
GAME PROGRAMMING
imgflip.com


I CAN DO IT!!
imgflip.com

Comment j'ai porté DOOM sur le navigateur grâce au WebAssembly

TNT

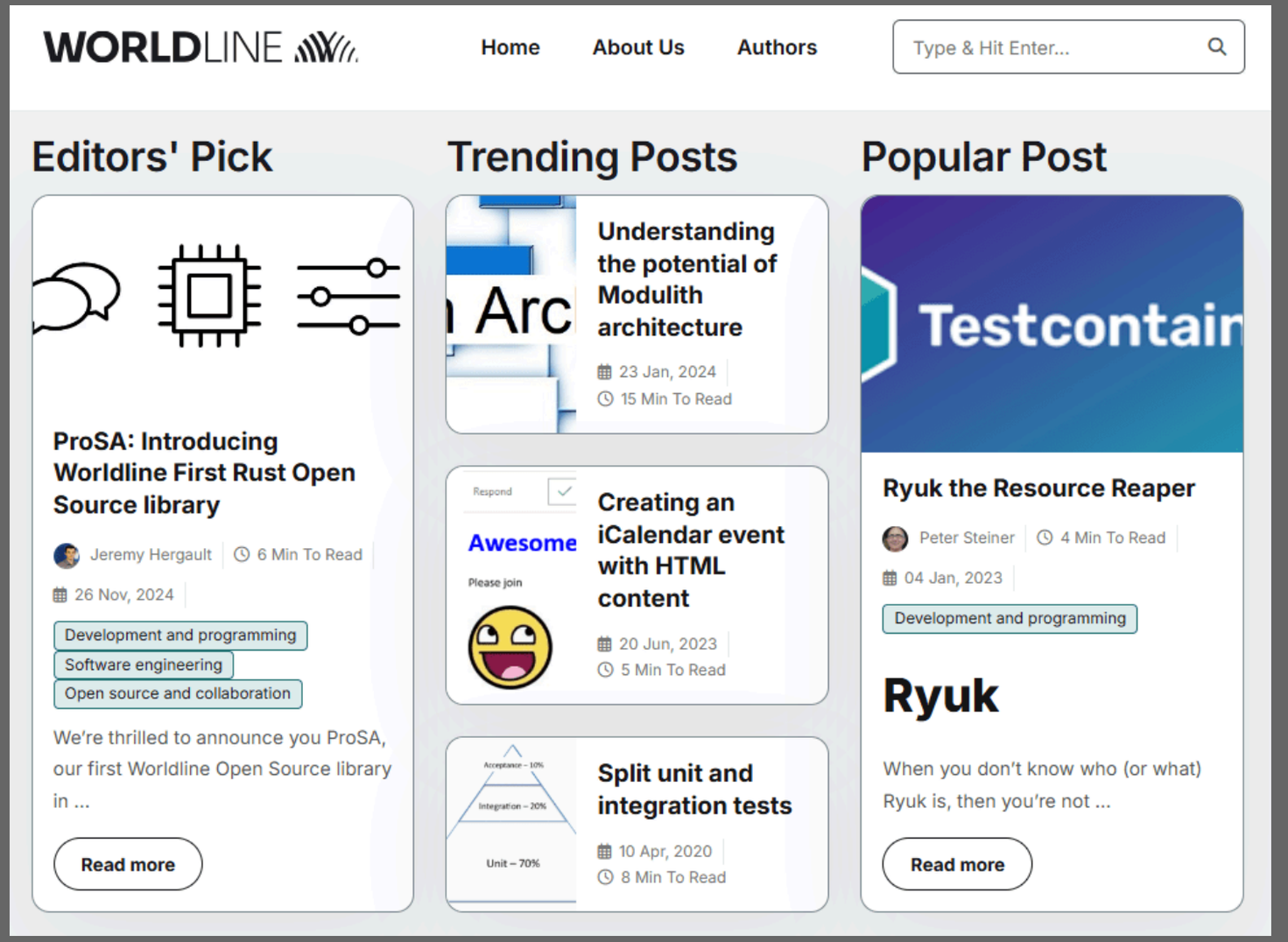
Yassine
Benabbas

DevRel / Enseignant / membre du LAUG
Amateur de jeux rétro
😍 Kotlin, WASM, AI, ...

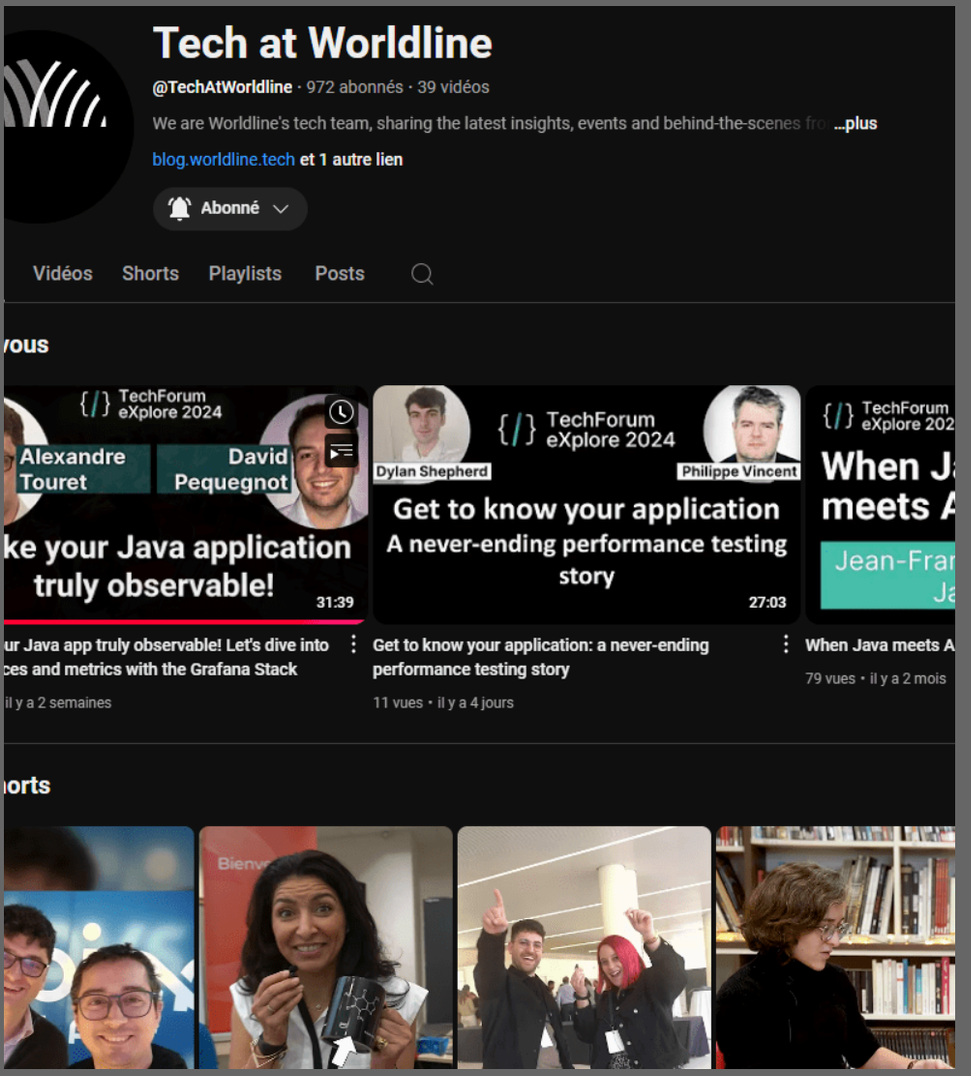
@yostane

WORLDLINE

#TechAtWorldline



blog.worldline.tech



@TechAtWorldline

@techatworldline.bsky.social

@worldlinetech


PROLOG
WEBASSEMBLY

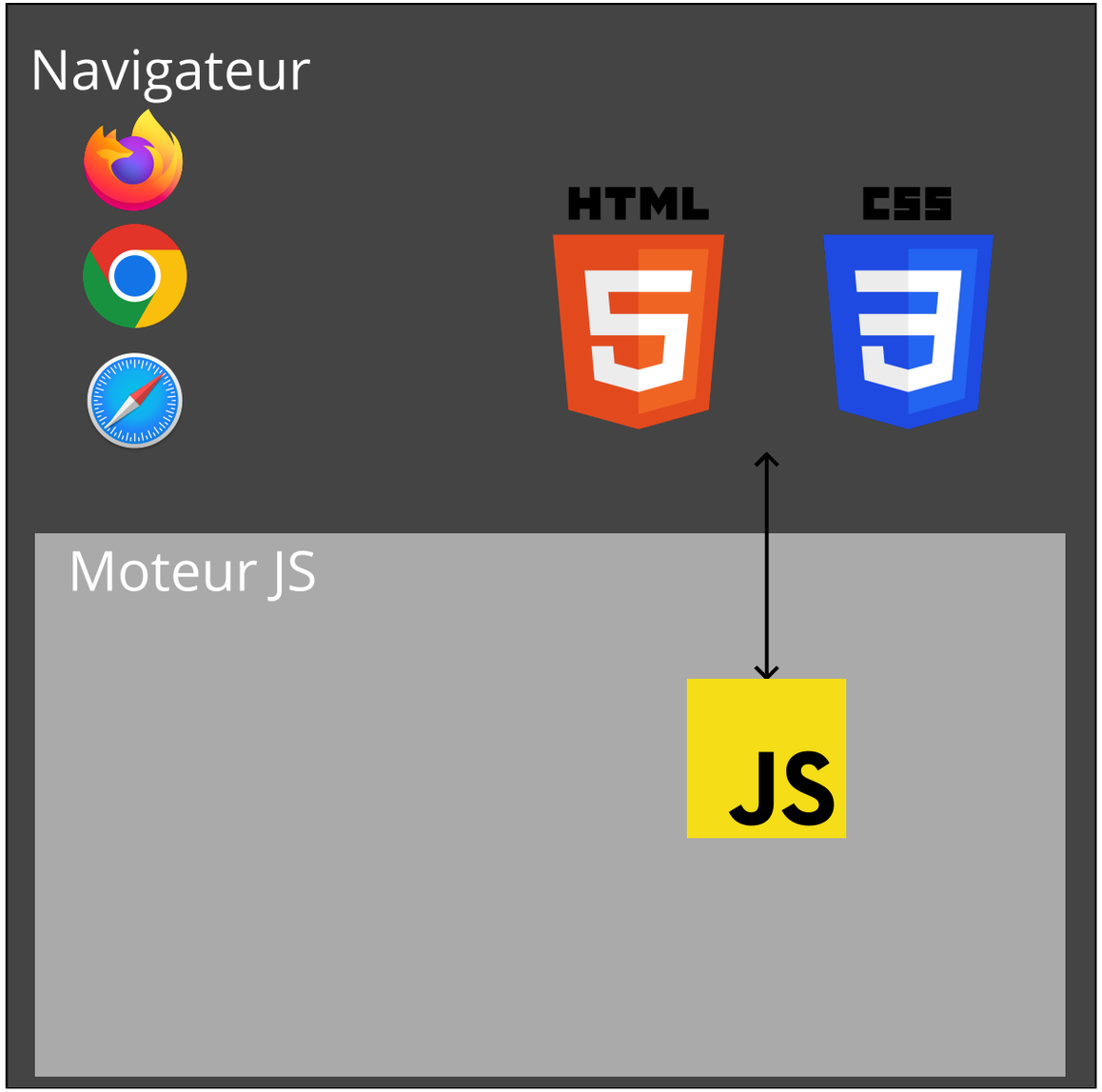

```
1  00 61 73 6d 01 00 00 00   01 05 01 60 00 01 7f 03   |.asm.......`....|
2  02 01 00 07 16 01 12 67   65 74 55 6e 69 76 65 72   |.......getUniver|
3  73 61 6c 4e 75 6d 62 65   72 00 00 0a 06 01 04 00   |salNumber.......|
4  41 2a 0b 00 0a 04 6e 61   6d 65 02 03 01 00 00      |A*....name......|
```

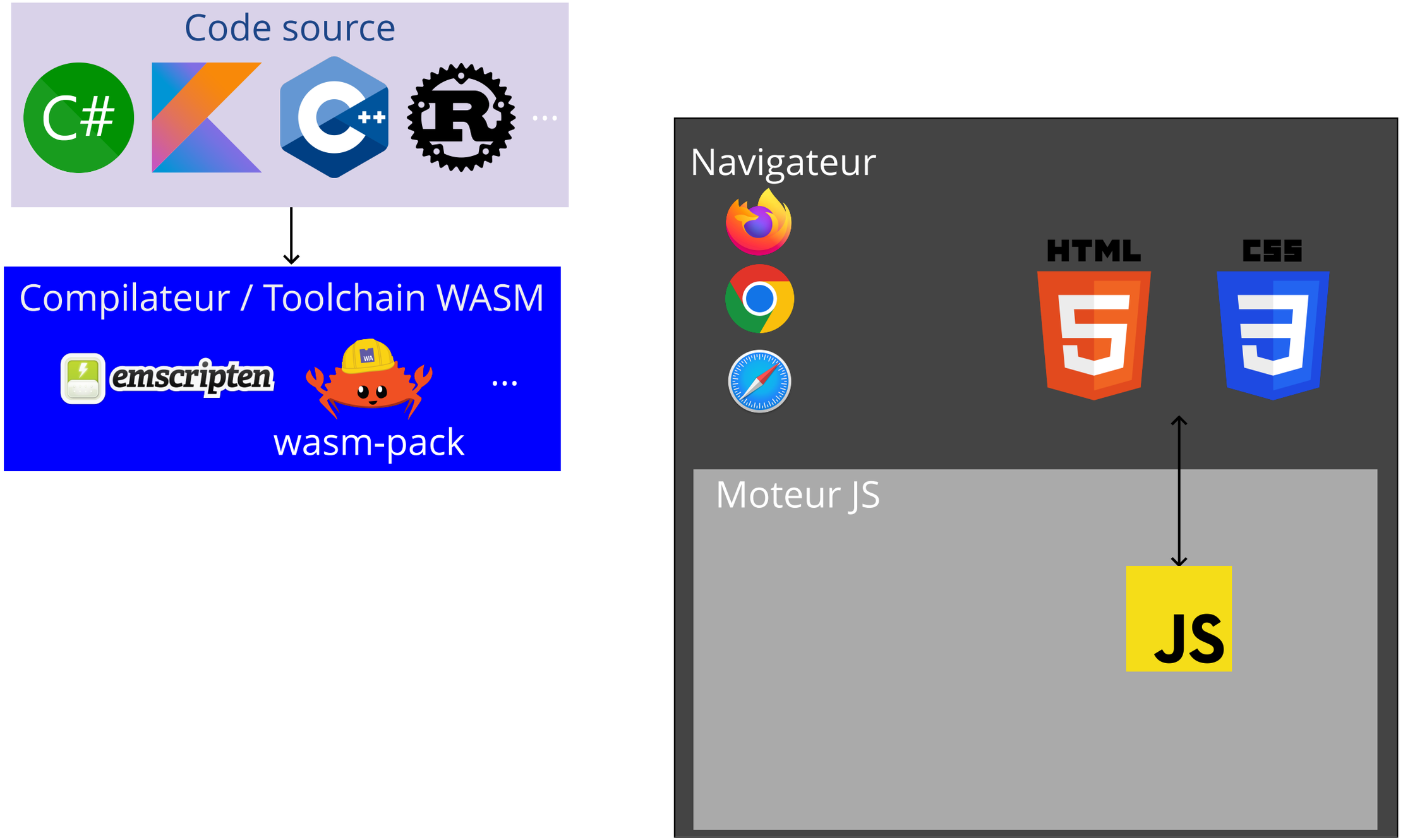
```
1  00 61 73 6d 01 00 00 00   01 05 01 60 00 01 7f 03   |.asm.......`....|
2  02 01 00 07 16 01 12 67   65 74 55 6e 69 76 65 72   |.......getUniver|
3  73 61 6c 4e 75 6d 62 65   72 00 00 0a 06 01 04 00   |salNumber.......|
4  41 2a 0b 00 0a 04 6e 61   6d 65 02 03 01 00 00      |A*....name.....|
```

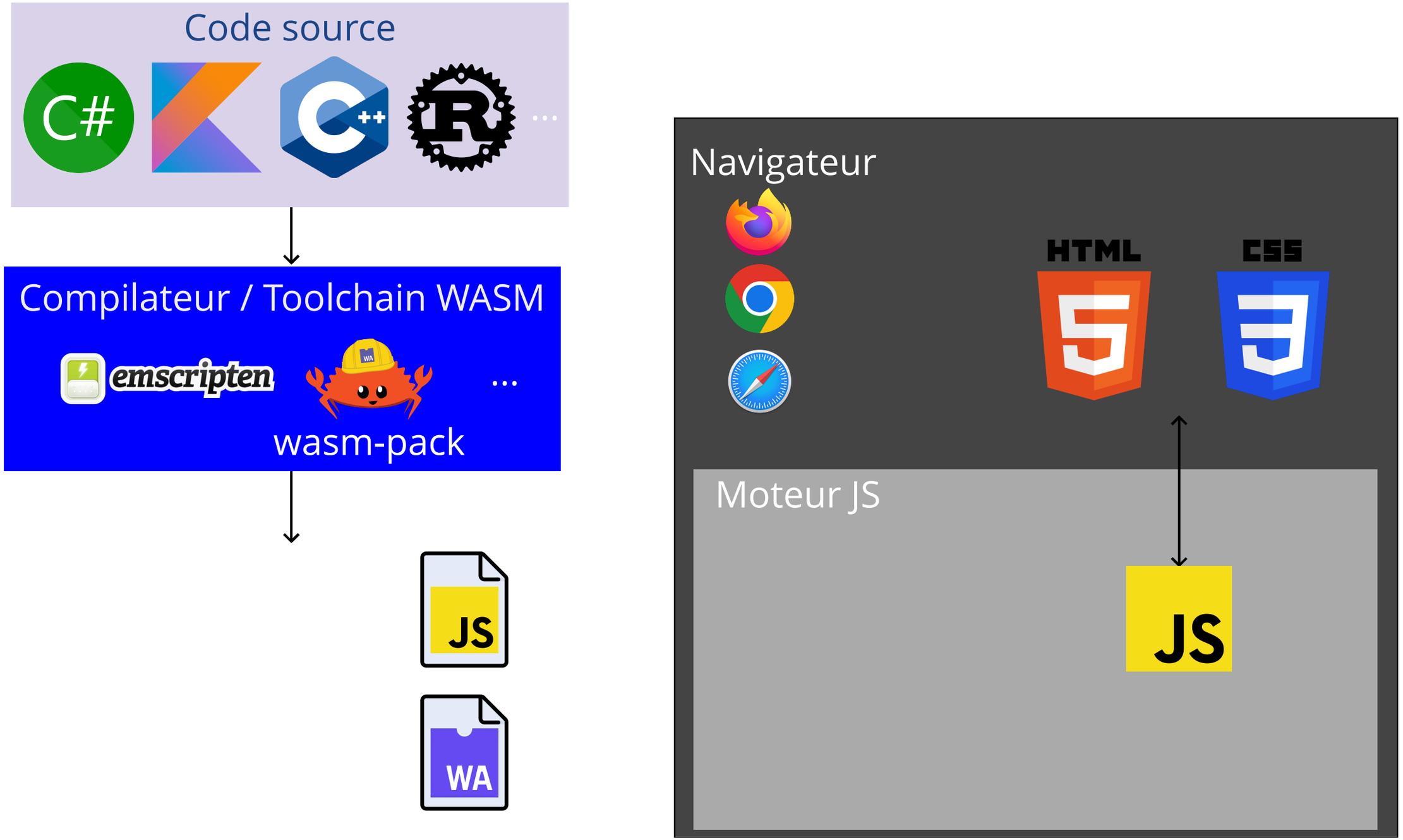
```
00 61 73 6d 01 00 00 00  01 05 01 60 00 01 7f 03  |.asm.......`....|
02 01 00 07 16 01 12 67  65 74 55 6e 69 76 65 72  |.......getUniver|
73 61 6c 4e 75 6d 62 65  72 00 00 0a 06 01 04 00  |salNumber.......|
41 2a 0b 00 0a 04 6e 61  6d 65 02 03 01 00 00     |A*....name.....|
```

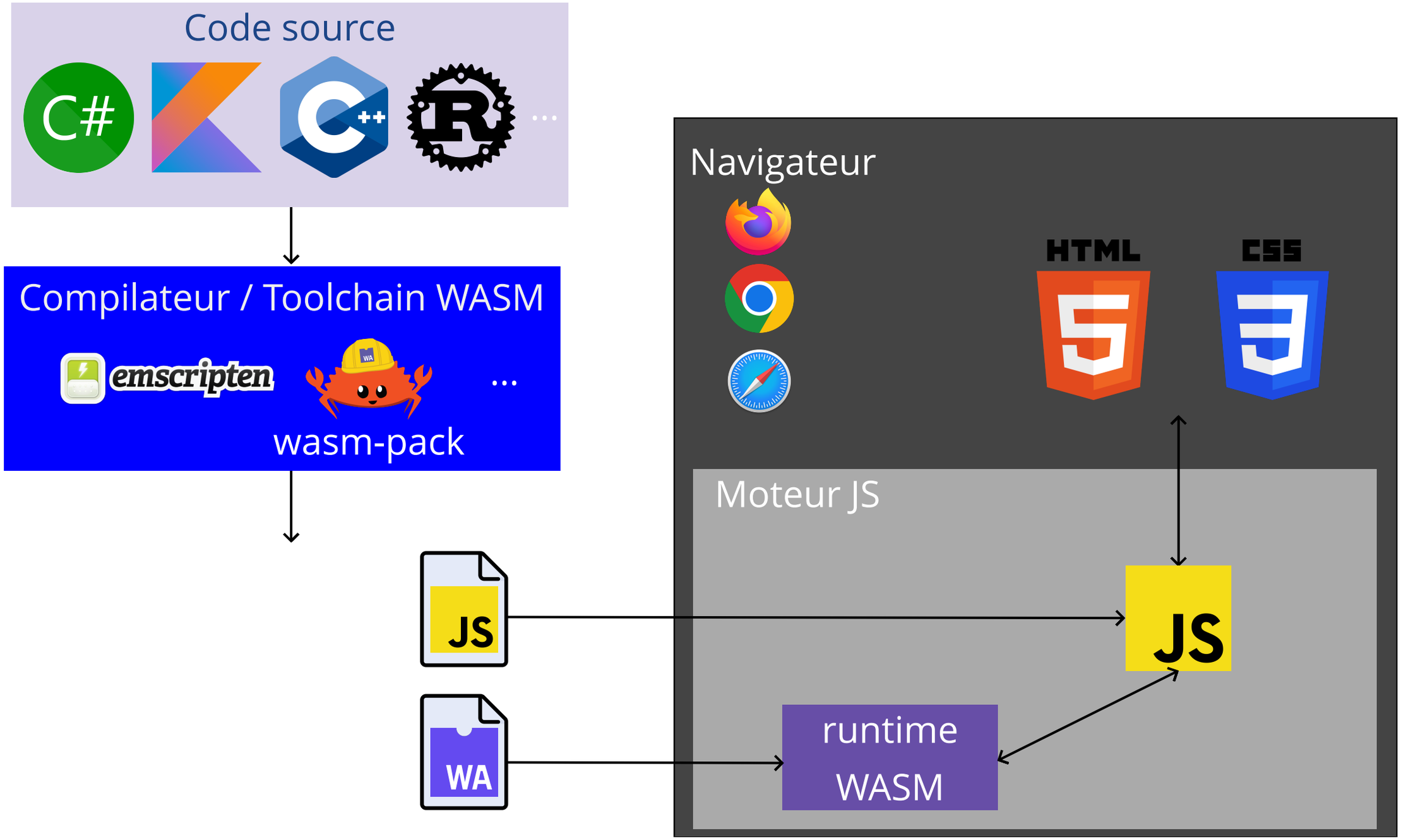

```
(module
  (func (result i32)
    (i32.const 42)
  )
  (export "getUniversalNumber" (func 0))
)
```

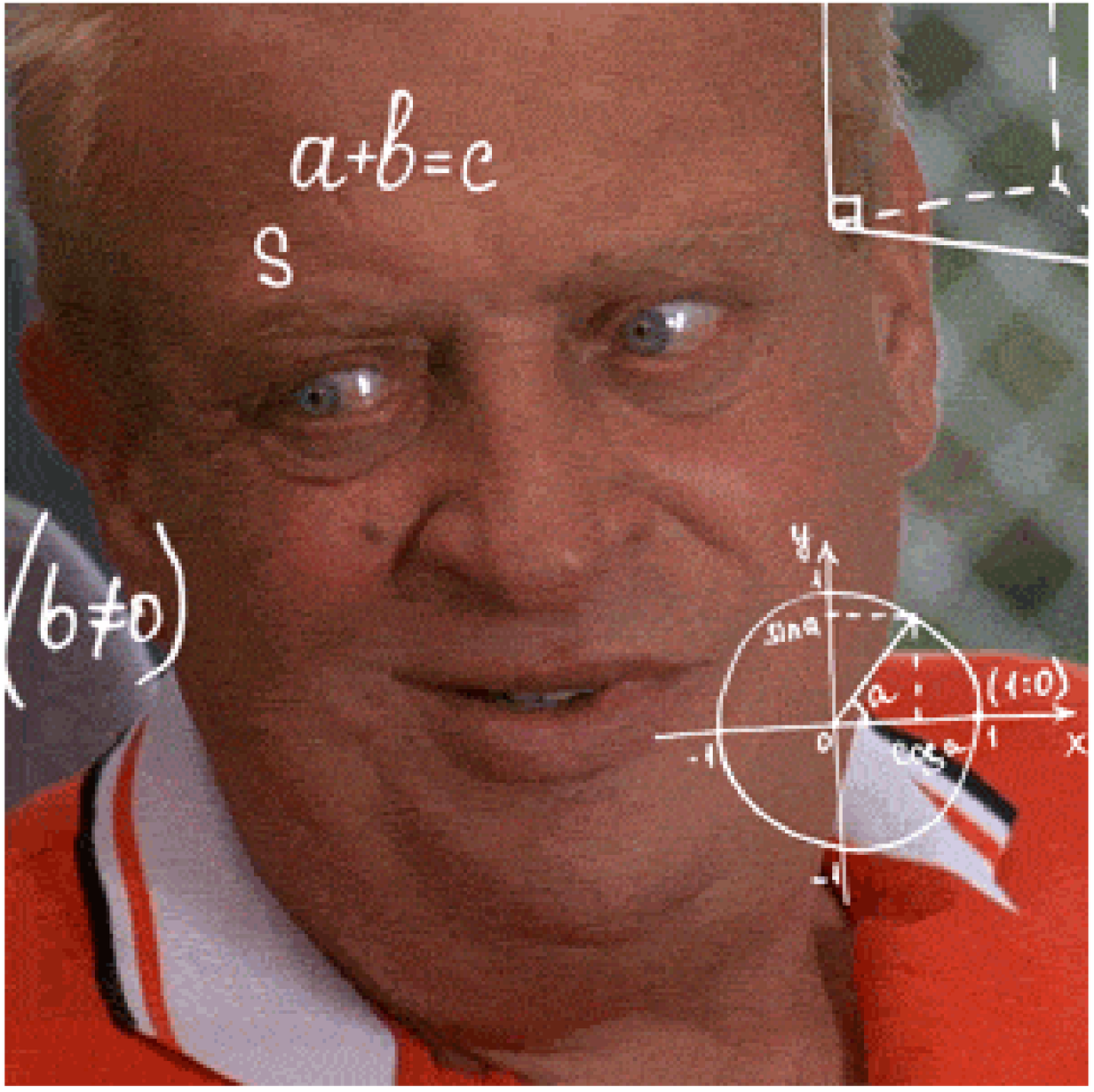
Code source




Code source
C#
C++
...
Compilateur / Toolchain WASM
emscripten
...
wasm-pack
Navigateur
HTML
CSS
Moteur JS
JS


Code source
C#
...
Compilateur / Toolchain WASM
emscripten
...
wasm-pack
JS
WA
Navigateur
HTML
CSS
Moteur JS
JS


Code source
C#
...
Compilateur / Toolchain WASM
emscripten
...
wasm-pack
Navigateur
HTML
CSS
Moteur JS
JS
JS
WA
runtime
WASM



Je dois réaliser un projet en WASM !

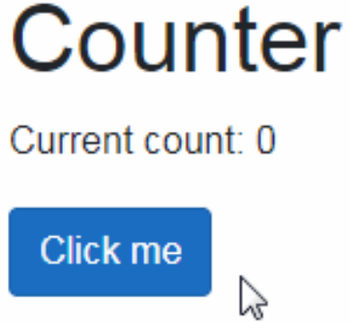

CHAPTER 1
.NET & WASM

.NET

.NET

Blazor

```razor
@page "/counter"

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;
    private void IncrementCount() { currentCount++; }
}
```



```csharp
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

C#

```javascript
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

JS

```csharp
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

C#

```javascript
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

JS

```csharp
public partial class MyClass
{
    [JSExport]
    internal static string Greeting()
    {
        var text = $"Hello, {GetHRef()}";
        Console.WriteLine(text);
        return text;
    }

    [JSImport("window.location.href", "main.js")]
    internal static partial string GetHRef();
}
```

C#

```javascript
setModuleImports('main.js', {
    window: {
        location: {
            href: () => globalThis.window.location.href
        }
    }
});

const text = exports.MyClass.Greeting();
console.log(text);
```

JS

Je dois porter un jeu .NET vers WASM !


CHAMP 2
MANAGEDOOM

YouTubeur MVG


DOOM


100%
AMMO
HEALTH
ARMS
0%
ARMOR
BULL 23 / 200
SHEL 0 / 50
ROKT 0 / 50
CELL 0 / 300


DOOM


100%
AMMO
HEALTH
ARMS
0%
ARMOR
BULL 23 / 200
SHEL 0 / 50
ROKT 0 / 50
CELL 0 / 300


DOOM

wad

sf2


100%
AMMO
HEALTH
ARMS
0%
ARMOR
BULL 23 / 200
SHEL 0 / 50
ROKT 0 / 50
CELL 0 / 300

wad

sf2

🌟 Doom est facilement portable par conception 🌟


DOOM 64
MATURE
SOLD BY
MIDWAY
DOOM
JAGUAR
The Master Levels
VER 1.7
04-30415
Disk 4
DOOM II
Created by id Software
© 1994 id Software
Distributed by GT Interactive Software.
All Rights Reserved.
FINAL
DOOM
PlayStation
Williams
DISC
DOOM
Williams
32X
DOOM


DOOM
DOOM
DOOM 64
id


HCR
THE ULTIMATE
id


Kodak
DISPLAY
MENU
RECORD
CAPTURE
REVIEW
CONNECT


19:19
DOOM
id
Start


DOOM
PIANO
GUNS
STRAFE
TURN
BACK
FRONT
BACK
TURN
STRAFE
GUNS


JE PORTE
DOOM DANS
LE NAVIGATEUR
AVEC SDL2 ET
EMSCRIPTEN
54:39


33
101%
200%

LinuxDoom

SFML silk.net

LinuxDoom

SFML

silk.net

id-Software/DOOM

LinuxDoom

SFML

silk.net

id-Software/DOOM

↓

sinshu/managed-doom

LinuxDoom

SFML silk.net

id-Software/DOOM

↓

sinshu/managed-doom

↓

yostane/MangedDoom-Blazor

CHAMTAR 2

Réalisation du portage

sininshu/managed-doom

WASM

sininshu/managed-doom

WASM

sininshu/managed-doom

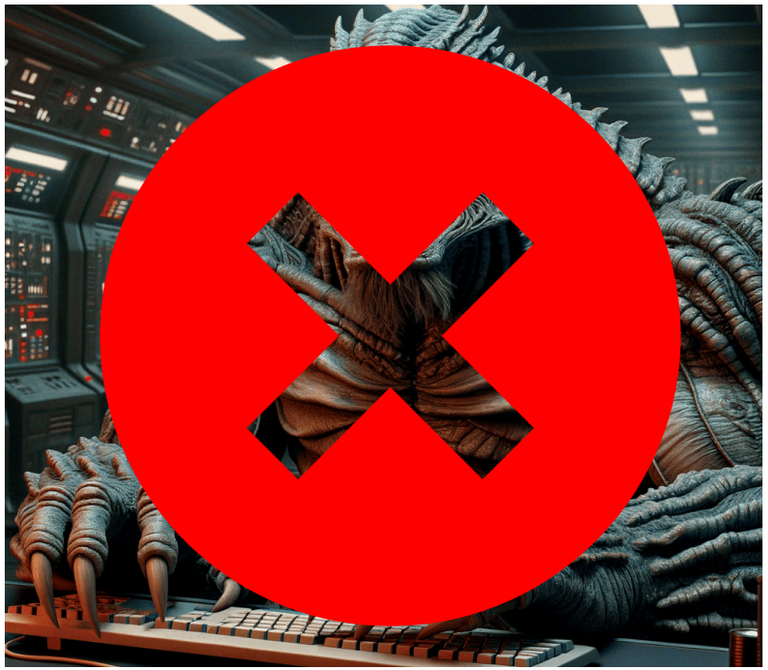
WASM

sininshu/managed-doom

WASM

sininshu/managed-doom

WASM



sininshu/managed-doom

WASM

SFML

```
1 using SFML.Audio;
2
3 namespace ManagedDoom.Audio
4 {
5     public sealed class SfmlSound : ISound, IDisposable
6     {
7         private SoundBuffer[] buffers;
8     }
9 }
```

SFML

```
1 using SFML.Audio;
2
3 namespace ManagedDoom.Audio
4 {
5     public sealed class SfmlSound : ISound, IDisposable
6     {
7         private SoundBuffer[] buffers;
8     }
9 }
```

SFML.Audio

SoundBuffer

```
 1 namespace SFML.Audio
 2 {
 3     public class SoundBuffer
 4     {
 5         public SoundBuffer(short[] samples, int v, uint sampleRate)
 6         {
 7             // TODO: implement
 8         }
 9
10         internal void Dispose()
11         {
12             // TODO: implement
13         }
14     }
15 }
```

```
 1 namespace SFML.Audio
 2 {
 3     public class SoundBuffer
 4     {
 5         public short[] samples;
 6         private int v;
 7         public uint sampleRate;
 8 
 9         public SoundBuffer(short[] samples, int v, uint sampleRate)
10         {
11             this.samples = samples;
12             this.v = v;
13             this.sampleRate = sampleRate;
14         }
15 
16         public Time Duration { get; internal set; }
17     }
18 }
```

```
 1 namespace SFML.Audio
 2 {
 3     public class SoundBuffer
 4     {
 5         public short[] samples;
 6         private int v;
 7         public uint sampleRate;
 8 
 9         public SoundBuffer(short[] samples, int v, uint sampleRate)
10         {
11             this.samples = samples;
12             this.v = v;
13             this.sampleRate = sampleRate;
14         }
15 
16         public Time Duration { get; internal set; }
17     }
18 }
```

```
 1 namespace SFML.Audio
 2 {
 3     public class SoundBuffer
 4     {
 5         public short[] samples;
 6         private int v;
 7         public uint sampleRate;
 8 
 9         public SoundBuffer(short[] samples, int v, uint sampleRate)
10         {
11             this.samples = samples;
12             this.v = v;
13             this.sampleRate = sampleRate;
14         }
15 
16         public Time Duration { get; internal set; }
17     }
18 }
```

```
while (waitForNextFrame()){
  const input = getPlayerInput();
  const { frame, audio }
          = UpdateGameState(input, WAD);
  render(frame);
  play(audio);
}
```

```
while (waitForNextFrame()){
  const input = getPlayerInput();
  const { frame, audio }
          = UpdateGameState(input, WAD);
  render(frame);
  play(audio);
}
```

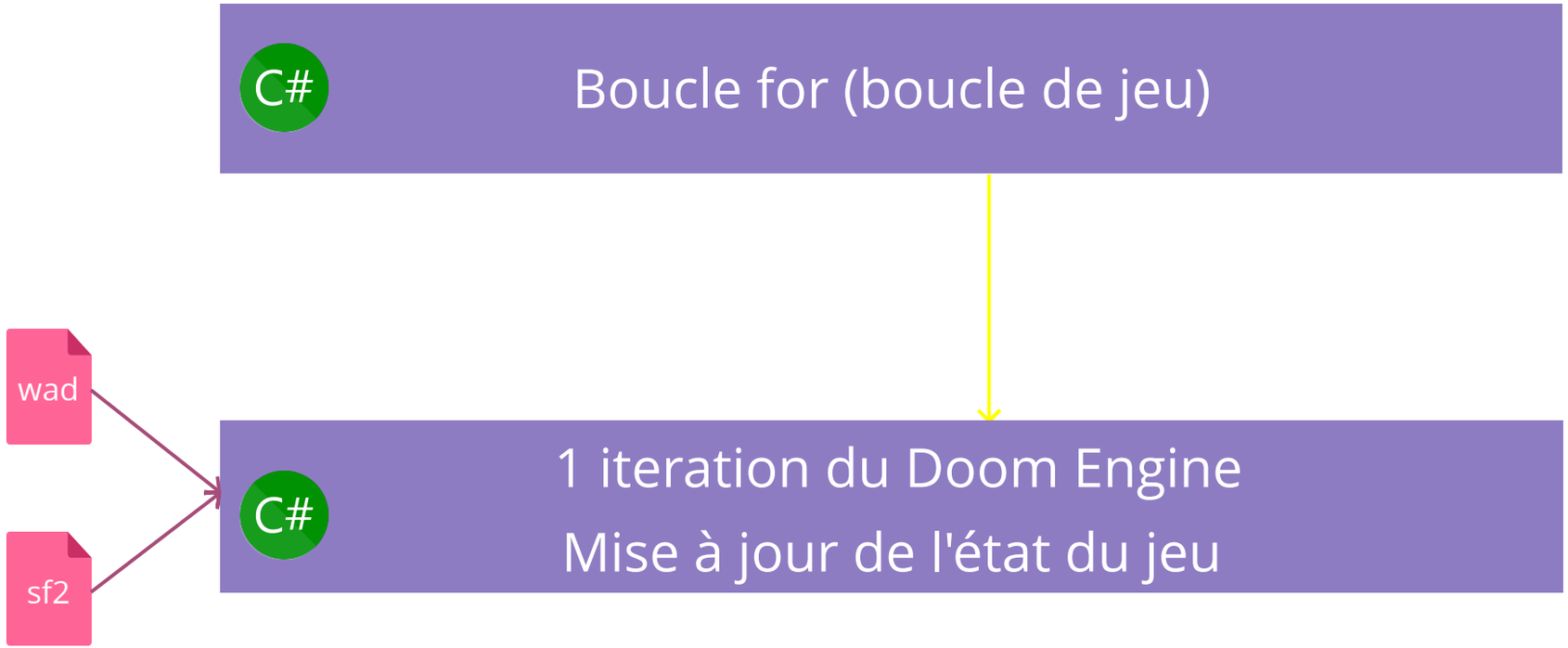
```
function gameLoop(){
  if (canAdvanceFrame()){
    const input = getPlayerInput();
    const { frame, audio }
              = UpdateGameState(input, WAD);
    render(frame);
    play(audio);
  }
  requestAnimationFrame(gameLoop);
}
requestAnimationFrame(gameLoop);
```

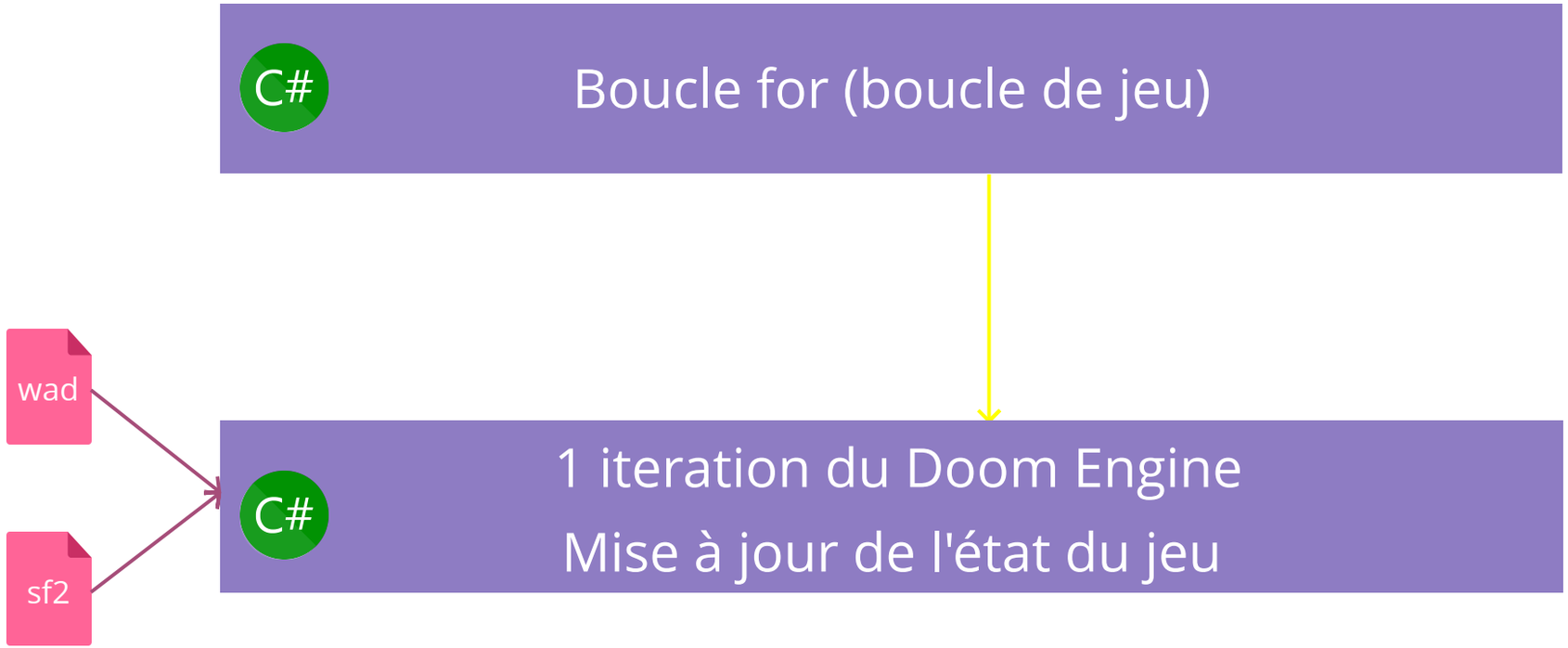
```
function gameLoop(){
  if (canAdvanceFrame()){
    const input = getPlayerInput();
    const { frame, audio }
              = UpdateGameState(input, WAD);
    render(frame);
    play(audio);
  }
  requestAnimationFrame(gameLoop);
}
requestAnimationFrame(gameLoop);
```

```
function gameLoop(){
  if (canAdvanceFrame()){
    const input = getPlayerInput();
    const { frame, audio }
              = UpdateGameState(input, WAD);
    render(frame);
    play(audio);
  }
  requestAnimationFrame(gameLoop);
}
requestAnimationFrame(gameLoop);
```

```
function gameLoop(){
  if (canAdvanceFrame()){
    const input = getPlayerInput();
    const { frame, audio }
              = UpdateGameState(input, WAD);
    render(frame);
    play(audio);
  }
  requestAnimationFrame(gameLoop);
}
requestAnimationFrame(gameLoop);
```

Boucle for (boucle de jeu)


C#
Boucle for (boucle de jeu)
wad
sf2
C#
1 iteration du Doom Engine
Mise à jour de l'état du jeu


C#
Boucle for (boucle de jeu)
wad
sf2
C#
1 iteration du Doom Engine
Mise à jour de l'état du jeu

C# Boucle for (boucle de jeu)

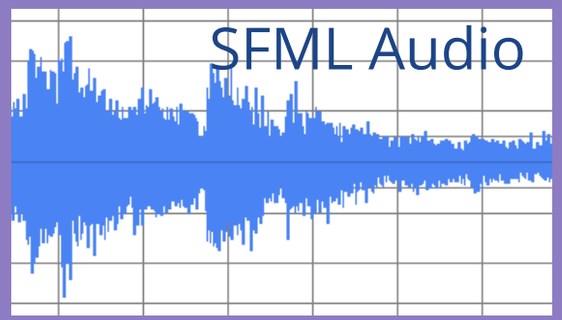
wad

sf2

C# 1 iteration du Doom Engine

Mise à jour de l'état du jeu

C#



C# Boucle for (boucle de jeu)

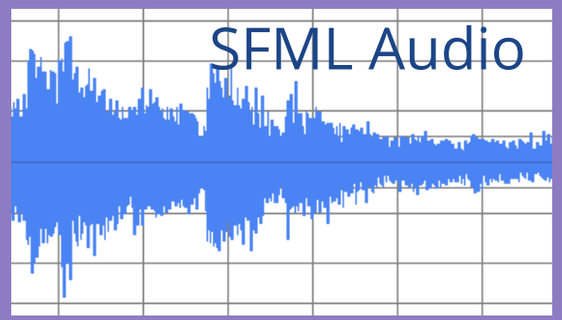
wad

sf2

C# 1 iteration du Doom Engine

Mise à jour de l'état du jeu

C#



C# Boucle de jeu (.NET WASM)

wad

sf2

C# 1 iteration du Doom Engine
Mise à jour de l'état du jeu

~70% OK

C# SFML Audio .NET WASM

JS

requestAnimationFrame

Appel .Net / WASM

wad

sf2

C#

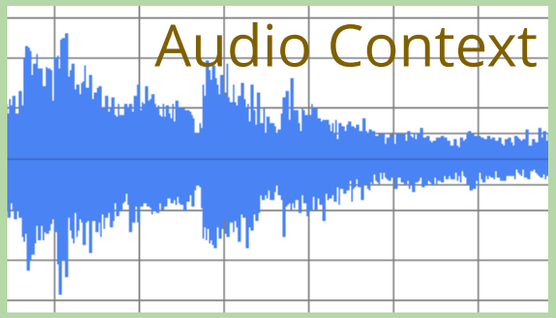
1 iteration du Doom Engine

Mise à jour de l'état du jeu

Appel JS

Appel JS

JS



Knee-deep in the code

Chapter 4

Knee-deep in the code

Point d'entrée

```html
<html>
  <head>
    <!-- Sets .Net interop and starts the game loop -->
    <script type="module" src="./main.js"></script>
   </head>
  <body>
    <canvas id="canvas" width="320" height="200"
            style="image-rendering: pixelated" />
  </body>
</html>
```

Point d'entrée

```html
<html>
  <head>
    <!-- Sets .Net interop and starts the game loop -->
    <script type="module" src="./main.js"></script>
  </head>
  <body>
    <canvas id="canvas" width="320" height="200"
            style="image-rendering: pixelated" />
  </body>
</html>
```

Point d'entrée

```html
<html>
  <head>
    <!-- Sets .Net interop and starts the game loop -->
    <script type="module" src="./main.js"></script>
   </head>
  <body>
    <canvas id="canvas" width="320" height="200"
            style="image-rendering: pixelated" />
  </body>
</html>
```

Exécution de la boucle de jeu

main.js

```
1  import { dotnet } from "./dotnet.js";
2  const { getAssemblyExports, getConfig } = await dotnet.create();
3  const exports = await getAssemblyExports(getConfig().mainAssemblyName);
4  await dotnet.run();
5
6  function gameLoop(timestamp) {
7      if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8          lastFrameTimestamp = timestamp;
9          exports.BlazorDoom.MainJS.UpdateGameState(keys);
10     }
11     requestAnimationFrame(gameLoop);
12 }
```

JS

Exécution de la boucle de jeu

main.js

```js
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Exécution de la boucle de jeu

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Exécution de la boucle de jeu

main.js

```
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

Exécution de la boucle de jeu

main.js

```
1  import { dotnet } from "./dotnet.js";
2  const { getAssemblyExports, getConfig } = await dotnet.create();
3  const exports = await getAssemblyExports(getConfig().mainAssemblyName);
4  await dotnet.run();
5
6  function gameLoop(timestamp) {
7      if (timestamp - lastFrameTimestamp >= 1000 / 30) {
8          lastFrameTimestamp = timestamp;
9          exports.BlazorDoom.MainJS.UpdateGameState(keys);
10     }
11     requestAnimationFrame(gameLoop);
12 }
```

JS

Exécution de la boucle de jeu

main.js

```js
import { dotnet } from "./dotnet.js";
const { getAssemblyExports, getConfig } = await dotnet.create();
const exports = await getAssemblyExports(getConfig().mainAssemblyName);
await dotnet.run();

function gameLoop(timestamp) {
    if (timestamp - lastFrameTimestamp >= 1000 / 30) {
        lastFrameTimestamp = timestamp;
        exports.BlazorDoom.MainJS.UpdateGameState(keys);
    }
    requestAnimationFrame(gameLoop);
}
```

JS

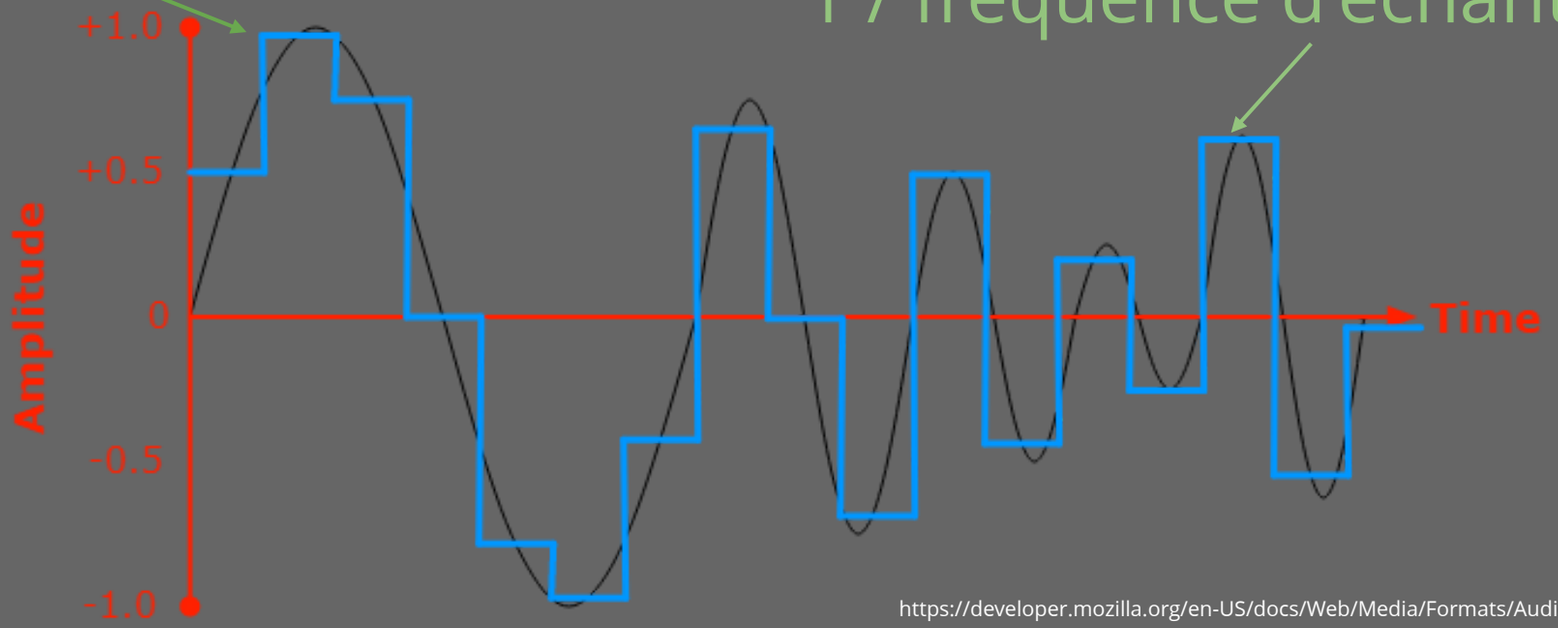
```cs
public partial class MainJS // this name is required
{
  public static void Main()
  {
    app = new ManagedDoom.DoomApplication();
  }

  [JSExport] // Can be imported from JS
  public static void UpdateGameState(int[] keys)
  { // computes the next frame and sounds
     managedDoom.UpdateGameState(keys);
  }
}
```

C#

Audio

Echantillon

1 / fréquence d'échantillonage

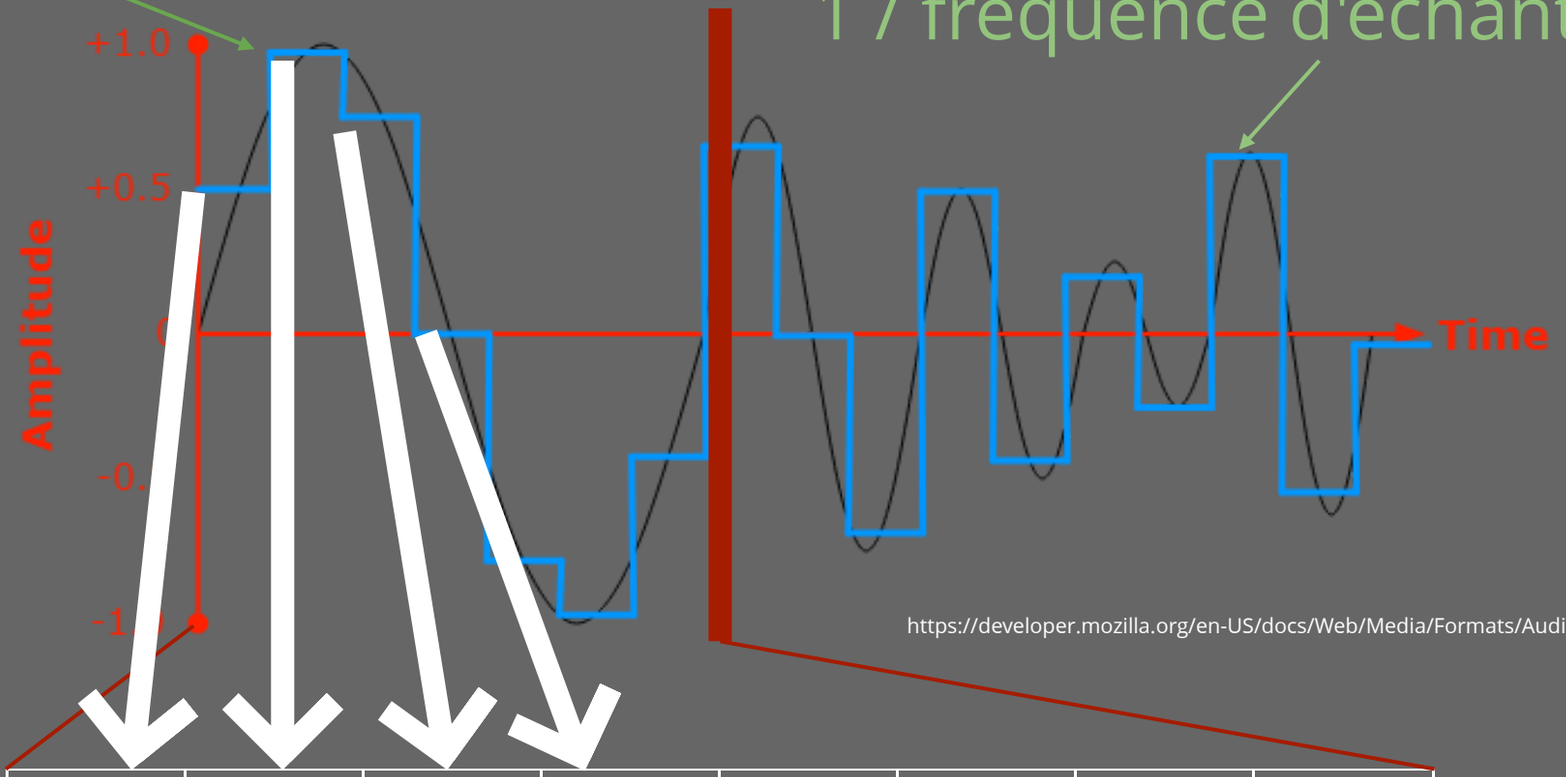


https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

Audio

Echantillon

1 / fréquence d'échantillonage



https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

C#

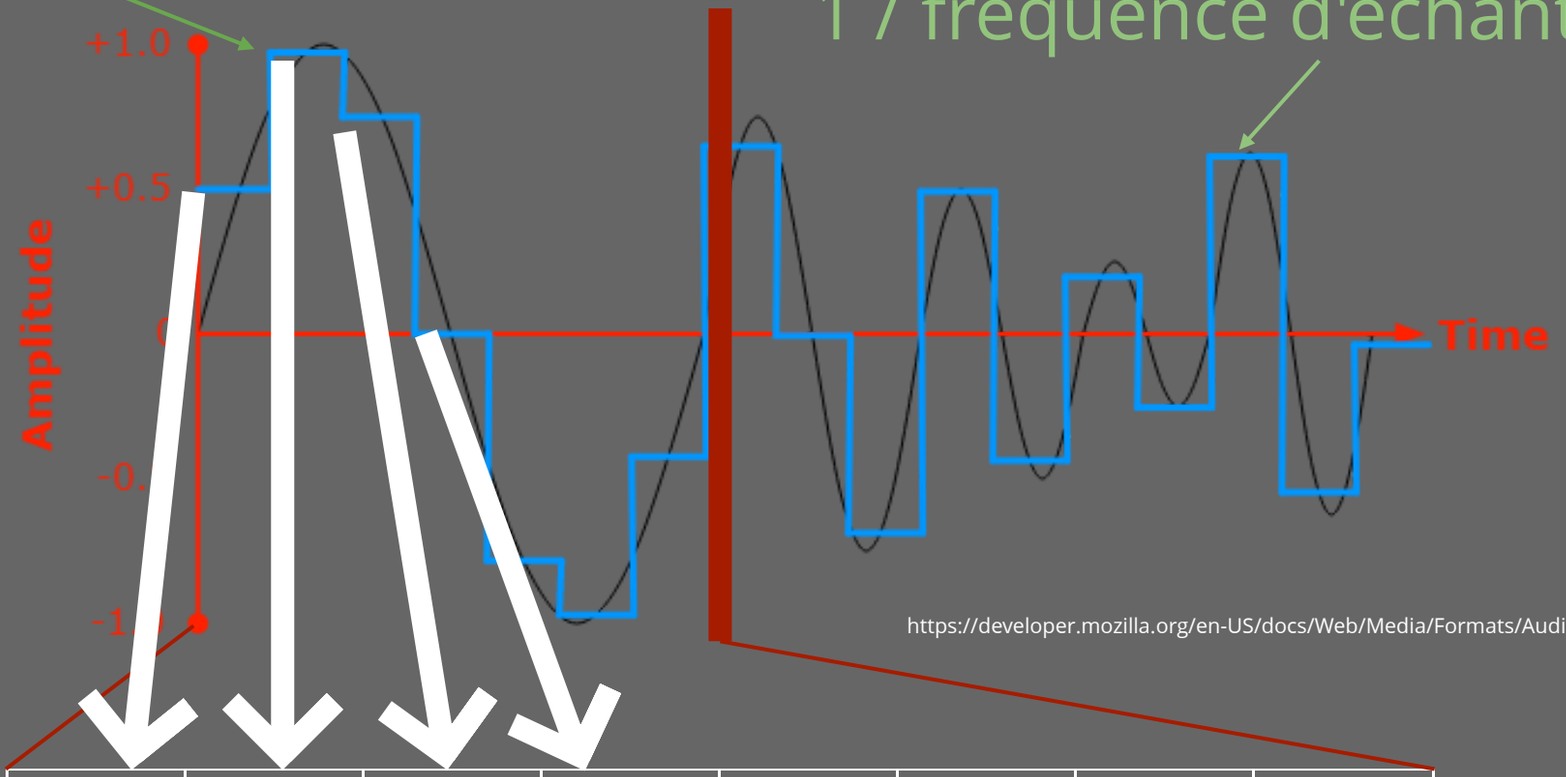
0.5	1	0.75	0	-0.75	-1	-0.5	0

+ Fréquence d'échantillonage

Audio

Echantillon

1 / fréquence d'échantillonage



https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

C#

0.5	1	0.75	0	-0.75	-1	-0.5	0

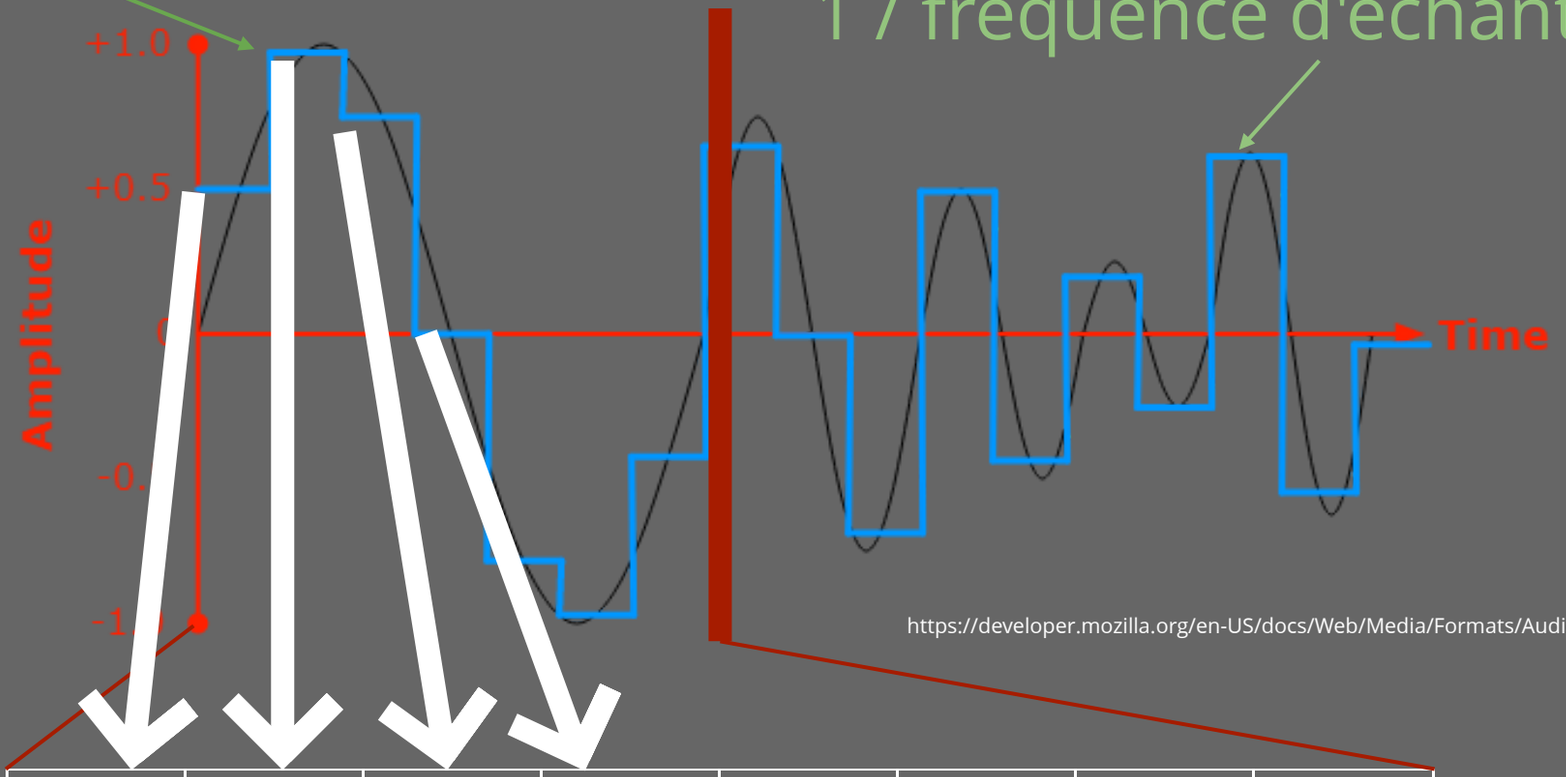
+ Fréquence d'échantillonage

JS AudioContext

Audio

Echantillon

1 / fréquence d'échantillonage



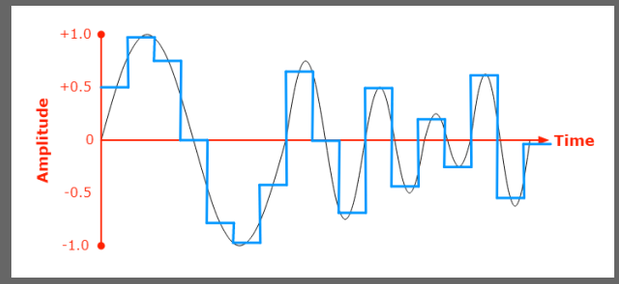
https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts

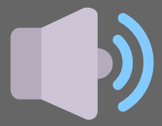
C#

0.5	1	0.75	0	-0.75	-1	-0.5	0

+ Fréquence d'échantillonage

JS AudioContext





Gestion des bruitages

```
1 void PlayCurrentFrameSound(SoundBuffer soundBuffer)
2 {
3     int[] samples = Array.ConvertAll(soundBuffer.samples, Convert.ToInt32);
4     BlazorDoom.Renderer.playSoundOnJS(samples, (int)soundBuffer.sampleRate);
5 }
```

C#

```
 1 namespace BlazorDoom
 2 {
 3     [SupportedOSPlatform("browser")]
 4     public partial class Renderer
 5     {
 6         [JSImport("playSound", "blazorDoom/renderer.js")]
 7         internal static partial string playSoundOnJS(
 8             int[] samples,
 9             int sampleRate
10         );
11     }
12 }
```

C#

Gestion des bruitages

```
export function playSound(samples, sampleRate) {
    audioContext = new AudioContext({
        sampleRate: sampleRate,
    });
    const length = samples.length;
    const audioBuffer = audioContext.createBuffer(
        1,
        length,
        sampleRate
    );

    var channelData = audioBuffer.getChannelData(0);
    for (let i = 0; i < length; i++) {
        // noralize the sample to be between -1 and 1
        channelData[i] = samples[i] / 0xffff;
    }

    var source = audioContext.createBufferSource();
    source.buffer = audioBuffer;
    source.connect(audioContext.destination);
    source.start();
}
```

JS

Gestion des bruitages

```
export function playSound(samples, sampleRate) {
    audioContext = new AudioContext({
        sampleRate: sampleRate,
    });
    const length = samples.length;
    const audioBuffer = audioContext.createBuffer(
        1,
        length,
        sampleRate
    );

    var channelData = audioBuffer.getChannelData(0);
    for (let i = 0; i < length; i++) {
        // noralize the sample to be between -1 and 1
        channelData[i] = samples[i] / 0xffff;
    }

    var source = audioContext.createBufferSource();
    source.buffer = audioBuffer;
    source.connect(audioContext.destination);
    source.start();
}
```

JS

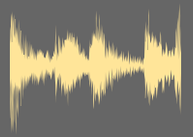
Gestion des bruitages

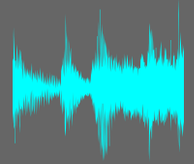
```
export function playSound(samples, sampleRate) {
    audioContext = new AudioContext({
        sampleRate: sampleRate,
    });
    const length = samples.length;
    const audioBuffer = audioContext.createBuffer(
        1,
        length,
        sampleRate
    );

    var channelData = audioBuffer.getChannelData(0);
    for (let i = 0; i < length; i++) {
        // noralize the sample to be between -1 and 1
        channelData[i] = samples[i] / 0xffff;
    }

    var source = audioContext.createBufferSource();
    source.buffer = audioBuffer;
    source.connect(audioContext.destination);
    source.start();
}
```

JS

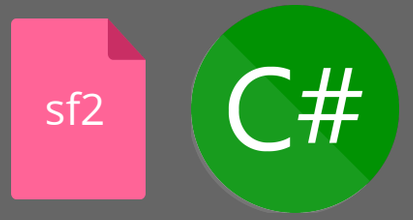
Musique

SF2: format qui définit quel son que doit émettre chaque note

Note 0 

Note 1 

Streaming de la musique

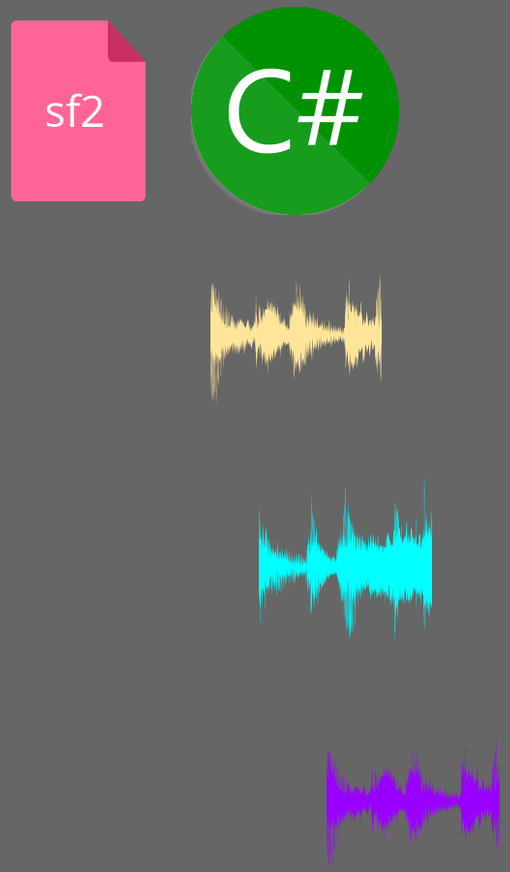


Streaming de la musique

Temps

Streaming de la musique

Streaming de la musique



Streaming de la musique

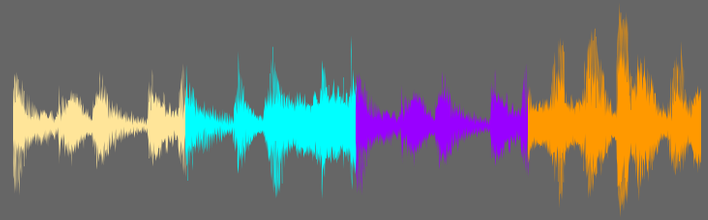
Streaming de la musique

Streaming de la musique

sf2

C#

JS



Temps

Temps

Streaming de la musique

sf2

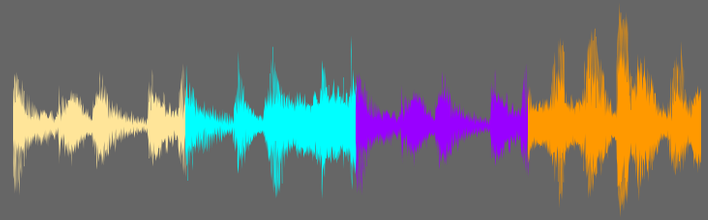
C#

JS

AudioContext 😢

émet des **glitchs** sur des petits extraits

+ ne gère pas le streaming **nativement**



Temps

Temps

Streaming de la musique

sf2

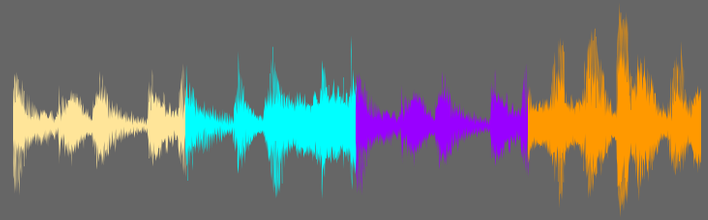
C#

JS

AudioContext 😢

émet des **glitchs** sur des petits extraits

+ ne gère pas le streaming **nativement**



Solution 🎶🥳

Regrouper les extraits en un tampon assez grand

+ planifier **le moment** de lancement

Streaming de la musique

```
function playMusic(samples) {
    // On lit le tampon audio s'il est assez rempli
    if (this.#currentMusicBufferIndex >= this.#musicBuffer.length) {
      const currentTime = this.#audioContext.currentTime;
      const duration = this.#currentMusicBufferIndex / this.musicSampleRate;
      // On planifie l'extrait pour qu'il se lance après le précédent
      source.start(this.expectedBufferEndTime, 0, duration);
      this.expectedBufferEndTime = currentTime + duration;
    }
    // Remplissage du tampon avec l'extrait courant
    for (let i = 0; i < samples.length; i++) {
      this.#currentChannelData[this.#currentMusicBufferIndex + i]
        = samples[i] / 32767;
    }
    this.#currentMusicBufferIndex += samples.length;
}
```

Inconvénient : la musique se lance avec un retard

(le temps de remplir le premier tampon)

Streaming de la musique

```
 1 function playMusic(samples) {
 2     // On lit le tampon audio s'il est assez rempli
 3     if (this.#currentMusicBufferIndex >= this.#musicBuffer.length) {
 4       const currentTime = this.#audioContext.currentTime;
 5       const duration = this.#currentMusicBufferIndex / this.musicSampleRate;
 6       // On planifie l'extrait pour qu'il se lance après le précédent
 7       source.start(this.expectedBufferEndTime, 0, duration);
 8       this.expectedBufferEndTime = currentTime + duration;
 9     }
10     // Remplissage du tampon avec l'extrait courant
11     for (let i = 0; i < samples.length; i++) {
12       this.#currentChannelData[this.#currentMusicBufferIndex + i]
13         = samples[i] / 32767;
14     }
15     this.#currentMusicBufferIndex += samples.length;
16 }
```

Inconvénient : la musique se lance avec un retard

(le temps de remplir le premier tampon)

Streaming de la musique

```
function playMusic(samples) {
    // On lit le tampon audio s'il est assez rempli
    if (this.#currentMusicBufferIndex >= this.#musicBuffer.length) {
      const currentTime = this.#audioContext.currentTime;
      const duration = this.#currentMusicBufferIndex / this.musicSampleRate;
      // On planifie l'extrait pour qu'il se lance après le précédent
      source.start(this.expectedBufferEndTime, 0, duration);
      this.expectedBufferEndTime = currentTime + duration;
    }
    // Remplissage du tampon avec l'extrait courant
    for (let i = 0; i < samples.length; i++) {
      this.#currentChannelData[this.#currentMusicBufferIndex + i]
        = samples[i] / 32767;
    }
    this.#currentMusicBufferIndex += samples.length;
}
```

Inconvénient : la musique se lance avec un retard

(le temps de remplir le premier tampon)

Rendu des images

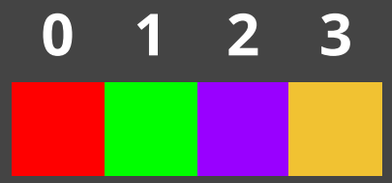
tableau à 1 dimension + palette de couleurs

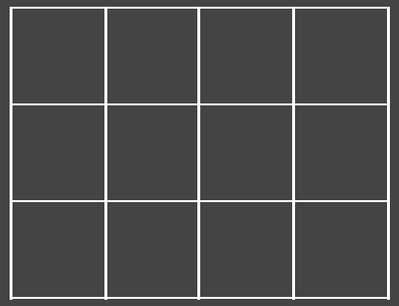
Image

0	1	2	3	1	1	2	3	2	2	0	1

chaque pixel contient
l'id de la couleur

Palette de couleurs





Canvas

12 * 4 bytes (r, g, b, a)

Rendu des images

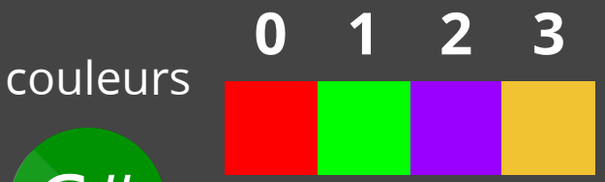
tableau à 1 dimension + palette de couleurs

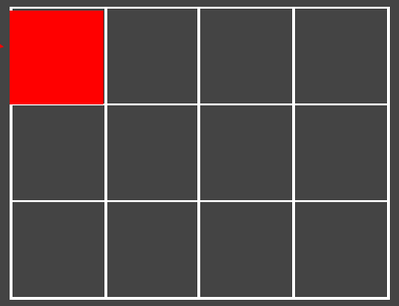
Image

0	1	2	3	1	1	2	3	2	2	0	1

chaque pixel contient
l'id de la couleur

Palette de couleurs





Canvas

12 * 4 bytes (r, g, b, a)

Rendu des images

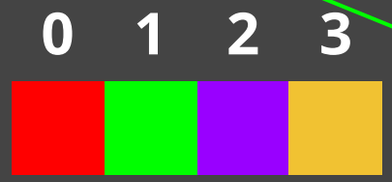
tableau à 1 dimension + palette de couleurs

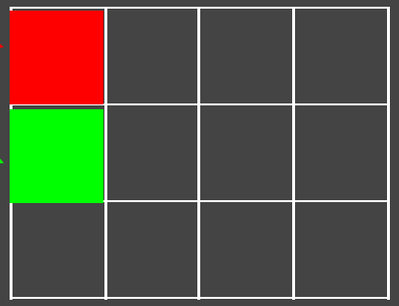
Image

0	1	2	3	1	1	2	3	2	2	0	1

chaque pixel contient
l'id de la couleur

Palette de couleurs





Canvas

12 * 4 bytes (r, g, b, a)

Rendu des images

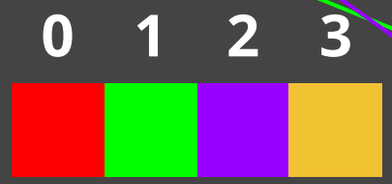
tableau à 1 dimension + palette de couleurs

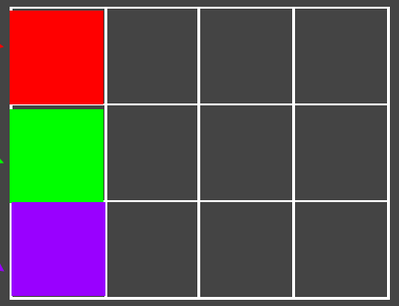
Image

0	1	2	3	1	1	2	3	2	2	0	1

chaque pixel contient
l'id de la couleur

Palette de couleurs



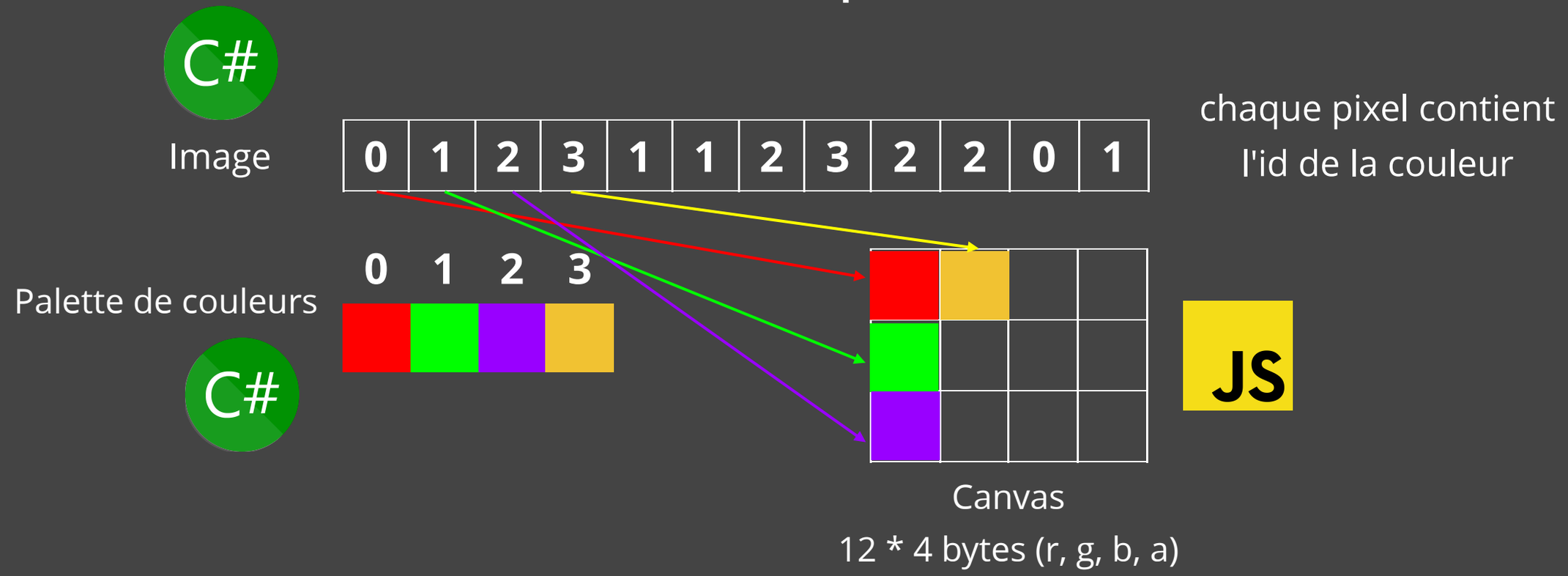


Canvas

12 * 4 bytes (r, g, b, a)

Rendu des images

tableau à 1 dimension + palette de couleurs



Rendu des images

tableau à 1 dimension + palette de couleurs

C#

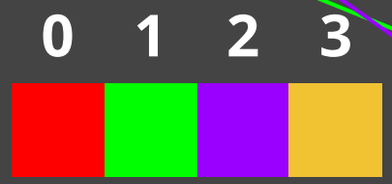
Image

0	1	2	3	1	1	2	3	2	2	0	1

chaque pixel contient
l'id de la couleur

Palette de couleurs

C#



JS

Canvas

12 * 4 bytes (r, g, b, a)

Rendu des images

tableau à 1 dimension + palette de couleurs

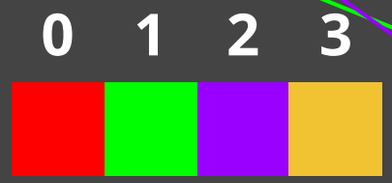
C#

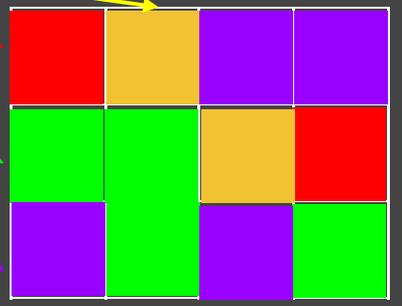
Image

chaque pixel contient
l'id de la couleur

Palette de couleurs

C#





JS

Canvas

12 * 4 bytes (r, g, b, a)

Pixels distribués de haut en bas puis et de gauche à droite

Transmission des pixels du C# au JS

```C#
public class DoomRenderer
{
  // Called by updateGameState
  private void Display(uint[] colors)
  {
    BlazorDoom.Renderer.renderOnJS(screen.Data, (int[])((object)colors));
  }
}
```


```C#
namespace BlazorDoom
{
    [SupportedOSPlatform("browser")]
    public partial class Renderer
    {
        [JSImport("drawOnCanvas", "blazorDoom/renderer.js")]
        internal static partial string renderOnJS(byte[] screenData,
                                                  int[] colors);
    }
}
```


```JS
export function drawOnCanvas(screenData, colors) {
  //
}
```

Remplissage du canvas

```
1  export function drawOnCanvas(screenData, colors) {
2      const context = getCanvas().context;
3      const imageData = context.createImageData(320, 200);
4      let y = 0;
5      let x = 0;
6      for (let i = 0; i < screenData.length; i += 1) {
7          const dataIndex = (y * width + x) * 4;
8          setSinglePixel(imageData, dataIndex, colors, screenData[i]);
9          if (y >= height - 1) {
10             y = 0;
11             x += 1;
12         } else {
13             y += 1;
14         }
15     }
16     context.putImageData(imageData, 0, 0);
17 }
18
19 function setSinglePixel(imageData, dataIndex, colors, colorIndex) {
20     const color = colors[colorIndex];
21     imageData.data[dataIndex] = color & 0xff; // R
22     imageData.data[dataIndex + 1] = (color >> 8) & 0xff; // G
23     imageData.data[dataIndex + 2] = (color >> 16) & 0xff; // B
24     imageData.data[dataIndex + 3] = 255; // Alpha
25 }
```

JS

Remplissage du canvas

```
export function drawOnCanvas(screenData, colors) {
    const context = getCanvas().context;
    const imageData = context.createImageData(320, 200);
    let y = 0;
    let x = 0;
    for (let i = 0; i < screenData.length; i += 1) {
        const dataIndex = (y * width + x) * 4;
        setSinglePixel(imageData, dataIndex, colors, screenData[i]);
        if (y >= height - 1) {
            y = 0;
            x += 1;
        } else {
            y += 1;
        }
    }
    context.putImageData(imageData, 0, 0);
}

function setSinglePixel(imageData, dataIndex, colors, colorIndex) {
    const color = colors[colorIndex];
    imageData.data[dataIndex] = color & 0xff; // R
    imageData.data[dataIndex + 1] = (color >> 8) & 0xff; // G
    imageData.data[dataIndex + 2] = (color >> 16) & 0xff; // B
    imageData.data[dataIndex + 3] = 255; // Alpha
}
```

JS

Remplissage du canvas

```
1  export function drawOnCanvas(screenData, colors) {
2      const context = getCanvas().context;
3      const imageData = context.createImageData(320, 200);
4      let y = 0;
5      let x = 0;
6      for (let i = 0; i < screenData.length; i += 1) {
7          const dataIndex = (y * width + x) * 4;
8          setSinglePixel(imageData, dataIndex, colors, screenData[i]);
9          if (y >= height - 1) {
10             y = 0;
11             x += 1;
12         } else {
13             y += 1;
14         }
15     }
16     context.putImageData(imageData, 0, 0);
17 }
18
19 function setSinglePixel(imageData, dataIndex, colors, colorIndex) {
20     const color = colors[colorIndex];
21     imageData.data[dataIndex] = color & 0xff; // R
22     imageData.data[dataIndex + 1] = (color >> 8) & 0xff; // G
23     imageData.data[dataIndex + 2] = (color >> 16) & 0xff; // B
24     imageData.data[dataIndex + 3] = 255; // Alpha
25 }
```

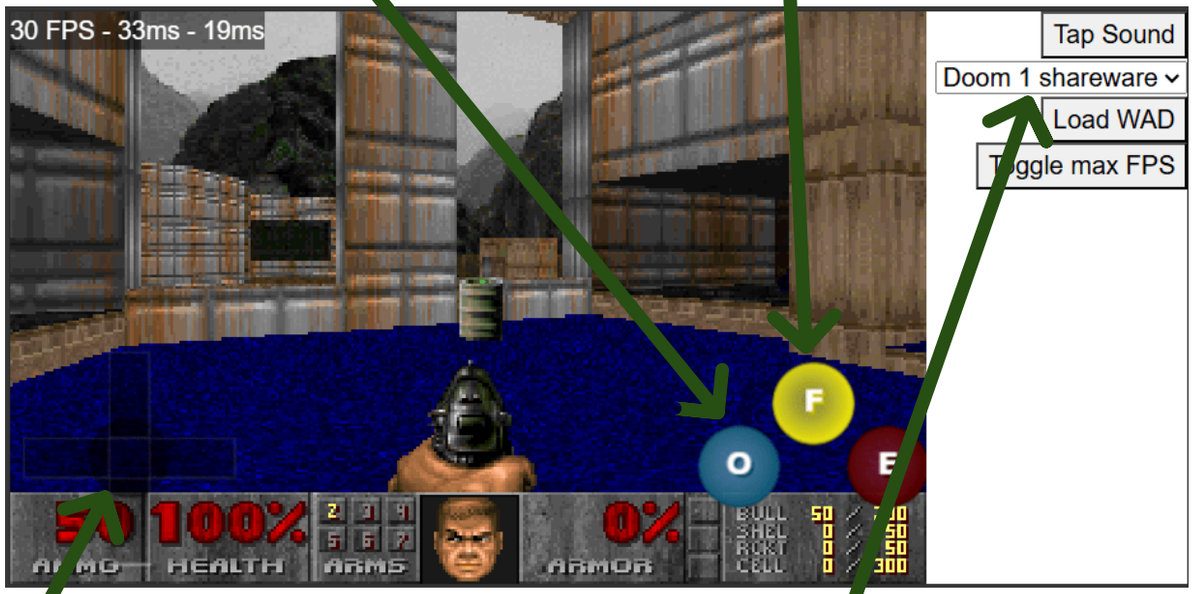
JS


CHAMBERS
DEMON

Fire / Tirer

Valider / validate

Open / Ouvrir



Se déplacer

Sélectionner
un autre WAD

EPILOGUE

CONCLUSION


HAVING FUN?

WORLDLINE


WORLDLINE


WORLDLINE