Wasm Garbage Collection in JSC

Asumu Takikawa (Igalia)
WebKit Contributors Meeting 2023

Photo credit: https://www.pexels.com/@zydeaosika-2261055/
Motivation of proposal
Wasm is a target language for compiling web programs
Adobe brings a simplified Photoshop to the web

By Jacob Kastrenakes, a deputy editor who oversees tech and news coverage. Since joining The Verge in 2012, he’s published 5,000+ stories and is the founding editor of the creators desk.

Oct 30, 2021, 6:00 AM PDT | 0 Comments / 9 News

WEBASSEMBLY (WASM)

In order to provide both security and portability, it was decided to move away from add-ons distributed as ZIPs in favour of add-ons distributed as WebAssembly modules. In order to do so without requiring a full rewrite of existing add-ons, a new platform toolset was designed for Visual Studio with the following capabilities:

- Direct compilation of C/C++ projects into WebAssembly (WASM).
- Debugging of WebAssembly modules by attaching to the game executable.
- Full support for the standard C library.
- Large support for the standard C++ library (see below).
- GLFW wrapper based on the NativeGL API to facilitate porting existing add-ons.
Wasm MVP great for C/C++/Rust programs

- Works well for languages without GC/managed runtimes
- But what about programs in languages needing GC?
Key missing piece was Wasm support for allocatable memory with GC.
GC Proposal enables those languages

- Supports allocatable data types like structs & arrays
- New kinds of reference types pointing to allocated values
- Type casts & advanced types
- Takes advantage of browser JS engines’ built-in GC

https://github.com/WebAssembly/gc/ for the nitty-gritty details
A concrete example

(module
  (type $s (struct (field $x i32)
                     (field $y i32)))
  (global (ref $s)
           (struct.new (i32.const 42)
                       (i32.const 42)))
  (func $get-x (param (ref $s))
               (result i32)
               (struct.get $s $x (local.get 0))))
A concrete example

```plaintext
(module
  (type $s (struct (field $x i32)
                   (field $y i32)))
  (global (ref $s)
    (struct.new (i32.const 42)
                (i32.const 42)))
  (func $get-x (param (ref $s)) (result i32)
    (struct.get $s $x (local.get 0))))
)
```

Globals can be init with new types
A concrete example

(module
  (type $s (struct (field $x i32)
      (field $y i32))
  (global (ref $s)
    (struct.new (i32.const 42)
      (i32.const 42)))
  (func $get-x (param (ref $s))(result i32)
    (struct.get $s $x (local.get 0))))

New instructions to access GC data
A concrete example

(module
  (type $s (struct (field $x i32)
                (field $y i32)))
  (global (ref $s)
    (struct.new (i32.const 42)
                (i32.const 42)))
  (func $get-x (param (ref $s)) (result i32)
    (struct.get $s $x (local.get 0))))

Ref types let you reference GC values
Progress in JSC
We’re implementing Wasm GC for JSC

- WIP implementation underway
- Most features already implemented
- Should get to shippable state in a few months
What's implemented & what's not

✅ Structs/arrays
✅ i31 references
✅ Type hierarchy
✅ Subtyping
✅ Recursive types
✅ Type casts
✅ Locals with non-nullable types
✅ Table initializers

〇 JS API support
〇 Bulk array operations
〇 Misc other missing instructions

Plus more testing and optimization!
To track our progress: ☂
https://bugs.webkit.org/show_bug.cgi?id=247394
The takeaway
This is an exciting time for Wasm!

- Other browsers have shipped or will ship Wasm GC soon (the proposal is at phase 4)
- Once JSC also ships, Wasm GC supported in all major browser engines
- Developers can then target Wasm with Java, OCaml, and many more languages to come

Future Wasm will likely build further on GC
Q & A

Thanks for listening!