WebAssembly

Introducción básica a Wasm.

Presentado por Gonzalo Nández

(**infogon** en las redes sociales) 12 / 10 / 2019

Historia (corta)

- Flash y Applets Java.

Historia (corta)

- Flash y Applets Java.
- JavaScript: Rey del frontend.

Historia (corta)

- Flash y Applets Java.
- JavaScript: Rey del frontend.
- Todo migra a la web.

Historia (corta)

- Flash y Applets Java.
- JavaScript: Rey del frontend.
- Todo migra a la web.
- Google y el Cliente Nativo.

Historia (corta)

- Flash y Applets Java.
- JavaScript: Rey del frontend.
- Todo migra a la web.
- Google y el Cliente Nativo.
- Mozilla y asm.js

Historia (corta)

- Flash y Applets Java.
- JavaScript: Rey del frontend.
- Todo migra a la web.
- Google y el Cliente Nativo.
- Mozilla y asm.js
- WW3C y WebAssembly

WebAssembly

Definición oficial:

Wasm es un formato de instrucciones binario para una máquina virtual basada en una pila (stack). Wasm esta diseñado como un destino portátil para la compilación de lenguajes de alto nivel como C/ C++/ Rust, permitiendo la implementación en la web para aplicaciones de cliente y servidor.

Creando...

Lenguajes y herramientas para crear WebAssembly.

- C / C++ → EmScripten (Sera el que usaremos para las pruebas)
- C # → Blazor
- Haxe → WebIDL
- Java → TeaVM o Bytecoder
- Kotlin → TeaVM
- TypeScript → AssemblyScript
- Rust → wasm-pack (paquete cargo)

Continuación

- Python → Pyodide
- Go → ahora soporta WebAssembly.
- D → LDC (LLVM compiler).
- PHP → Experimental, pero con un prototipo en funcionamiento
- Perl → WebPerl
- Scala → Emscriptenr
- Ruby -> proyecto run.rb
- Swift → SwiftWasm
- Y seguramente alguno se me quedo en el tintero. Para mas referencias: https://github.com/appcypher/awesome-wasm-langs

Ejemplos de usos.

- VLC → https://code.videolan.org/jbk/vlc.js
- Blender → https://www.blend4web.com/en/
- OpenCV → https://docs.opencv.org/master/d4/da1/tutorial_js_setup.html
- Unity (Juegos) → https://blogs.unity3d.com/es/2018/08/15/webassembly-is-here/
- UnRealEngine → https://docs.unrealengine.com/en-US/Platforms/HTML5/GettingStarted/index.html
- Y muchos proyectos mas sobre videos, sonidos, OpenGL, etc.
- Mas ejemplos: https://github.com/emscripten-core/emscripten/wiki/Porting-Examples-and-Demos

No solo web

Aunque la información que se va a encontrar en Internet se centra en su mayoría en el uso de la web de esta tecnología existen proyectos que van mucho mas allá Aquí algunos ejemplos:

- Binarios portables a cualquier OS.
- Nebulet es un microkernel que ejecuta módulos WebAssembly
- Seguridad en los contratos inteligentes (BlockChain) véase: https://wiki.parity.io/WebAssembly-Design
- Librerías comunes a diferentes lenguajes de programación

Casos de uso.
Dentro del navegador

- Mejor ejecución para lenguajes y kits de herramientas que actualmente están compilados en la Web (C/C++, GWT, ...).
- Juegos
- Aplicaciones peer-to-peer (juegos, edición colaborativa, descentralizadas y centralizadas).
- Aplicaciones de música (streaming, caching).
- Reconocimiento de imágenes. Edición de imágenes/vídeo.
- Aumento de vídeo en directo (por ejemplo, poner sombreros en la cabeza de las personas).
- RV y realidad aumentada (latencia muy baja).
- Aplicaciones CAD.
- Visualización y simulación científica.
- Software educativo interactivo y artículos de noticias.
-

Continuación

- Simulación / emulación de plataformas (ARC, DOSBox, QEMU, MAME,...).
- Intérpretes de idiomas y máquinas virtuales.
- Entorno de espacio de usuario POSIX, que permite la transferencia de las aplicaciones POSIX existentes.
- Herramientas para desarrolladores (editores, compiladores, depuradores,...).
- Escritorio remoto.
- VPN.
- Encriptación.
- Servidor web local.
- Usuarios comunes de NPAPI, dentro del modelo de seguridad de la web y APIs.
- Cliente graso para aplicaciones empresariales (p. ej. bases de datos).

Continuación

- **Fuera del navegador**

- Servicio de distribución de juegos (portátil y seguro).
- Cálculo de código no fiable en el lado del servidor.
- Aplicación del lado del servidor.
- Aplicaciones nativas híbridas en dispositivos móviles.
- Computaciones simétricas a través de múltiples nodos

Archivos .wat

WebAssembly puede ser escrito directamente sin lenguaje de origen. De la forma y anotaciones solo dejare la referencia de la web y un ejemplo de código

El código en C:

```
int main() {
 return 42;
}
```

Escrito en un archivo .wat:

```
(module
 (table 0 anyfunc)
 (memory $0 1)
 (export "memory" (memory $0))
 (export "main" (func $main))
 (func $main (; 0 ;) (result i32)
  (i32.const 42)
 )
)
```

Mas información: https://developer.mozilla.org/en-US/docs/WebAssembly/Understanding_the_text_format

Editor Online: https://wasdk.github.io/WasmFiddle/

EmScripten

EmScripten es un programa informático, un tipo de compilador denominado compilador Source- to-source o transcompilador. Puede procesar bytecode de LLVM, normalmente creado al compilar código C o C++. Este nos devuelve como salida un archivo en el lenguaje de programación JavaScript que puede procesarse en navegadores web. Es compatible con el estándar de la API de desarrollo de C/C++ como STL, SDL o incluso OpenGL.

- El fundador de este proyecto es **Alon Zakai**.

Codificando

Estos ejemplos serán usando las siguientes herramientas:

- EmScripten
- C / C++ con Clion de JetBrains
- HTML / JS con WebStorm de JetBrains
- Kubuntu 18.04

Instalar y configurar EmScripten

Pasos para instalar EmScripten.

- git clone https://github.com/emscripten-core/emsdk.git
- cd emsdk
- ./emsdk install latest
- ./emsdk activate latest
- source ./emsdk_env.sh
- emcc -v

Para actualizar a la ultima versión:

- cd emsdk
- git pull
- ./emsdk install latest
- ./emsdk activate latest
- source ./emsdk_env.sh

Configurar cmake

- EMSDK=path_to_emsdk/emsdk
- EM_CONFIG=path_to_user_home/.eemscripten
- EMSDK_NODE=path_to_emsdk/emsdk/node/12.9.1_64bit/bin/node

Proyectos relacionados

- Wasmer.io

  Es un proyecto para poder crear Wasm desde varios lenguajes con un único compilador.

  En este momento Wasmer soporta los siguientes lenguajes:

Languages

Wasmer runtime can be used as a library embedded in different languages, so you can use WebAssembly anywhere:

	Language	Author(s)	Maintenance	Release	Stars
	Rust	Wasmer	actively developed	crates.io v0.8.0	stars 4.6k
	C/C++	Wasmer	actively developed	release v0.8.0	stars 4.6k
	Python	Wasmer	actively developed	pypi v0.3.0	stars 670
	Go	Wasmer	actively developed	release v0.2.0	stars 719
	PHP	Wasmer	actively developed	release v0.5.0	stars 545
	Ruby	Wasmer	actively developed	gem v0.3.0	stars 196
	Postgres	Wasmer	actively developed	release v0.1.0	stars 114
	C#/.Net	Miguel de Icaza	actively developed	nuget v0.6.0	stars 88
	R	Dirk Schumacher	actively developed		stars 45
	Swift	Mark Malström	passively maintained		stars 6

La url del proyecto es: https://wasmer.io/

Aaron Turner

Creador de WasmByExample, WasmBoy, y parte del equipo de AssemblyScript

- **WasmByExample**: Es un sitio donde comenzar a aprender Wasm que esta en pleno desarrollo y acepta colaboraciones. Con códigos de ejemplos en Rust, C++ y AssemblyScript (por ahora). *https://wasmbyexample.dev/*

- **WasmBoy**: Librería de emuladores de Gameboy escrita en Web Assembly usando AssemblyScript, Debugger/Shell en Preact. *https://wasmboy.app/*

Medium: https://medium.com/@torch2424

Github: https://github.com/torch2424

YouTube: https://www.youtube.com/watch?v=ZlL1nduatZQ

Lin Clark

Lin Clark es una referencia de WebAssembly y Rust. Con varios artículos publicados en www.hacks.mozilla.org

Lo que la destaca es que explica todo a través de caricaturas muy creativas permitiendo tener una visualización del problema, el proceso y la solución

https://hacks.mozilla.org/author/lclarkmozilla-com/

Web de referencia para este material

Editor Online que convierte código C a Wat
https://wasdk.github.io/WasmFiddle/

Sean Voisen » Renderizacion de imágenes con Emscripten, WASM y OpenGL
https://sean.voisen.org/blog/2018/03/rendering-images-emscripten-wasm/

Referencia de la API — Emscripten documentación
https://emscripten.org/docs/api_reference/index.html

Ejemplos y guía de compilación de C++ con Emscripten para web
https://medium.com/@tdeniffel/pragmatic-compiling-from-c-to-webassembly-a-guide-a496cc5954b8

Un poco de historia y el caso de uso de pspdfkit
https://pspdfkit.com/blog/2017/webassembly-a-new-hope/

Un acercamiento a WebAssembly en español.
https://pablomagaz.com/blog/empezando-con-webassembly

Un sistema de partículas para comparar el rendimiento de js y Wasm.
https://adndevblog.typepad.com/cloud_and_mobile/2016/07/boost-your-web-application-with-c-emscripten-asmjs-web-assembly-.html

Otro articulo en español para comenzar con WebAssembly
https://programacion.net/articulo/comenzando_con_webassembly_1835

Continua

Historia, explicaciones de conceptos y posibles casos de uso. En español.
http://dotnetuy.com/blog/2019/03/24/que-es-webassembly/

Un articulo mas amplio sobre las diferencias de los Applets Java, Flash y WebAssembly
https://words.steveklabnik.com/is-webassembly-the-return-of-java-applets-flash

cheerp es un compilador de C++ para la web que permite el manejo del DOM de html. Licencia Free limitada y licencia comercial.
https://www.leaningtech.com/cheerp/docs/

VLC para la Web
https://code.videolan.org/jbk/vlc.js

Blender en la web a través de WebAssembly. Licencia Free y Comercial
https://www.blend4web.com/en/

Sliders creados por Alon Zakai creador de EmScripten sobre C, Js y de como soluciona el compilador las diferencias.
https://kripken.github.io/mloc_emscripten_talk/cppcon.html#/

Pagina oficial del proyecto WASM
http://webassembly.github.io/

Código que ya se ha portado a WebAssembly (como versiones de Doom o Quake), ejemplos y demos.
https://github.com/emscripten-core/emscripten/wiki/Porting-Examples-and-Demos

Sliders con ideas y ejemplos de WebAssembly con C++
https://nxxm.github.io/cppcon2018/CPP_EVERYWHERE_WITH_WASM.html#/

Comparativa de EmScripten, Rust y Blazor
https://www.webassemblyman.com/webassembly_front_end_web_development.html