: ใช้ WebAssembly.Instance() เพื่อสร้าง instance ที่พร้อมใช้งาน
4. **การเรียกใช้ฟังก์ชัน**: เข้าถึงฟังก์ชันผ่าน instance.exports

ภาพที่ 1 console log จากการทดลองที่ 1



การจัดการหน่วยความจำ (Memory Management)

สิ่งที่น่าสนใจจากการทดลองตอนที่ 2 คือการทำงานกับหน่วยความจำใน WASM โดยเฉพาะกับฟังก์ชัน c_hello() ที่คืนค่าเป็น pointer ซึ่งเป็นตำแหน่งในหน่วยความจำ (ไม่ใช่ข้อมูลโดยตรง) การเข้าถึงข้อมูลจริงต้องอ่านจาก memory buffer



ภาพที่ 2 และ 3 console log จากการทดลองที่ 2

สิ่งนี้แสดงให้เห็นถึงการทำงานระดับต่ำของ WASM ที่**ต้องจัดการหน่วยความจำด้วยตนเอง**

ข้อดีและข้อจำกัด

ข้อดี:

- ประสิทธิภาพสูง: ทำงานเร็วกว่า JavaScript มาก
- ความปลอดภัย: ทำงานใน sandboxed environment

- ความเป็นอิสระจากแพลตฟอร์ม: รันได้บนเบราว์เซอร์ต่างๆ
- ความเข้ากันได้: สามารถทำงานร่วมกับ JavaScript ได้

ข้อจำกัด:

- ความซับซ้อนในการพัฒนา: ต้องเข้าใจการทำงานระดับต่ำ (Low-level)
- การจัดการหน่วยความจำ: ต้องจัดการ memory access ด้วยตนเอง
- ขนาดไฟล์: อาจมีขนาดใหญ่กว่า JavaScript ในบางกรณี

ตอนที่ 2: ความคิดเห็นของตนเองต่อตัวเทคโนโลยี WASM

WebAssembly เป็นเทคโนโลยีที่ศักยภาพสูงสำหรับอนาคตของการพัฒนา Web Application โดยเฉพาะอย่างยิ่งสำหรับงานที่ต้องการประสิทธิภาพสูง เช่น การประมวลผลภาพ การเล่นเกม การจำลองทางวิทยาศาสตร์ หรือเครื่องมือออกแบบกราฟิก WASM ช่วยเชื่อมช่องว่างระหว่างประสิทธิภาพของ Desktop Application กับความสะดวกในการเข้าถึงของ Web Application

อย่างไรก็ตาม การนำ WASM ไปใช้งานจริงยังคงมีความท้าทายในเรื่องของความซับซ้อนในการ develop และการ debug เมื่อเทียบกับ JavaScript แบบดั้งเดิม Devต้องมีความเข้าใจในการทำงานของ Low-level language และต้องพิจารณาอย่างรอบคอบว่าเมื่อใดควรใช้ WASM และเมื่อใด JavaScript ธรรมดา

ตอนที่ 3: ตัวอย่างการนำ WASM ไปใช้งานจริง

กรณีศึกษา: Figma - เครื่องมือออกแบบกราฟิกบนเว็บ

Figma เป็นตัวอย่างที่โดดเด่นของการใช้ Web Assembly ในการพัฒนา Application ที่ต้องการประสิทธิภาพสูง Figma ได้ใช้ WebAssembly ในการพัฒนาเครื่องมือออกแบบบนเว็บ ซึ่งช่วยให้สามารถสร้างประสบการณ์การใช้งานที่มีคุณภาพเทียบเท่า Desktop Application โดยไม่ต้องลดทอนประสิทธิภาพ (Keeping Figma Fast | Figma Blog, 2023b, sec. 3)

ผลลัพธ์ที่ได้รับ: การย้ายจาก asm.js ไปเป็น WebAssembly ทำให้ Figma มีประสิทธิภาพเพิ่มขึ้น 3 เท่า และโหลดเร็วขึ้น 3 เท่าเช่นกันจากการปรับปรุงการเรนเดอร์เอกสารและแก้ไขข้อบกพร่องของ WebAssembly ในปี 2018 (Figma Is Powered by WebAssembly | Figma Blog, 2017b, fig. 1)

เหตุผลที่เลือกใช้ WASM:

- ต้องการประสิทธิภาพสูงในการเรนเดอร์กราฟิกที่ซับซ้อน
- จำเป็นต้องจัดการกับข้อมูลเวกเตอร์จำนวนมาก
- ต้องการการตอบสนองแบบ real-time ในการแก้ไขออกแบบ
- ต้องการใช้โค้ด C++ ที่มีอยู่แล้วซึ่งมีประสิทธิภาพสูง

ประโยชน์ที่ได้รับ:

- ผู้ใช้สามารถใช้งานเครื่องมือออกแบบระดับมืออาชีพผ่านเว็บเบราว์เซอร์
- ไม่ต้องติดตั้งซอฟต์แวร์เพิ่มเติม
- สามารถทำงานร่วมกันแบบ real-time ได้อย่างมีประสิทธิภาพ
- มีประสิทธิภาพเทียบเท่า Desktop Application

แหล่งที่มาของข้อมูล:

- Figma is powered by WebAssembly | Figma Blog. (2017, June 8). Figma. https://www.figma.com/blog/webassembly-cut-figmas-load-time-by-3x/
- Keeping Figma fast | Figma Blog. (2023, August 29). Figma. https://www.figma.com/blog/keeping-figma-fast/