Debugging WebAssembly with modern tools

PLCT Intern Report

Liang Bin

2021.1.27

What is WebAssembly?

WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine.

WebAssembly is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.

What we need when debugging

Emscripten from: https://github.com/emscripten-core/emsdk

Serve from: https://github.com/vercel/serve

Chrome Canary from: https://www.google.com/chrome/canary/

A helper extension from: https://goo.gle/wasm-debugging-extension

A simple C example

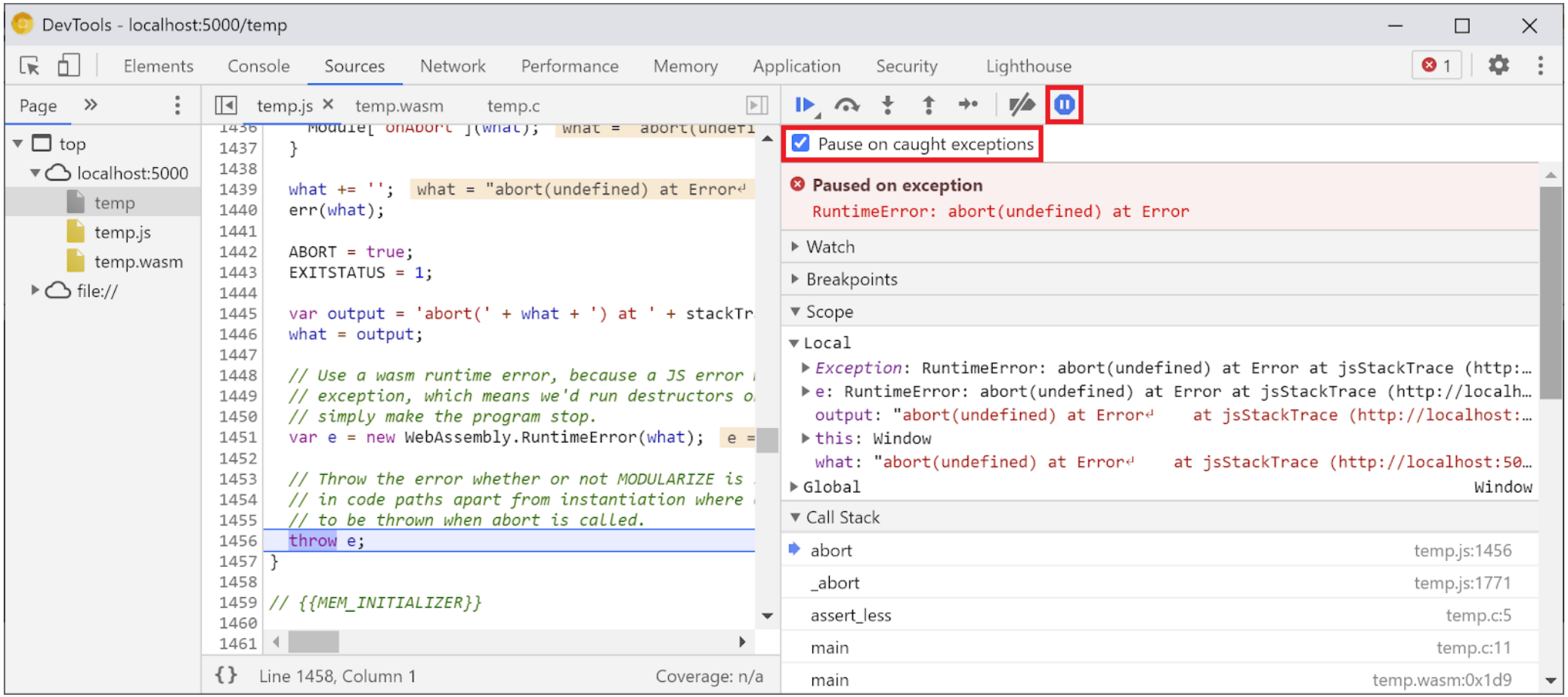
```
#include <stdlib.h>

void assert_less(int x, int y) {
  if (x >= y) {
    abort();
  }
}

int main() {
  assert_less(10, 20);
  assert_less(30, 20);
}
```

use:emcc -g temp.c -o temp.html

enable Pause on exceptions (⏸ icon), then check Pause on caught exceptions and reload the page.



See a Call Stack view and navigate to the original C line that invoked abort

DevTools - localhost:5000/temp

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse

Page

- top
  - localhost:5000
    - temp
    - temp.js
    - temp.wasm
  - file://

temp.js | temp.wasm | temp.c

```
1  #include <stdlib.h>
2
3  void assert_less(int x, int y) {
4    if (x >= y) {
5      abort();
6    }
7  }
8
9  int main() {
10   assert_less(10, 20);
11   assert_less(30, 20);
12 }
13
```

{} Line 5, Column 5 (provided via debug info by temp.wasm) Covera

Pause on caught exceptions

Paused on exception

RuntimeError: abort(undefined) at Error

Watch

Breakpoints

Scope

Parameter

x: 30

y: 20

Call Stack

abort	temp.js:1456
_abort	temp.js:1771
assert_less	temp.c:5
main	temp.c:11
main	temp.wasm:0x1d9
(anonymous)	temp.js:1525
callMain	temp.js:2046
doRun	temp.js:2106

A complicated C++ example

d: > sdl2 > mandelbrot.cc

```cpp
#include <SDL2/SDL.h>
#include <complex>

int main() {
  // Init SDL.
  int width = 600, height = 600;
  SDL_Init(SDL_INIT_VIDEO);
  SDL_Window* window;
  SDL_Renderer* renderer;
  SDL_CreateWindowAndRenderer(width, height, SDL_WINDOW_OPENGL, &window,
                              &renderer);

  // Generate a palette with random colors.
  enum { MAX_ITER_COUNT = 256 };
  SDL_Color palette[MAX_ITER_COUNT];
  srand(time(0));
  for (int i = 0; i < MAX_ITER_COUNT; ++i) {
    palette[i] = {
        .r = (uint8_t)rand(),
        .g = (uint8_t)rand(),
        .b = (uint8_t)rand(),
        .a = 255,
    };
  }

  // Calculate and draw the Mandelbrot set.
  std::complex<double> center(0.5, 0.5);
  double scale = 4.0;
  for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
      std::complex<double> point((double)x / width, (double)y / height);
      std::complex<double> c = (point - center) * scale;
      std::complex<double> z(0, 0);
      int i = 0;
      for (; i < MAX_ITER_COUNT - 1; i++) {
        z = z * z + c;
        if (abs(z) > 2.0)
          break;
      }
      SDL_Color color = palette[i];
      SDL_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a);
      SDL_RenderDrawPoint(renderer, x, y);
    }
  }

  // Render everything we've drawn to the canvas.
  SDL_RenderPresent(renderer);

  // SDL_Quit();
}
```

use: emcc -g mandelbrot.cc -s USE_SDL = 2 -s ALLOW_MEMORY_GROWTH -o mandelbrot.html

Set some breakpoint

DevTools - localhost:5000/mandelbrot

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse

Page

- top
  - localhost:5000
    - mandelbrot
    - mandelbrot.js
    - mandelbrot.wasm
  - file://

mandelbrot.cc

```
1  #include <SDL2/SDL.h>
2  #include <complex>
3
4  int main() {
5    // Init SDL.
6    int width = 600, height = 600;
7    SDL_Init(SDL_INIT_VIDEO);
8    SDL_Window* window;
9    SDL_Renderer* renderer;
10   SDL_CreateWindowAndRenderer(width, height, SDL_WINDOW_OPENGL, &window,
11                               &renderer);
12
13   // Generate a palette with random colours.
14   enum { MAX_ITER_COUNT = 256 };
15   SDL_Color palette[MAX_ITER_COUNT];
16   srand(time(0));
17   for (int i = 0; i < MAX_ITER_COUNT; ++i) {
18     palette[i] = {
19         .r = (uint8_t)rand(),
20         .g = (uint8_t)rand(),
21         .b = (uint8_t)rand(),
22         .a = 255,
```

{} Line 7, Column 3 (provided via debug info by mandelbrot.wasm) Coverage: n/a

Pause on caught exceptions

Paused on breakpoint

Watch

Breakpoints

mandelbrot.cc:7
SDL_Init(SDL_INIT_VIDEO);

Scope

Local

- center: std::complex<double>
- height: 600
- palette: SDL_Color [256]
- renderer: SDL_Renderer *
- scale: 0
- width: 600
- window: SDL_Window *

Call Stack

main mandelbrot.cc:7

DevTools - localhost:5000/mandelbrot

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse

Page

- top
  - localhost:5000
    - mandelbrot
    - mandelbrot.js
    - mandelbrot.wasm
  - file://

mandelbrot.cc

```
26  // Calculate and draw the Mandelbrot set.
27  std::complex<double> center(0.5, 0.5);
28  double scale = 4.0;
29  for (int y = 0; y < height; y++) {
30    for (int x = 0; x < width; x++) {
31      std::complex<double> point((double)x / width, (double)y / hei
32      std::complex<double> c = (point - center) * scale;
33      std::complex<double> z(0, 0);
34      int i = 0;
35      for (; i < MAX_ITER_COUNT - 1; i++) {
36        z = z * z + c;
37        if (abs(z) > 2.0)
38          break;
39      }
40      SDL_Color color = palette[i];
41      SDL_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a);
42      SDL_RenderDrawPoint(renderer, x, y);
43    }
44  }
45
46  // Render everything we've drawn to the canvas.
47
```

{} Line 31, Column 42 (provided via debug info by mandelbrot.wasm) Coverage: n/a

Pause on caught exceptions

Scope

- Local
  - c: std::complex<double>
  - center: std::complex<double>
    - __im_: 0.5
    - __re_: 0.5
  - color: SDL_Color
  - height: 600
  - i: 0
  - palette: SDL_Color [256]
    - 0: SDL_Color
      - a: 255
      - b: 3
      - g: 135
      - r: 237
    - 1: SDL_Color
    - 2: SDL_Color
    - 3: SDL_Color
    - 4: SDL_Color

DevTools - localhost:5000/mandelbrot

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse

Page

- top
  - localhost:5000
    - mandelbrot
    - mandelbrot.js
    - mandelbrot.wasm
  - file://

mandelbrot.cc × | mandelbrot.js

```
26  // Calculate and draw the Mandelbrot set.
27  std::complex<double> center(0.5, 0.5);
28  double scale = 4.0;
29  for (int y = 0; y < height; y++) {
30    for (int x = 0; x < width; x++) {
31      std::complex<double> point((double)x / width, (double)y / hei
32      std::complex<double> c = (point - center) * scale;
33      std::complex<double> z(0, 0);
34      int i = 0;
35      for (; i < MAX_ITER_COUNT - 1; i++) {
36        z = z * z + c;
37        if (abs(z) > 2.0)
38          break;
39      }
40      SDL_Color color = palette[i];
41      SDL_SetRenderDrawColor(renderer, color.r, color.g, color.b, color.a);
42      SDL_RenderDrawPoint(renderer, x, y);
43    }
44  }
45
46  // Render everything we've drawn to the canvas.
47
```

3

{} Line 31, Column 42 (provided via debug info by mandelbrot.wasm) Coverage: n/a

Pause on caught exceptions

Paused on breakpoint

Watch

x: 3

y: 0

Breakpoints

mandelbrot.cc:7
SDL_Init(SDL_INIT_VIDEO);

mandelbrot.cc:31
std::complex<double> point((double)x…

Scope

Local

- c: std::complex<double>
- center: std::complex<double>
- color: SDL_Color
- height: 600
- i: 0

Raw WebAssembly debugging

DevTools - localhost:5501/mandelbrot.html

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse

Page

- top
  - localhost:5501
    - mandelbrot.html
    - mandelbrot.js
    - mandelbrot.wasm
  - file://

mandelbrot.js | mandelbrot.wasm | mandelbrot.cc

```
0x010fc8   )
0x010fc9   (func $SDL_SetRenderDrawColor (;444;) (param $var0 i32) (param $var1 i
0x010fcb     block $label1
0x010fcd       block $label0
0x010fcf         local.get $var0
0x010fd1         i32.eqz
0x010fd2         br_if $label0
0x010fd4         local.get $var0
0x010fd6         i32.load
0x010fd9         i32.const 64641
0x010fdd         i32.eq
0x010fde         br_if $label1
0x010fe0       end $label0
0x010fe1       i32.const 8833
0x010fe5       i32.const 0
0x010fe7       call $SDL_SetError
0x010fea       drop
0x010feb       i32.const -1
0x010fed       return
0x010fee     end $label1
0x010fef     local.get $var0
```

Line 27107, Column 1 | Coverage: n/a

Pause on caught exceptions

Scope

Module

- env.me
- global
- instan

Local

- var0:
- var1:
- var2:
- var3:
- var4:

Copy property path
Copy object
Add property path to watch
Inspect memory
Store object as global variable

Stack

Call Stack

SDL_SetRenderDrawColor	mandelbrot.wasm:0x10fcf
main	mandelbrot.cc:41
main	mandelbrot.wasm:0x3ef2

Some improvements

New name generation scheme:

They're generating names similarly to other disassembly tools, by using hints from the WebAssembly name section, import/export paths and, finally, if everything else fails, generating them based on the type and the index of the item .

Memory inspection:

They added a new feature to help with this, too: a linear memory inspector. You can inspect the WebAssembly memory in hexadecimal and ASCII views, navigate to specific addresses

Some extensions

The speed of your code:

Use the DevTools Performance panel to measure speed of your code.

Building and debugging on different machines:

They have implemented a path mapping functionality in the C/C++ extension options. You can use it to remap arbitrary paths and help the DevTools locate sources.

Debugging optimized builds:

Current example:emcc -g temp.c -o temp.html \-O3 -fno-inline

Separating the debug information:

emcc -g temp.c -o temp.html \ -gseparate-dwarf=temp.debug.wasm

Conclusion

Plenty of tools for debugging

Raw WebAssembly debugging is still not easy

To be continued…

Thanks

Sources from: https://developers.google.com/web/updates/2020/12/webassembly
https://emscripten.org/index.html