


JAVA DE REGRESO A LA WEB

**CON A WEBASSEMBLY**

MIGUEL USECHE @ JCONF DOMINICANA | JUNIO | 2019

# ¡HOLA! SOY MIGUEL



- Desarrollador web independiente y profesor universitario.
- Voluntario en comunidades de código abierto: 
- Mozillero de 🇨🇴 viviendo en 🇨🇴



# San Cristobal

Táchira  
Venezuela



Directions



SAVE



NEARBY



SEND TO YOUR  
PHONE



SHARE



Photos

## Quick facts

San Cristóbal is the capital city of the Venezuelan state of Táchira. It is located in a mountainous region of Western Venezuela. The city is situated 818 metres above sea level in the northern Andes overlooking the Torbes River, 56 kilometres from the Colombian border. [Wikipedia](#)



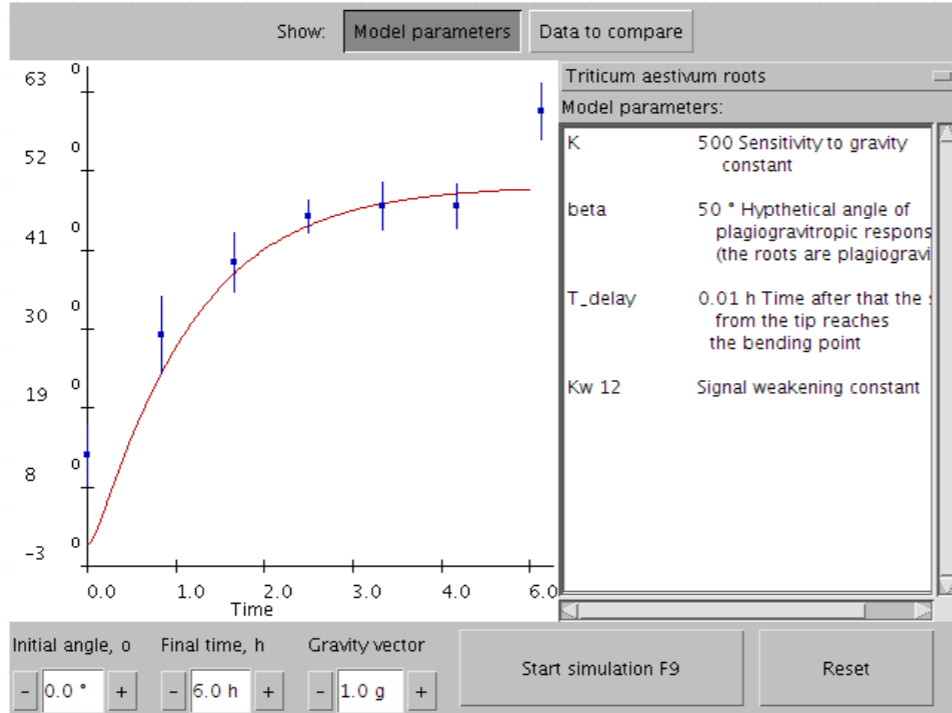
**¡EMPECEMOS!**







# JAVA APPLETS



**Aplicación matemática**



**NASA World Wind**

# ¿POR QUÉ DESAPARECIERON LOS JAVA APPLETS?

- Requería instalar Java en tu equipo.
- Nula integración con la web.
- Perdió fuerza debido a Flash, Silverlight, HTML5
- Problemas en dispositivos móviles
- ¿Problemas de seguridad?



**.....LOS APPLETS DE JAVA  
FUERON DESPLAZADOS...**

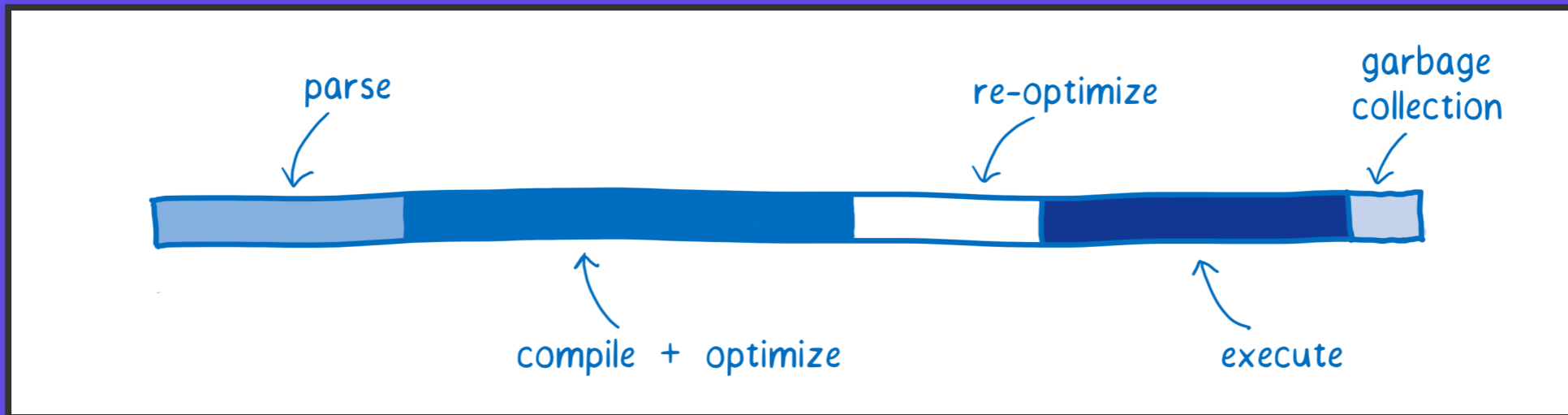
....ante JavaScript



A close-up photograph of a grey, textured metal surface, likely a piece of machinery. A circular hole is visible in the center of the frame. A white rectangular box is superimposed over the image, containing the word "RENDIMIENTO" in a serif font. The lighting is dramatic, with strong highlights and deep shadows, emphasizing the metallic texture and the circular feature.

RENDIMIENTO

# CÓMO LOS NAVEGADORES EJECUTAN JS



Fuente: Lin Clark © 2017.

**VEAMOS UN EJEMPLO...**

# UNA SUMA EN JS

```
function suma(a, b) {  
  return a + b;  
}
```

Sencillo, ¿no?

Debería "*ser una única*" instrucción del CPU

# CÓMO ECMA-262 DEFINE UNA ADICIÓN

1. Let *lref* be the result of evaluating *AdditiveExpression*.
2. Let *lval* be *GetValue(lref)*.
3. Let *rref* be the result of evaluating *MultiplicativeExpression*.
4. Let *rval* be *GetValue(rref)*.
5. Let *lprim* be *ToPrimitive(lval)*.
6. Let *rprim* be *ToPrimitive(rval)*.
7. If *Type(lprim)* is *String* or *Type(rprim)* is *String*, then
  - Return the *String* that is the result of concatenating *ToString(lprim)* followed by *ToString(rprim)*
  - Return the result of applying the addition operation to *ToNumber(lprim)* and *ToNumber(rprim)*.

Fuente: [ECMAScript® Language Specification](#). (Sec. 11.6.1)



# ¿POR QUÉ?

Hay que hacer sacrificios



Entienden lenguajes de programación



Entienden binario o código de bajo nivel

**¿CÓMO RESOLVER ESTO?**



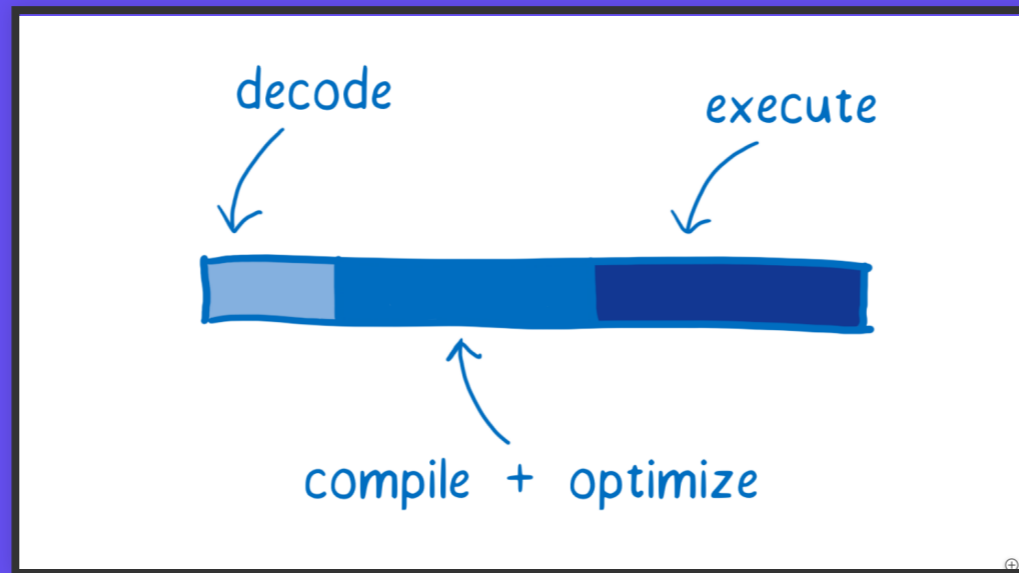
WEBASSEMBLY

# WEBASSEMBLY



- Formato binario
- **NO** reemplaza JS
- Interface desde/hacia tu lenguaje (Java)
- Integración con WebAPI
- float32,int64, threads, SIMD
- Fácil de compilar, verificar y extensible
- Bloques de memoria de 1 byte

# CÓMO LOS NAVEGADORES EJECUTAN WASM

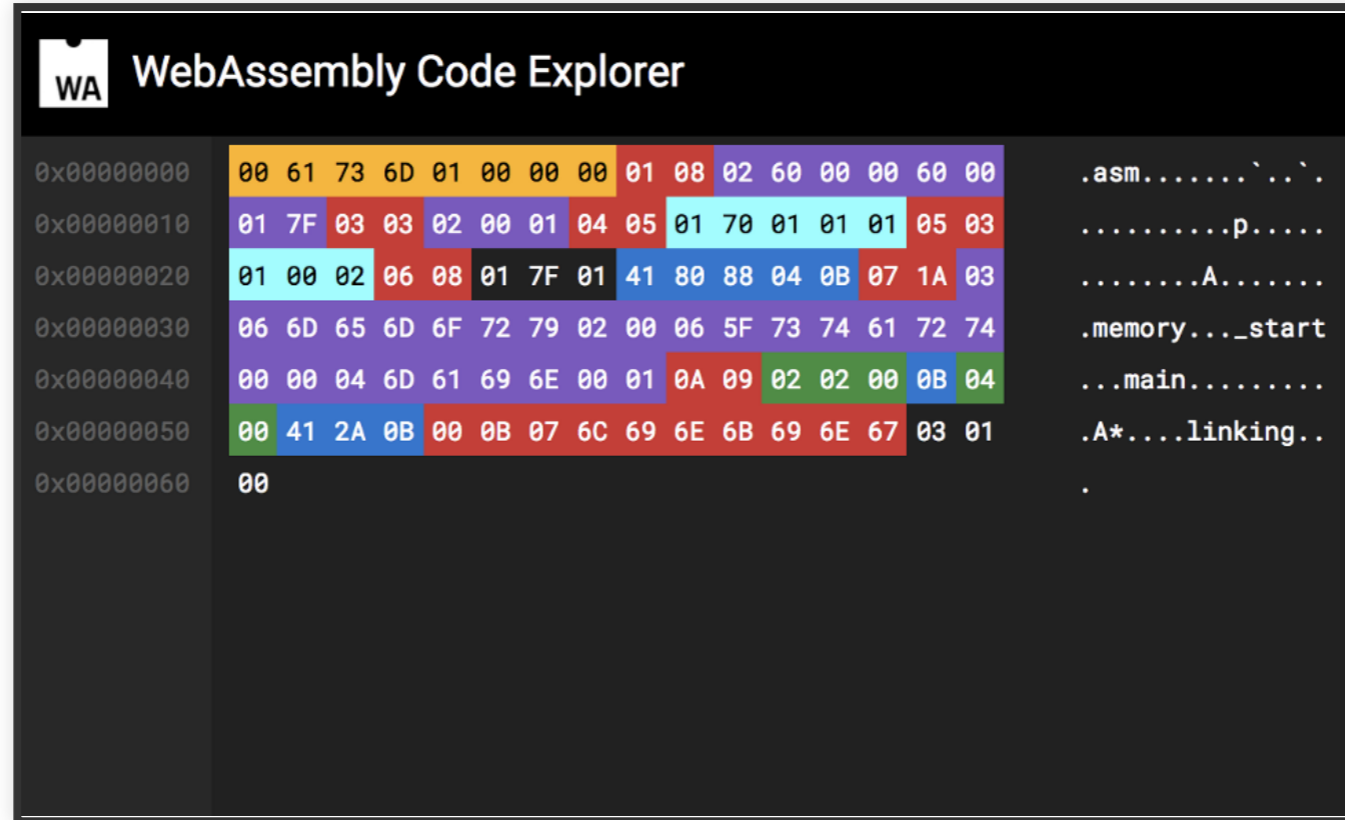


Fuente: Lin Clark © 2017.



# FORMATOS DE WASM

# FORMATO BINARIO



The screenshot displays the WebAssembly Code Explorer interface. It features a dark background with a 'WA' icon in the top left. The main content is organized into three columns: memory addresses on the left, binary code in the center, and assembly instructions on the right. The binary code is presented as a grid of hex values, with each value highlighted in a different color. The assembly instructions are aligned with the corresponding rows of binary code.

Address	Binary Code	Assembly
0x00000000	00 61 73 6D 01 00 00 00 01 08 02 60 00 00 60 00	.asm.....`...`.
0x00000010	01 7F 03 03 02 00 01 04 05 01 70 01 01 01 05 03	.....p.....
0x00000020	01 00 02 06 08 01 7F 01 41 80 88 04 0B 07 1A 03	.....A.....
0x00000030	06 6D 65 6D 6F 72 79 02 00 06 5F 73 74 61 72 74	.memory..._start
0x00000040	00 00 04 6D 61 69 6E 00 01 0A 09 02 02 00 0B 04	...main.....
0x00000050	00 41 2A 0B 00 0B 07 6C 69 6E 6B 69 6E 67 03 01	.A*....linking..
0x00000060	00	.

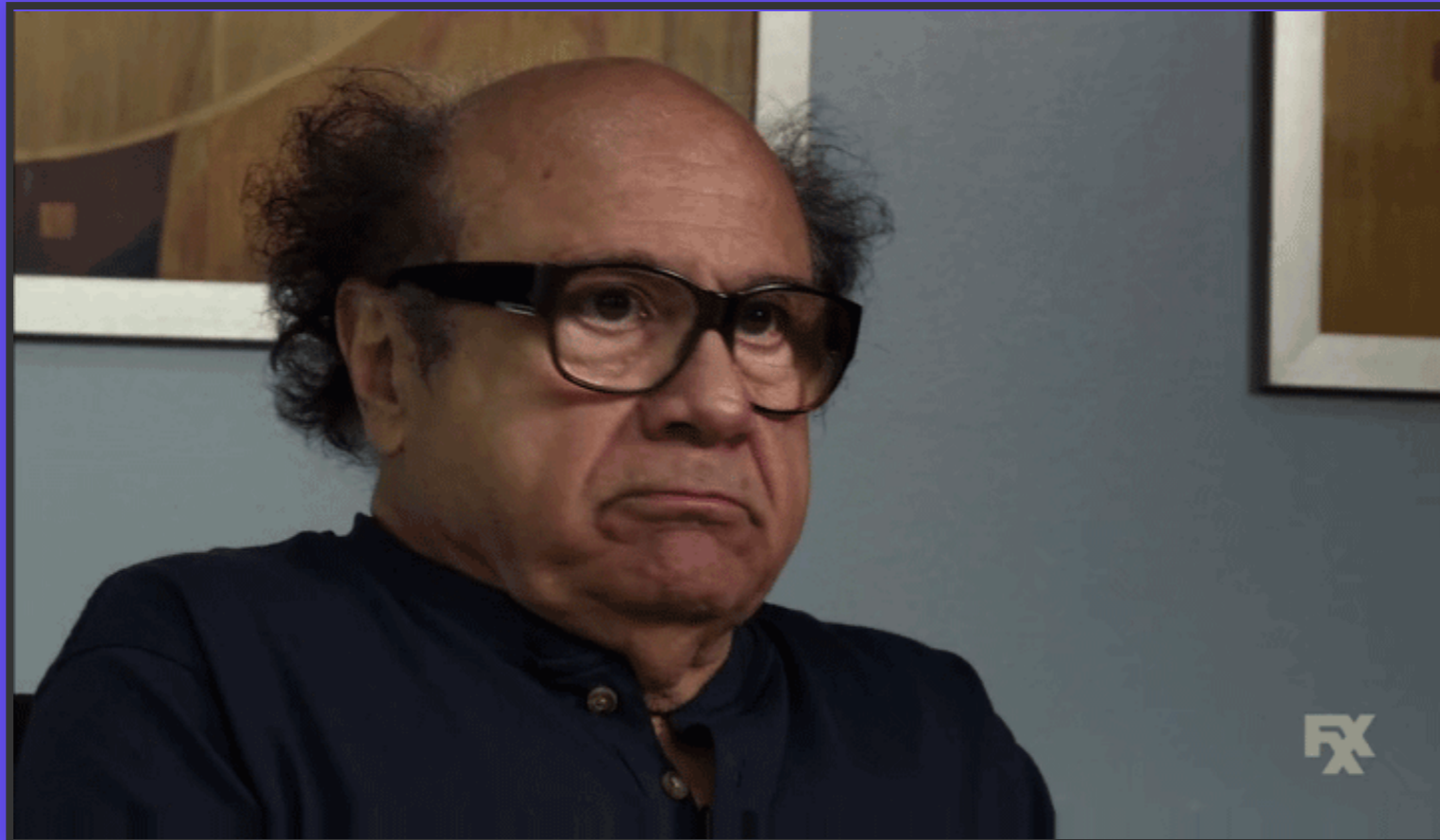
Puedes verlo con [WASM Code Explorer](#)

# WEBASSEMBLY TEXT FORMAT

```
(module
  (type $type0 (func (param i32)))
  (type $type1 (func))
  (import "sys" "print" (func $import0 (param i32)))
  (memory (;0;) 200 200)
  (export "memory" (memory 0))
  (export "main" (func $func1))
  (func $func1
    i32.const 0
    call $import0
  )
  (data (i32.const 0)
    "Hello, world\00"
  )
)
```

Hola Mundo en WAT

**TAL VEZ PIENSES QUE ES UNA  
MODA PASAJERA....**



**¡PERO NO!**

**¡TIENE GRAN SOPORTE!**





**¿CÓMO SE CREA?**

# WEBASSEMBLY EN JAVA

Existen 3 proyectos:

- JWebAssembly
- TeaVM
- Bytecoder

# WEBASSEMBLY EN JAVA

## COMPILACIÓN

- JWebAssembly y TeaVM

```
Generados con MAVEN
```

- Bytecoder:

```
java -jar bytecoder-cli-2019-06-13-executable.jar  
-classpath=. -mainclass=TU_PAQUETE.ArchivoPrincipa  
-builddirectory=. -backend=js -minify=false
```

**¿KOTLIN?**

# KOTLIN A WASM CON EMSCRIPTEN

Emscripten es conjunto de herramientas para llevar lenguajes de alto nivel a WASM

Kotlinsup → LLVM → Emscripten + Binaryen →  
WASM

\* Todo este proceso se hace automáticamente con un script de Graddle.

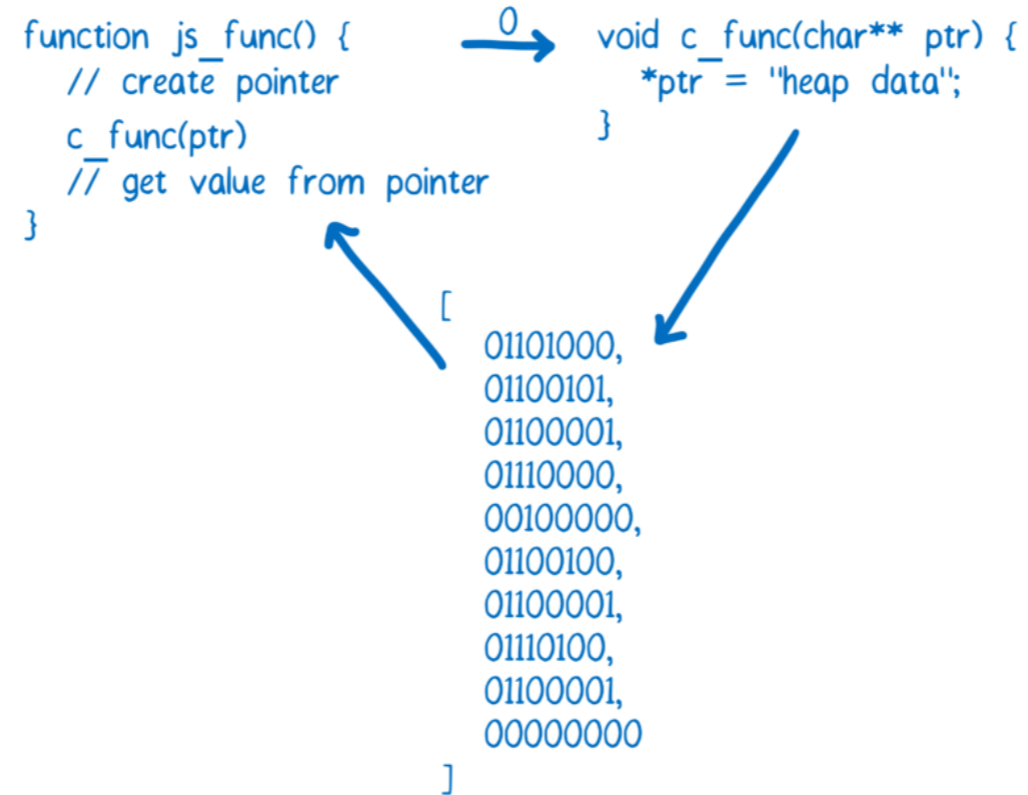
**¿QUÉ OBTENGO CON ESTO?**

# RESULTADOS DE LA COMPILACIÓN

Vas a obtener 3 archivos:

1. El archivo **.wasm**.
2. El código de JS para importar el módulo.
3. Una página HTML para ejecutar el módulo.

# INTERACCIÓN DEL CÓDIGO



Comunica código entre JS y WASM



# INTERACCIÓN JAVA -> JS

## JWEBASSEMBLY

### Java

```
import de.inetsoftware.jwebassembly.api.annotation.Export;

@Export
public static int sumaEnJava( int a, int b ) {
    return a + b;
}
```

### JavaScript

```
let funcionJs;

WebAssembly.instantiateStreaming(fetch('simple.wasm'))
.then(obj => funcionJs = obj.instance.exports.sumaEnJava());
```

# INTERACCIÓN JAVA → JS

## BYTECODER

Java

```
public class MiClase {  
    @Export("funcionDeJava")  
    public static void funcionDeJava() {  
    }  
}
```

JavaScript

```
bytecoder.exports.funcionDeJava();
```

# INTERACCIÓN JS → JAVA

## JWEBASSEMBLY

### Re-implementación:

```
import de.inetsoftware.jwebassembly.api.annotation.Import;

@Import( module = "global.Math", name = "max" )
static int max( int a, int b ) {
    return Math.max( a, b );
}
```

### Usando native:

```
import de.inetsoftware.jwebassembly.api.annotation.Import;

@Import( module = "global.Math", name = "max" )
public static native double max( int a, int b );
```

# INTERACCIÓN JS → JAVA

## BYTECODER

```
import de.mirkosertic.bytecoder.api.Import;

public class CanvasRenderingContext2D {

    @Import(module = "math", name = "max")
    public static native long max(long aValue1, long aValue2);

}
```

**¿QUÉ PUEDO HACER CON  
WASM?**

**moz://a**

# EDITOR DE VÍDEO CON WASM

**WebAssembly Video Editor**  
github.com/shamadee/web-dsp

Switch to Video (✓ WebAssembly is supported in your browser)

- Normal
- Grayscale
- Invert
- Bacteria
- Sunset
- Emboss
- Super Edge
- Super Edge Inv
- Gaussian Blur
- Moss
- Robbery
- Brighten
- Swamp
- Ghost
- Good Morning
- Acid
- Urple
- Romance
- Hippo
- Longhorn
- Security
- Underground
- Rooster



Switch back to video for player controls

**Performance Comparison:**  
WASM is currently 50% faster than JS

Average computation time WASM: 2.4 ms, JS: 2.3 ms

WASM computation time: 2 ms JS computation time: 3 ms

**Frame Rate. WASM = Green; JS = Blue;**



Choppy Video? Disable Javascript below to see WebAssembly only

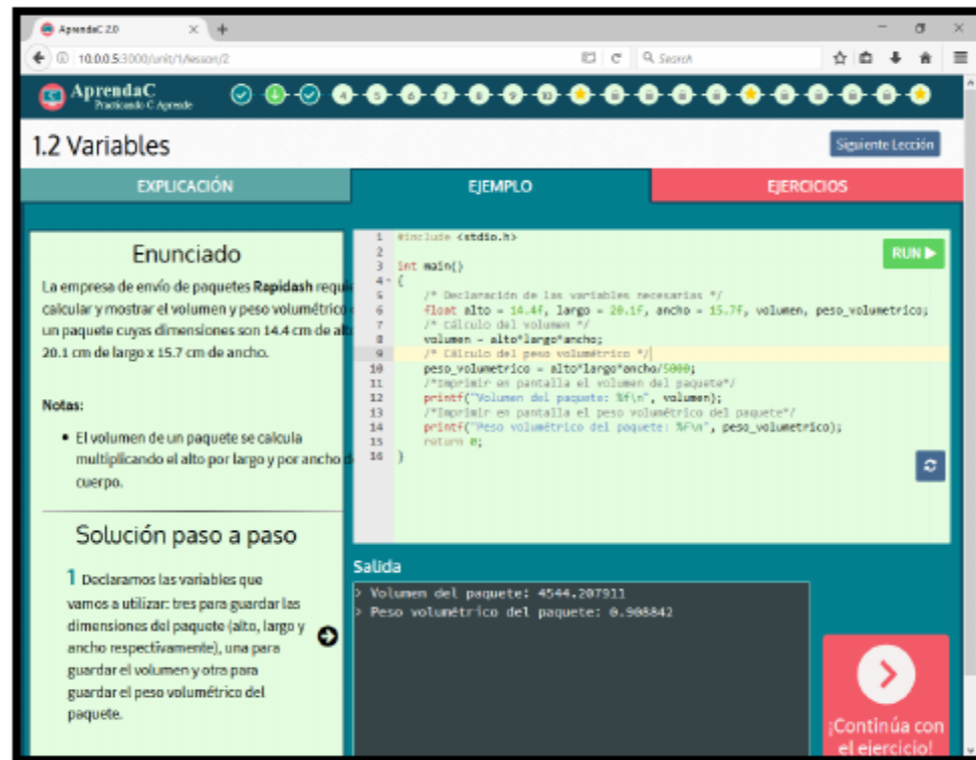
Disable Javascript Hide JS Canvas

!

Aplica los efectos en tiempo real



# APRENDE C (LEARN C)



The screenshot shows the AprendaC website interface. The page title is "1.2 Variables". There are three tabs: "EXPLICACIÓN", "EJEMPLO", and "EJERCICIOS". The "EJEMPLO" tab is active, displaying a C code snippet for calculating the volume and volumetric weight of a package. The code is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     /* Declaración de las variables necesarias */
6     float alto = 14.4f, largo = 20.1f, ancho = 15.7f, volumen, peso_volumetrico;
7     /* Cálculo del volumen */
8     volumen = alto*largo*ancho;
9     /* Cálculo del peso volumétrico */
10    peso_volumetrico = alto*largo*ancho/5000;
11    /*Imprimir en pantalla el volumen del paquete*/
12    printf("Volumen del paquete: %f\n", volumen);
13    /*Imprimir en pantalla el peso volumétrico del paquete*/
14    printf("Peso volumétrico del paquete: %f\n", peso_volumetrico);
15    return 0;
16 }
```

The output of the code is shown in the "Salida" section:

```
> Volumen del paquete: 4544.207911
> Peso volumétrico del paquete: 0.968842
```

The "Enunciado" section describes a problem: "La empresa de envío de paquetes Rapidash requiere calcular y mostrar el volumen y peso volumétrico de un paquete cuyas dimensiones son 14.4 cm de alto, 20.1 cm de largo x 15.7 cm de ancho." The "Notas" section states: "El volumen de un paquete se calcula multiplicando el alto por largo y por ancho del cuerpo." The "Solución paso a paso" section explains: "1 Declaramos las variables que vamos a utilizar: tres para guardar las dimensiones del paquete (alto, largo y ancho respectivamente), una para guardar el volumen y otra para guardar el peso volumétrico del paquete."

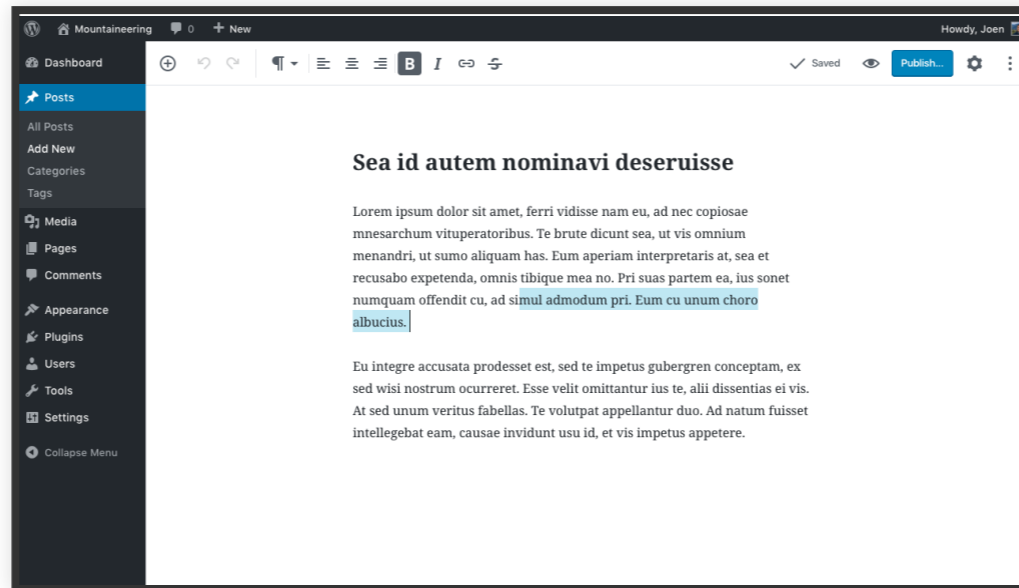
Un sitio para aprender C que se ejecuta en el navegador.

# AUTOCAD EN LA WEB



Fuente: [Twitter](#).

# INTÉRPRETE DE WORDPRESS GUTENBERG



El nuevo editor de WordPress utiliza un intérprete escrito en Rust.

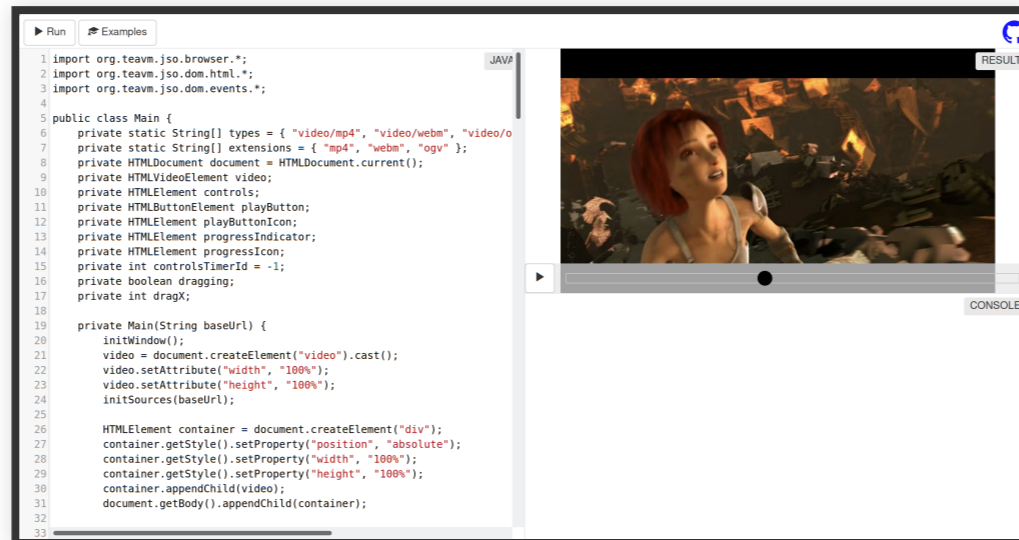
## WASM binary

A [yet-to-be-official benchmark](#) is used to compare the performance of the actual Javascript parser against the Rust parser compiled as a WASM binary so that it can run in the browser. Here are the results:

file	Javascript parser (ms)	Rust parser as a WASM binary (ms)	speedup
<a href="#">demo-post.html</a>	13.167	0.137	× 96
<a href="#">shortcode-shortcomings.html</a>	26.784	0.225	× 119
<a href="#">redesigning-chrome-desktop.html</a>	75.500	0.905	× 83
<a href="#">web-at-maximum-fps.html</a>	88.118	0.698	× 126
<a href="#">early-adopting-the-future.html</a>	201.011	0.927	× 217
<a href="#">pygmalian-raw-html.html</a>	311.416	1.016	× 307
<a href="#">moby-dick-parsed.html</a>	2,466.533	14.673	× 168

The WASM binary of the Rust parser is in average 159 times faster than the actual Javascript implementation. The median of the speedup is 126.

# TEA-VM DEMO: JAVA EN EL NAVEGADOR



The screenshot displays the TEA-VM demo interface. On the left, a code editor shows the following Java code:

```
1 import org.teavm.js.browser.*;
2 import org.teavm.js.dom.html.*;
3 import org.teavm.js.dom.events.*;
4
5 public class Main {
6     private static String[] types = { "video/mp4", "video/webm", "video/o
7     private static String[] extensions = { "mp4", "webm", "ogv" };
8     private HTMLDocument document = HTMLDocument.current();
9     private HTMLVideoElement video;
10    private HTMLFormElement controls;
11    private HTMLButtonElement playButton;
12    private HTMLImageElement playButtonIcon;
13    private HTMLFormElement progressIndicator;
14    private HTMLImageElement progressIcon;
15    private int controlsTimerId = -1;
16    private boolean dragging;
17    private int dragX;
18
19    private Main(String baseUrl) {
20        initWindow();
21        video = document.createElement("video").cast();
22        video.setAttribute("width", "100%");
23        video.setAttribute("height", "100%");
24        initSources(baseUrl);
25
26        HTMLFormElement container = document.createElement("div");
27        container.getStyle().setProperty("position", "absolute");
28        container.getStyle().setProperty("width", "100%");
29        container.getStyle().setProperty("height", "100%");
30        container.appendChild(video);
31        document.getBody().appendChild(container);
32
33
```

On the right, a video player is shown with a play button and a progress bar. The video content is a scene from a movie featuring a woman with red hair. The interface also includes a 'Run' button, 'Examples' link, and a 'CONSOLE' area.

Pruébalo en: <http://teavm.org/sandbox/index.html>

¡PODEMOS HACER  
FULL-STACK CON JAVA!

¡Adiós JS! (En parte)

**¿FALTAN MAS  
EJEMPLOS DE JAVA?**

**¡TE INVITAMOS A CREARLOS!**



# FUTURAS CARACTERÍSTICAS DE WASM

- Multi-threading y Atómicas
- Manejo de excepciones
- Recolector de basura
- Variables globales mutables
- SIMD.
- JS BigInt y WebAssembly i64.

Mas en: <https://webassembly.org/docs/future-features/>

## LIMITACIONES EN JAVA

- No existe un recolector de basura (GC).
- Soporte limitado de hilos.
- No hay un soporte oficial, solo proyectos de terceros.

A man in a dark suit and tie is seated at a wooden desk. In front of him is a black rotary telephone. To the right of the phone is a bottle of cognac in a wooden presentation box. The background is a dark wood-paneled wall.

¡NO ESPERES!

¡PRUÉBALO CON

**WEBASSEMBLY STUDIO!**

Rea

# ENALCES RECOMENDADOS

- [WASM official site](#)
- [WebAssembly Studio](#)
- [Web Docs information](#)
- [WASM Binary Structure](#)
- [Mozilla Hacks](#)
- [JWebAssembly](#)
- [TeaVM](#)
- [Bytecoder](#)



**GRACIAS POR SU ATENCIÓN**

[migueluseche.com](http://migueluseche.com)

[migueluseche@mozilla-hispano.org](mailto:migueluseche@mozilla-hispano.org)

[@skatox](#) | [@mozillahispano](#)