# Emscripten demo summary

# Purpose:

Try to use Emscripten tool to convert a C++ library to Javascript and run it in web browser.

All are based on Mac

# Emscripten install:

Emscripten is a tool for converting C++ to Javascript.

1. Emscripten main website: <http://kripken.github.io/emscripten-site/>
2. Download it from <http://kripken.github.io/emscripten-site/docs/getting_started/downloads.html>
3. Open a command window and type below to install:

**# Fetch the latest registry of available tools.**

.**/**emsdk update

**# Download and install the latest SDK tools.**

.**/**emsdk install latest

**# Make the "latest" SDK "active"**

.**/**emsdk activate latest

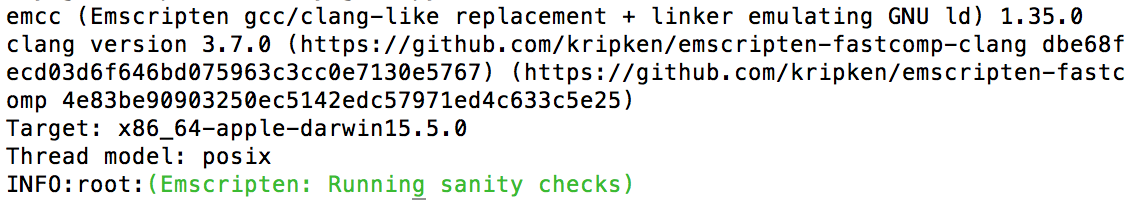
**# Linux/Mac OS X only: Set the current Emscripten path**

source .**/**emsdk\_env.sh

1. Switch to your working directory and add Emscripten tool path

export PATH=$PATH:…/emscripten/1.35.0

1. Type emcc –v and you’ll see below information



It means emscripten installed successfully.

In latest OSX EI Captian it may show

Env: python 2: No such file or directory

Type python emcc –v and it will be OK

# Acquire demo code:

1. Get code: git clone https://github.com/cnyzgkn/emscripten\_demo
2. Go to \Source you can find several folders

[closure\_compiler](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/closure_compiler): a tool for compressing Javascript file

[embind\_demo](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/embind_demo): a demo for testing what types of argument does Emscripten support

[webDemo\_EmBind](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSDemo_EmBind): a demo of using EmBind tool for Web

[webDemo\_cwrap](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSDemo_cwrap): a demo of using cwrap method for Web (not used anymore)

[webLib](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSLib): a demo of generating libso using emscripten tool

[webTest](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSTest): a demo for performance test using lib.so

[webTestUsingSrc](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSTestUsingSrc): a demo for performance test using code

# cwrap vs Embind:

1. Emscripten supports 2 method internally: cwrap and embin for Javascript <-> C++ conversion
2. cwrap and Embind has much in common

1): need lib.js generated before hand

2): need to write C++ wrapper code

3): need to release memory mannullay

1. However they still have several differences:

|  |  |  |  |
| --- | --- | --- | --- |
|  | summary | supported data type | Need extra JS wrapper |
| cwrap | Wrap existing C++ functions | Internal type(int, double), pointer | yes |
| Embind | Bind C++ functions and class | Internal type, user-defined type, pointer, reference | no |

So we choose Embind finally.

# Run Demo using Embind:

1. Go to webLib folder
2. Type emmake make [-gx] to generate lib.js

You can choose different debug levels. Higher debug level you choose, more readable lib.js is.

For example, use –g4, you’ll see the full function name.

Use –g0, you’ll see encrypted function name.

1. Go to [webDemo\_EmBind](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSDemo_EmBind) folder.
2. Find Demo\_1.js and Demo\_2.js. These are 2 simple cases that writing Javascript API calling functions.
3. Find runDemoEmBind.html. It’s a html file including 2 Javascript files above.
4. Open it with a browser and you can see the result.

# Capabilities and limitations of Embind:

1. Emscripten is for converting C++ source code to Javascript source code so it can’t handle complex characters of C++, like reference or enum. Unfortunately its official manual doesn’t mention it.

In sub-folder [embind\_demo](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/embind_demo) I did some tests and have lists below:

|  |  |  |
| --- | --- | --- |
| Data Type | As function return value | As function argument |
| internal type(int, double) | support | support |
| User defined type | support | support |
| Enum | Not support | support |
| Pointer to internal type | support | support |
| Pointer to user defined type | support | support |
| Reference to internal type | support | support |
| Reference to user defined type | support | support |
| Multiple pointer | Not support | Not support |
| Reference to pointer | Not support | Not support |

1. It means some APIs needs to be re-written, for example, for the API

Status Sys::create3dDistPtPt( ConHandle\*& h,

…)

Because Emscripten doesn’t support ConHandle\*&, we have to convert it to

ConHandle\*   Sys::create3dDistPtPt(

…)

{

   ConHandle\* h = NULL;

    sys->create3dDistPtPt(h, b1, b2, pt1, pt2, extG1, extG2, v, mode, var);

    return h;

}

1. Another method is to implement a new class JS\_System which contains a System pointer as below. We plan to implement it later.

    class JS\_System

{ System\* sys; }

Implement related wrapper functions as below and convert them to Javascript

    ConHandle\*   JS\_System::create3dDistPtPt(

)

{

   ConHandle\* h = NULL;

    sys->create3dDistPtPt(h, b1, b2, pt1, pt2, extG1, extG2, v, mode, var);

    return h;

}

1. Enum as return value seems has a bug in Emscripten. It always return the first element of Emum(default value == 0).

# Run Performance test:

1. Write 3 cases that call API

1). DistPtPt3d: simple PtPt constraint

2). SymmPtPtTol: moderate symmetric constraint

3). Ellipse: Complex Variable ellipse + symmetric constraint

1. Convert them to Javascript, open it with different browsers
2. Build and run them directly
3. Compare results.

All codes are in [webTestUsingSrc](https://git.autodesk.com/modeling-components/Variational-Constraint-Solver/tree/webVCS/Tools/WebVCS/Source/webVCSTestUsingSrc)

# Performance test:

1. Different web browsers has different performance
2. Using -s ALLOW\_MEMORY\_GROWTH=1 flag can increase the stability but lower down the performance.
3. Comparing pure Javascript demo and Emscripten demo, Emscripten still has better performance than Javascript.

Table 1: Performance in different web browser and native code

|  |  |
| --- | --- |
| Type | Time Cost |
| Pure C++ native | 1 |
| Emscripten + Firefox | 2 |
| Emscripten + Edge | 10 |
| Emscripten + Safari | 20 |
| Emscripten + Chrome | 30+ |
| Emscripten + IE | 30+ |
| Emscripten + Opera | 30+ |
| Pure Javascript code | 150+ |

Table 2: Performance using different optimization level

**Using optimization level == 3:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case | Descripten | Native C++ solve time using gcc (ms) | Emscripten solve time (Chrome) | Emscripten solve time (IE) | Emscripten solve time (FireFox) |
| DistPtPt3d | simplest PtPt constriant | **1** | **33** | **10** | **2** |
| SymmPtPtTol | More complex symmetric constraint | **5** | **262** | **159** | **5** |
| Ellipse | Very complex constraints:  Variable ellipse + symmetric | **20** | **394** | **289** | **21** |

**Using optimization level ==0:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case | Descripten | Native C++ solve time using gcc (ms) | Emscripten solve time (Chrome) | Emscripten solve time (IE) | Emscripten solve time (FireFox) |
| DistPtPt3d | simplest PtPt constriant | **1** | **32** | **23** | **2** |
| SymmPtPtTol | More complex symmetric constraint | **5** | **274** | **194** | **5** |
| Ellipse | Very complex constraints:  Variable ellipse + symmetric | **20** | **391** | **312** | **22** |

# Hints:

1. I was thinking wrapping data into structure and passing address number of these structures in Javascript, but haven’t tried it yet.

2. WebAssembly (<https://medium.com/javascript-scene/what-is-webassembly-the-dawn-of-a-new-era-61256ec5a8f6#.pmt7pa5jn>) seems a new approach between C++ and Javascript.

# Useful Links:

1. Emscripten introduction: http://kripken.github.io/emscripten-site/docs/getting\_started/index.html
2. Emscripten source code: <https://github.com/kripken/emscripten>
3. Forum: [https://groups.google.com/forum/#!topic/emscripten-discuss/](https://groups.google.com/forum/#!topic/emscripten-discuss/PwCfHyKELtk)