

PDF - SOMATIVA

Resumo do Emscripten:

Emscripten é um compilador que converte código escrito em C e C++ para JavaScript, permitindo que aplicações nativas sejam executadas em navegadores da web. Desenvolvido inicialmente por Alon Zakai, o Emscripten utiliza a LLVM (Low-Level Virtual Machine) como backend, facilitando a transição de aplicativos de desktop para a web sem reescrever o código em linguagens específicas da web. Esse processo de compilação gera WebAssembly (Wasm) ou asm.js, uma versão otimizada de JavaScript que oferece desempenho próximo ao código nativo, possibilitando a execução de programas complexos, como jogos e simulações, diretamente no navegador.

Uma das principais características do Emscripten é sua capacidade de permitir que desenvolvedores aproveitem bibliotecas e frameworks existentes em C e C++. Isso é particularmente útil para projetos que exigem um desempenho elevado, como jogos, aplicações de gráficos 3D e simuladores, que tradicionalmente são escritos nessas linguagens. O Emscripten oferece suporte a APIs da web, como OpenGL, através de WebGL, e permite a utilização de chamadas de sistema POSIX, facilitando a adaptação de projetos complexos para o ambiente web.

Além disso, o Emscripten inclui um sistema de gerenciamento de pacotes, chamado Emscripten Package Manager (emcmake), que simplifica a integração de bibliotecas de terceiros no projeto. Através do uso de Emscripten, os desenvolvedores podem criar aplicações multiplataforma, reduzindo o tempo e os custos de desenvolvimento, pois um único código-base pode ser executado em diferentes sistemas operacionais e navegadores.

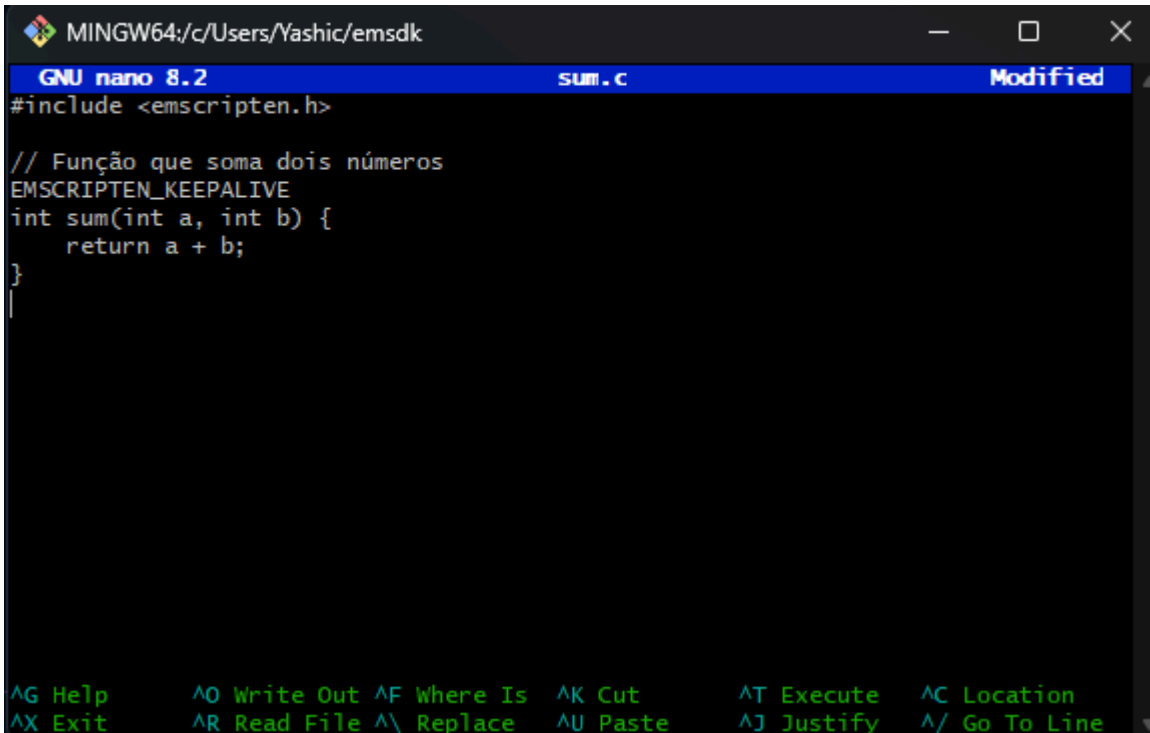
As aplicações do Emscripten são diversas. Desde jogos clássicos, como "Doom" e "Quake", que foram portados para o ambiente web, até ferramentas científicas e simuladores que requerem cálculos complexos. Além disso, a tecnologia também é utilizada em projetos educacionais e de visualização, permitindo a criação de experiências interativas que funcionam diretamente no navegador, eliminando a necessidade de instalação de software adicional.

O Emscripten é uma solução poderosa para a modernização de aplicações legadas, permitindo que projetos mais antigos sejam acessíveis em plataformas contemporâneas. A comunidade em torno do Emscripten é ativa, com atualizações constantes e melhorias no

desempenho e na compatibilidade com as mais recentes tecnologias da web, garantindo que desenvolvedores possam aproveitar ao máximo as capacidades da web moderna.

Em resumo, o Emscripten representa um marco significativo na interseção entre desenvolvimento nativo e web, oferecendo uma ponte eficaz que permite a execução de aplicações robustas e de alto desempenho em ambientes baseados em navegador.

Demonstração Prática:



A screenshot of a MINGW64 terminal window. The title bar shows the path 'MINGW64:/c/Users/Yashic/emsdk'. The editor window is titled 'GNU nano 8.2' and 'sum.c' is open. The code in the editor is as follows:

```
#include <emscripten.h>

// Função que soma dois números
EMSCRIPTEN_KEEPALIVE
int sum(int a, int b) {
    return a + b;
}
```

The bottom status bar of the nano editor displays the following shortcuts:

^G Help	^O Write Out	^F Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

```
MINGW64:/c/Users/Yashic/emsdk
$ ^C

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$ nano sum.c

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$ ls
LICENSE                emsdk_env.fish
README.md              emsdk_env.ps1
SECURITY.md            emsdk_env.sh
bazel/                 emsdk_manifest.json
docker/                java/
downloads/             legacy-binaryen-tags.txt
emcmdprompt.bat        legacy-emscripten-tags.txt
emscripten-releases-tags.json  llvm-tags-64bit.txt
emsdk*                 node/
emsdk.bat              python/
emsdk.ps1              scripts/
emsdk.py*              sum.c
emsdk_env.bat          test/
emsdk_env.csh          upstream/

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$
```

```
MINGW64:/c/Users/Yashic/emsdk
downloads/      legacy-binaryen-tags.txt
emcmdprompt.bat legacy-emscripTEN-tags.txt
emscripten-releases-tags.json llvm-tags-64bit.txt
emsdk*          node/
emsdk.bat       python/
emsdk.ps1       scripts/
emsdk.py*       sum.c
emsdk_env.bat   test/
emsdk_env.csh   upstream/

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$ emcc sum.c -o sum.js -s EXPORTED_FUNCTIONS=['["_sum"]'] -s EXTRA_EXPORTED_RUNTIME_METHODS=['["cwrap"]']
emcc: warning: EXTRA_EXPORTED_RUNTIME_METHODS is deprecated (please use EXPORTED_RUNTIME_METHODS instead). Please open a bug if you have a continuing need for this setting [-Wdeprecated]
cache:INFO: generating system asset: symbol_lists/9bb73d1d74bc0b12084402a515fb963d200363e7.json... (this will be cached in "C:\Users\Yashic\emsdk\upstream\emscripten\cache\symbol_lists\9bb73d1d74bc0b12084402a515fb963d200363e7.json" for subsequent builds)
cache:INFO: - ok

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$ |
```

```
MINGW64:/c/Users/Yashic/emsdk
GNU nano 8.2 index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>WebAssembly Sum Example</title>
</head>
<body>
  <h1>WebAssembly Sum Example</h1>
  <input type="number" id="a" placeholder="Enter first number">
  <input type="number" id="b" placeholder="Enter second number">
  <button id="calculate">Calculate Sum</button>
  <h2 id="result"></h2>

  <script src="sum.js"></script>
  <script>
    const calculateButton = document.getElementById('calculate');
    const resultElement = document.getElementById('result');

    Module['onRuntimeInitialized'] = () => {
      const sum = Module.cwrap('sum', 'number', ['number', 'number']);

      calculateButton.addEventListener('click', () => {
        const a = parseInt(document.getElementById('a').value);
        const b = parseInt(document.getElementById('b').value);
        const result = sum(a, b);
        resultElement.textContent = `Result: ${result}`;
      });
    };
  </script>
</body>
</html>

^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

WebAssembly Sum Example

2
2
Calculate Sum

DevTools is now available in Portuguese!

Always match Chrome's language

Switch DevTools to Portuguese

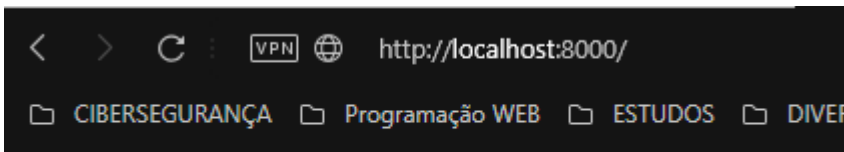
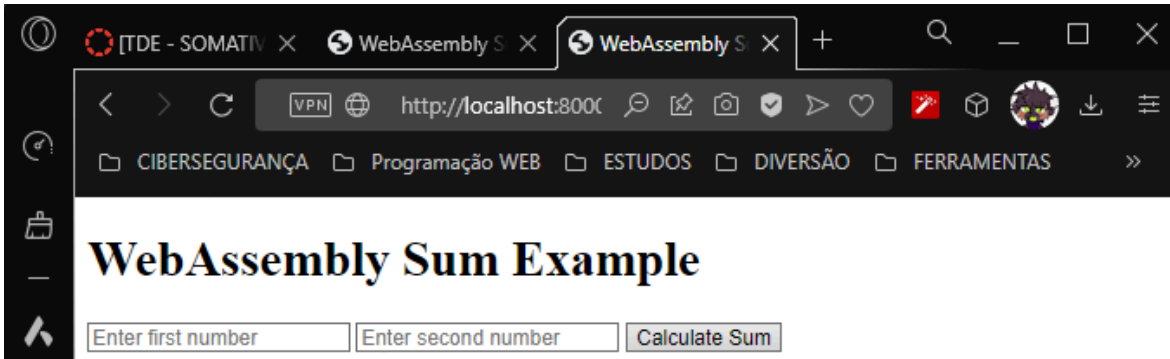
Don't show again

Elements Console >> 6

top Filter

Default levels No Issues

- Access to XMLHttpRequest at 'index.html:1' from origin 'null' has been blocked by CORS policy: Cross origin requests are only supported for protocol schemes: http, isolated-app, ipns, https, chrome-untrusted, ipfs, data, chrome-extension, chrome.
- failed to asynchronously prepare `sum.js:655`
wasm: both async and sync fetching of the wasm failed
- warning: Loading from a file URI (`sum.js:659`
`file:///C:/Users/Yashic/emSDK/sum.wasm`) is not supported in most browsers. See https://emscripten.org/docs/getting_started/FAQ...
- Aborted(both async and sync `sum.js:539`
fetching of the wasm failed)
- Failed to load resource: `sum.wasm:1`
net::ERR_FAILED
- Uncaught (in promise) `sum.js:557`
RuntimeError: Aborted(both async and sync
fetching of the wasm failed)
at abort (`sum.js:557:11`)
at `sum.js:661:5`



WebAssembly Sum Example

Result: 6

```
C:\Users\Yashic\AppData\Local\Microsoft\WindowsApps\PythonSoftware
qbz5n2kfra8p0\python.exe: No module named SimpleHTTPServer

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$ python --version
Python 3.11.9

Yashic@YashiCompiuter MINGW64 ~/emsdk (main)
$ python -m http.server 8000
:::1 - - [16/Oct/2024 13:57:37] "GET / HTTP/1.1" 200 -
:::1 - - [16/Oct/2024 13:57:37] "GET /sum.js HTTP/1.1" 200 -
:::1 - - [16/Oct/2024 13:57:37] "GET /sum.wasm HTTP/1.1" 200 -
:::1 - - [16/Oct/2024 13:57:37] code 404, message File not found
:::1 - - [16/Oct/2024 13:57:37] "GET /favicon.ico HTTP/1.1" 404 -
```