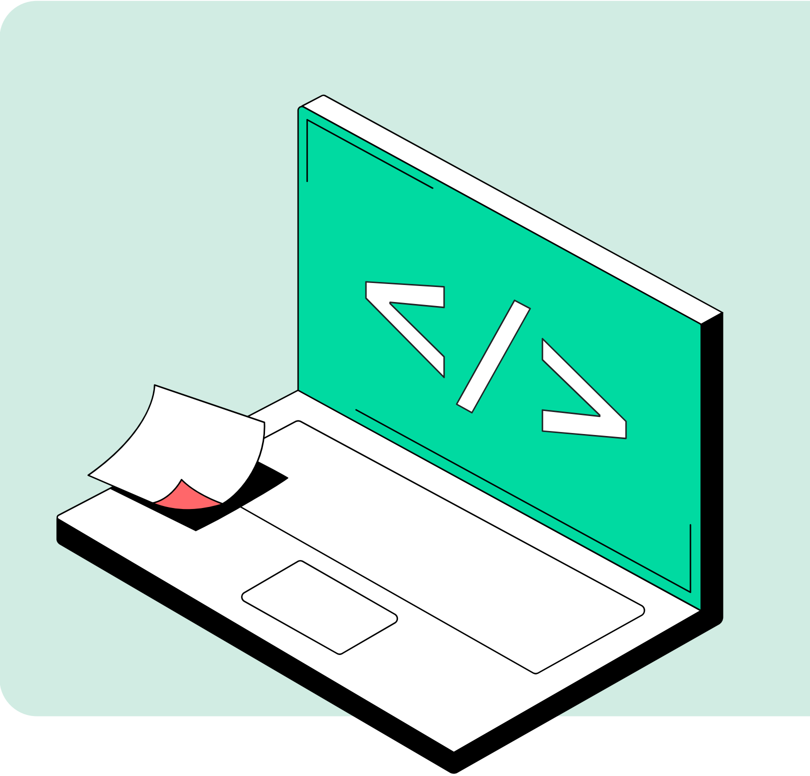
Women Techmakers
Taipei

WebCodecs + WebGPU

開啟個人化串流新視界



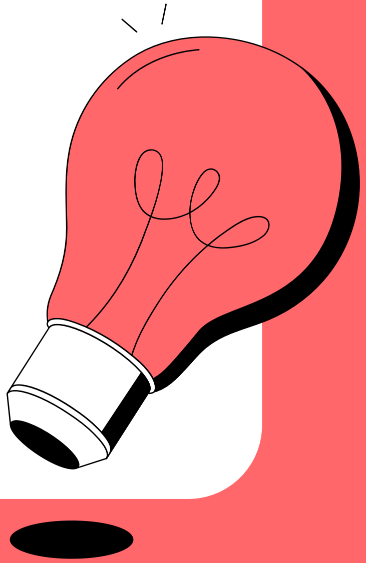
Claire Chang
Xuenn / IT Consultant
2024/11/30

Women Techmakers
Taipei

關於我

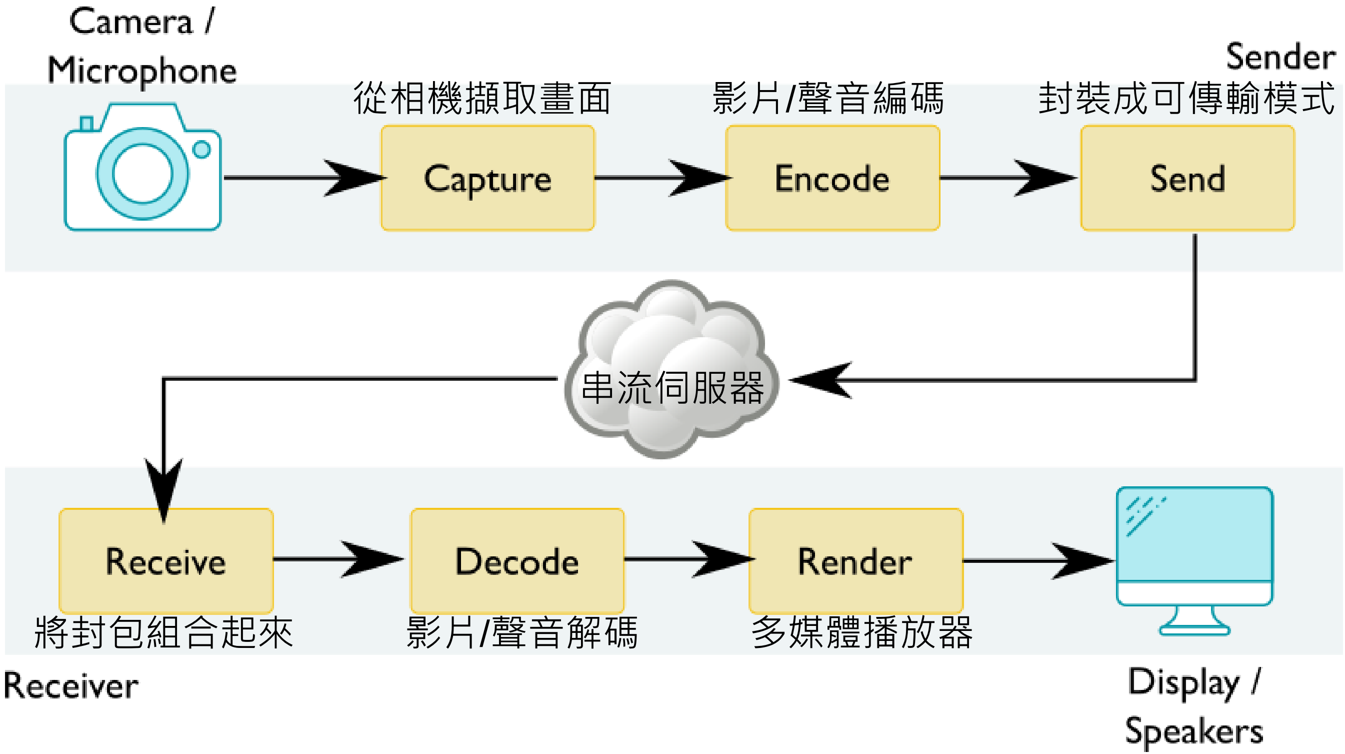
- ✓ Flash遊戲開發者
- ✓ H5網頁遊戲前端開發
- ✓ .Net, PHP等後端程式開發
- ✓ 串流播放器、伺服器開發
- ✓ OpenCV影像辨識程式開發
- ✓ 使用Tensorflow、YOLO等工具做影像辨識

直播概念與流程

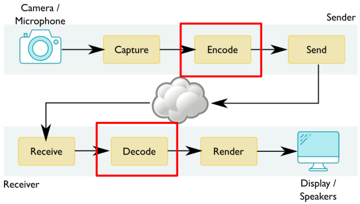


線上會議、網路串流的基礎概念

視訊通話的流程

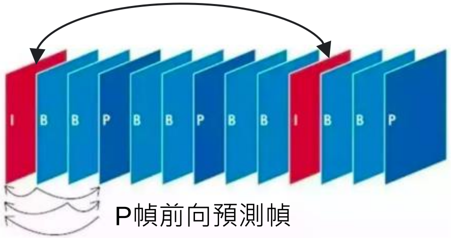

Camera /
Microphone
Sender
從相機擷取畫面
Capture
影片/聲音編碼
Encode
封裝成可傳輸模式
Send
串流伺服器
Receive
將封包組合起來
Decode
影片/聲音解碼
Render
多媒體播放器
Receiver
Display /
Speakers

什麼是編碼



了解I, P, B幀及GOP

GOP (Group of Pictures)



P幀前向預測幀

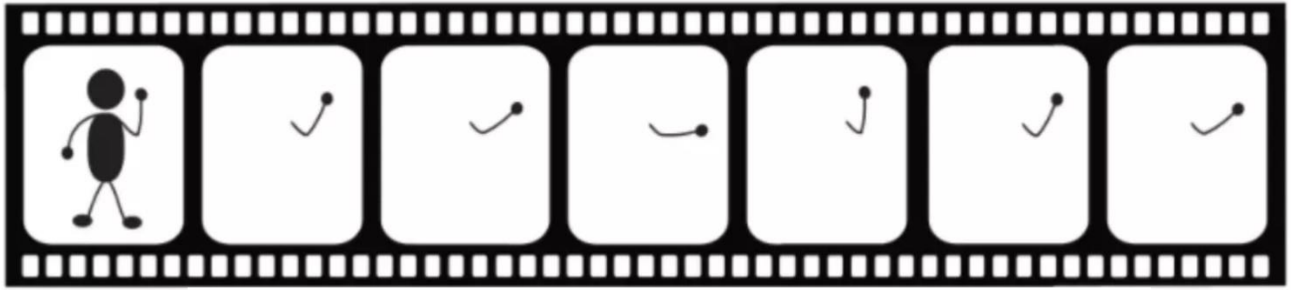
B幀雙向預測幀

I 幀獨立編碼幀

原始影片: 每一個影格都是完整的圖片 1920x1080（Full HD），30 FPS，一分鐘可達10.8 GB

影像壓縮

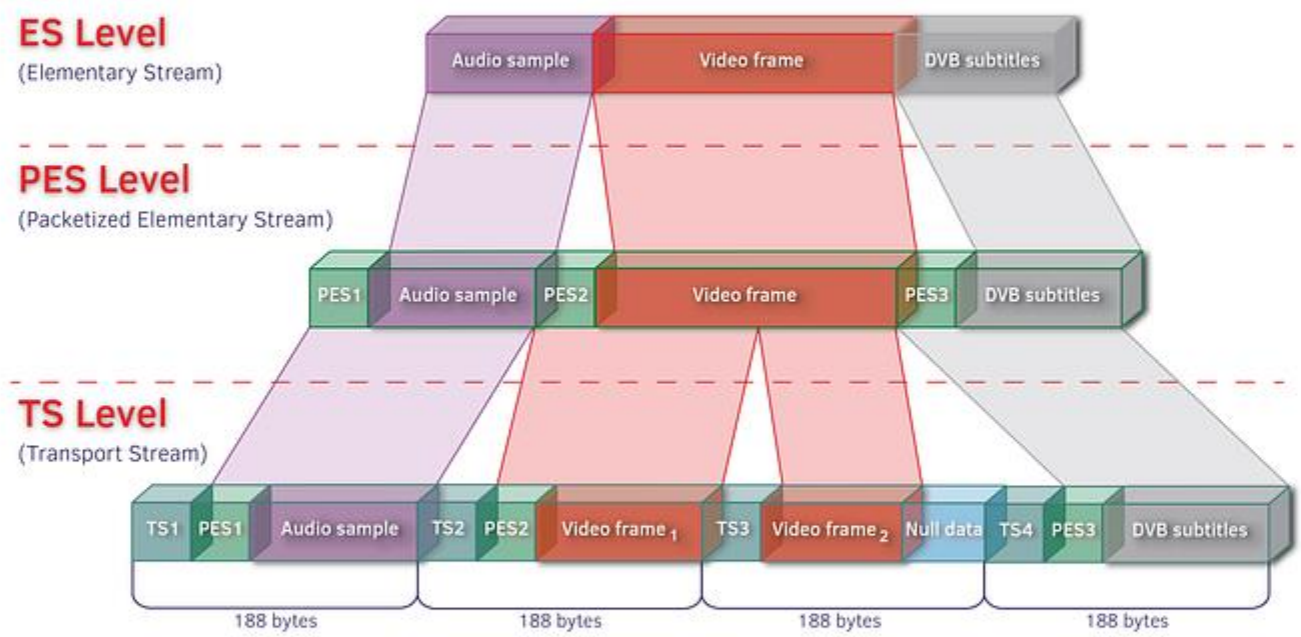
藉由去除時間、空間的冗餘、或使用用短的編碼記錄常見數據、去掉人眼無法察覺的細節，來壓縮影片大小。



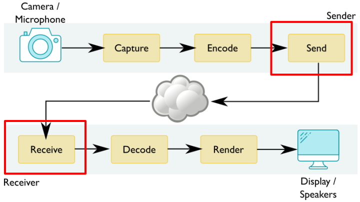
編碼過後可大幅壓縮影片大小 靜態場景僅為原始影片1~2%，動態場景約5~10%大小

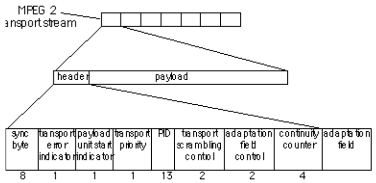
什麼是封裝

把影片的聲音、影片、字幕、編碼方式、色彩模式等資訊打包儲存起來，讓解碼器及播放器能夠順利解碼並播放影音



MPEG-2: Understanding the Transport Stream Structure



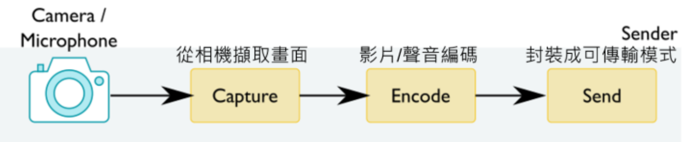


sync byte	transport error indicator	payload unit start indicator	transport priority	PID	transport scrambling control	adaptation field control	continuity counter	adaptation field
8	1	1	1	13	2	2	4	

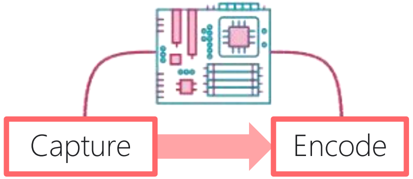
封裝的模式若要支持流式傳輸，應要能支持分塊傳輸、同步音視頻、並具備錯誤容忍機制

Women Techmakers
Taipei

把影像推流到伺服器

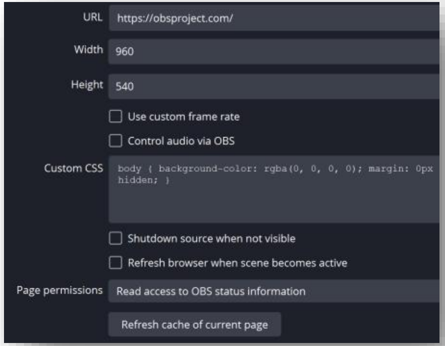


在此做影像處理

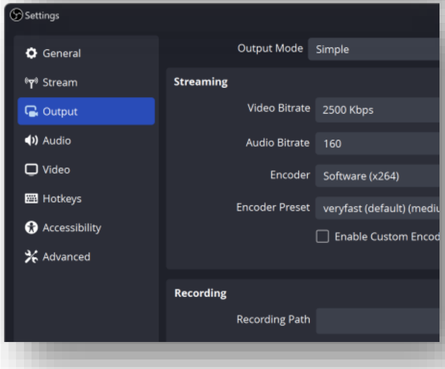


OBS
Open Broadcaster Software®

這是一款免費且開源的直播與錄影軟體，廣泛應用於遊戲實況、教學錄製、會議轉播等領域。



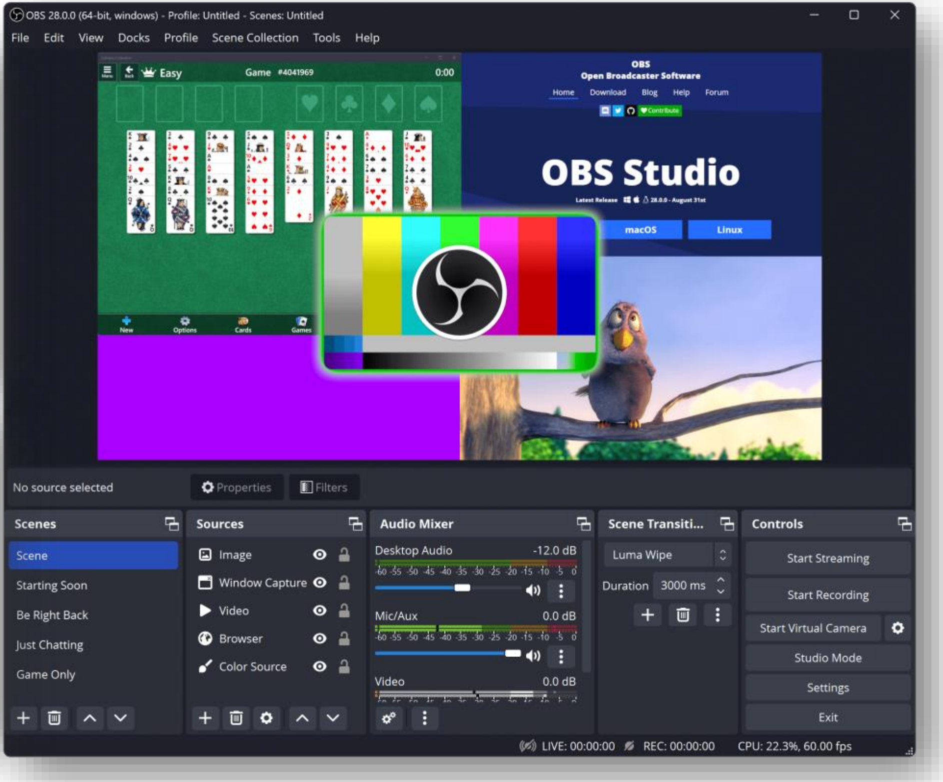
URL: https://obsproject.com/
Width: 960
Height: 540
☐ Use custom frame rate
☐ Control audio via OBS
Custom CSS: body { background-color: rgba(0, 0, 0, 0); margin: 0px
hidden; }
☐ Shutdown source when not visible
☐ Refresh browser when scene becomes active
Page permissions: Read access to OBS status information



Output Mode: Simple

Streaming
Video Bitrate: 2500 Kbps
Audio Bitrate: 160
Encoder: Software (x264)
Encoder Preset: veryfast (default) (medi
☐ Enable Custom Encod

Recording
Recording Path

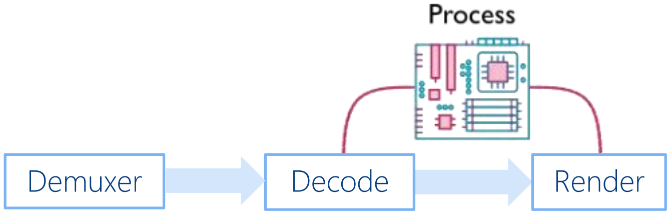

OBS 28.0.0 (64-bit, windows) - Profile: Untitled - Scenes: Untitled

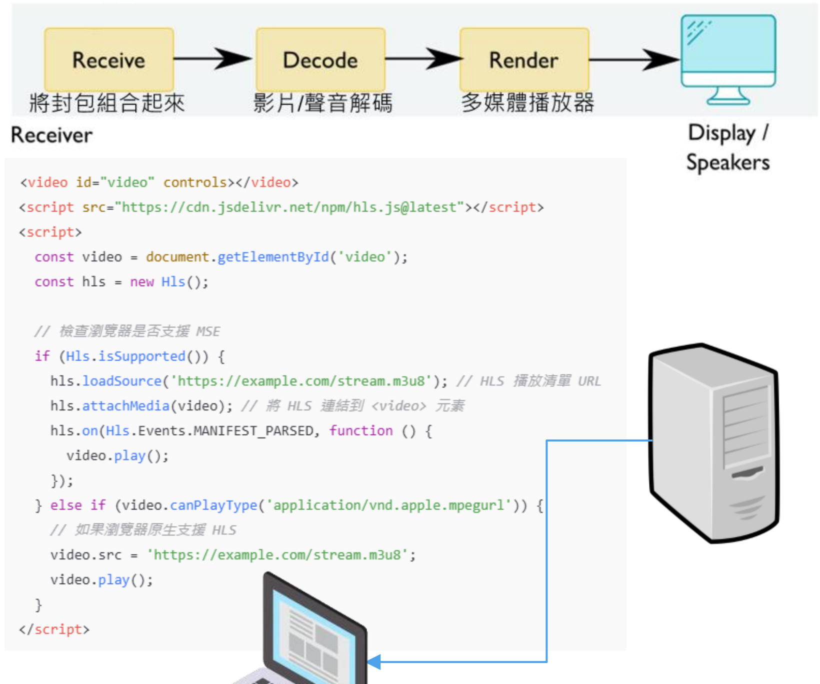
Desktop Audio -12.0 dB
Mic/Aux 0.0 dB
Video 0.0 dB
Duration 3000 ms
LIVE: 00:00:00 REC: 00:00:00 CPU: 22.3%, 60.00 fps

使用網頁從伺服器拉流

傳統在網頁裡操控影片的技術	
HTMLMediaElement	用於控制音樂和影片的 HTML 介面
WebRTC	網頁中即時音視頻通訊的技術
getUserMedia	抓取使用者裝置的相機和麥克風
Media Source Extensions	動態串流媒體的 API

HTMLMediaElement、WebRTC或MSE，
都無法讓我們在Demuxer > Render之間插入使用者自訂的動作





```
<video id="video" controls></video>
<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
<script>
  const video = document.getElementById('video');
  const hls = new Hls();

  // 檢查瀏覽器是否支援 MSE
  if (Hls.isSupported()) {
    hls.loadSource('https://example.com/stream.m3u8'); // HLS 播放清單 URL
    hls.attachMedia(video); // 將 HLS 連結到 <video> 元素
    hls.on(Hls.Events.MANIFEST_PARSED, function () {
      video.play();
    });
  } else if (video.canPlayType('application/vnd.apple.mpegurl')) {
    // 如果瀏覽器原生支援 HLS
    video.src = 'https://example.com/stream.m3u8';
    video.play();
  }
</script>
```

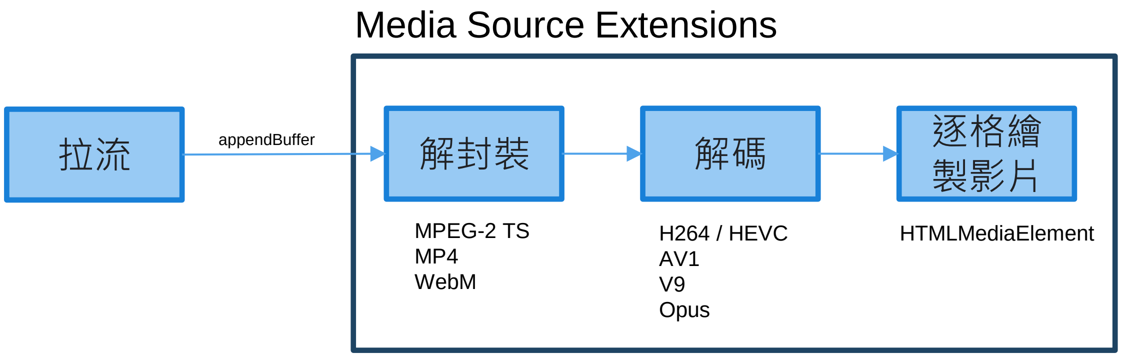
HLS（HTTP Live Streaming）的
平均延遲通常介於**10秒到30秒**之間

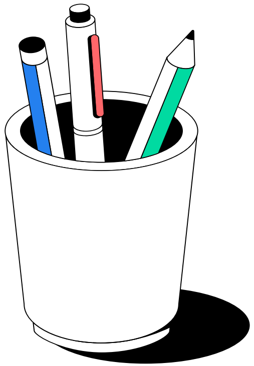
Women Techmakers
Taipei

難以滿足網頁互動遊戲的低延遲需求

- WebM (DASH)以及MPEG-2 TS(HLS) 都是將影片切成固定長度片段，會造成較高的延遲

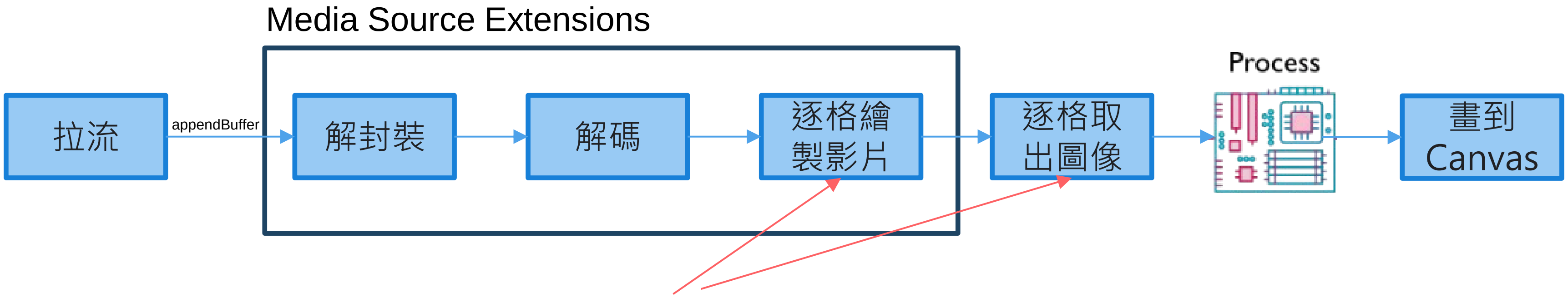
- 過去網頁裡操控多媒體的技術都是將多個流程封裝起來，因此只要其中一個部分不支持(如解封裝)，其他部分也無法使用(如H264)





難以滿足瀏覽器端修改影片內容的需求

如果真的需要修改，需要先將影片畫出來，再逐格取出影片來做修改...



Media Source Extensions

拉流

appendBuffer

解封裝

解碼

逐格繪製影片

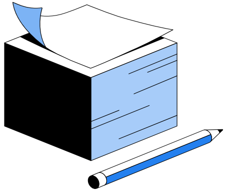
逐格取出圖像

Process

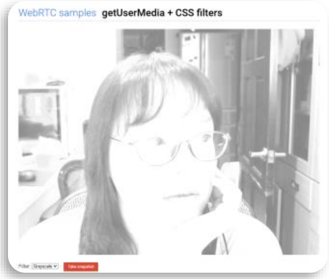
畫到Canvas

一般網路的影片的FPS約為30~60之間，電影則是24FPS

這樣的處理方式在效能上有非常大的問題



針對網頁的HTMLMediaElement元素
增加CSS filter
{filter: grayscale(1);}



WebRTC samples getUserMedia + CSS filters

Women Techmakers
Taipei

許多網站只能夠自己使用

JavaScript + WebAssembly

來滿足現有元件無法滿足的需求

缺點：

1. 技術困難度極高
2. 效能難以與瀏覽器原生功能相比

三大關鍵技術
實現即時串流瀏覽器端的內容編輯

WebCodecs

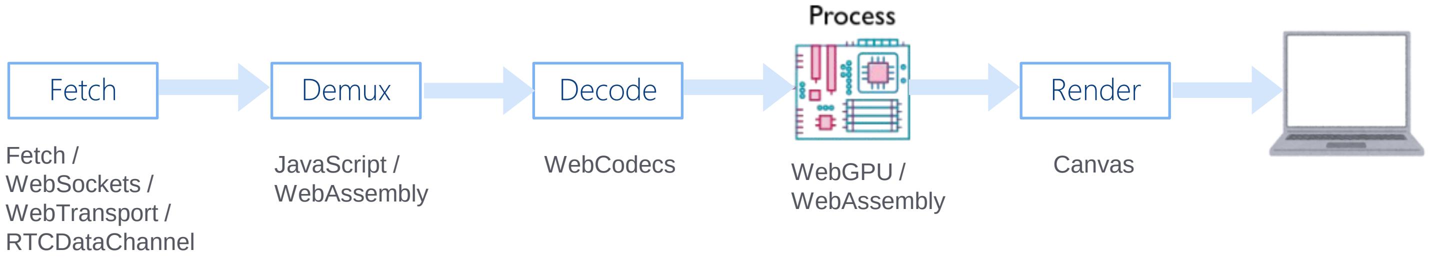
一個低階的JavaScript API，提供對媒體編解碼的直接存取。它允許開發者在瀏覽器中直接操作視頻和音訊的原始數據，而無需依賴於高階的媒體元素。

WebAssembly

一種高效的二進位格式，可以在現代網頁瀏覽器中執行，並且能夠以接近原生代碼的速度運行。它可以用於編寫高性能的Web應用程式，特別是那些計算密集型的任務。

WebGPU

WebGPU是一個新的JavaScript API，旨在提供對現代GPU硬件的低階訪問。它取代了WebGL，並提供了更現代、更強大的圖形編程接口。



Fetch → Demux → Decode → Process → Render

Fetch / WebSockets / WebTransport / RTCDataChannel

JavaScript / WebAssembly

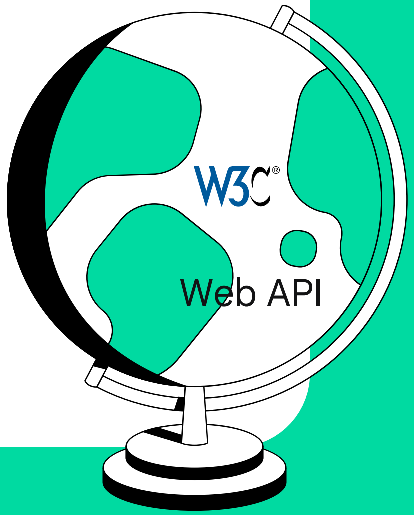
WebCodecs

WebGPU / WebAssembly

Canvas

WebCodecs介紹

為網頁的低延遲直播、雲端遊戲、
多媒體檔案編輯和轉碼等場景，
提出了一個絕佳的解決方案。



WebCodecs 的主要用途

直播（Live streaming）

用來即時編碼與解碼影片串流，例如即時直播影像或視訊會議。

雲端遊戲（Cloud gaming）

在伺服器處理畫面編碼，瀏覽器解碼並快速渲染。

媒體文件編輯與轉碼

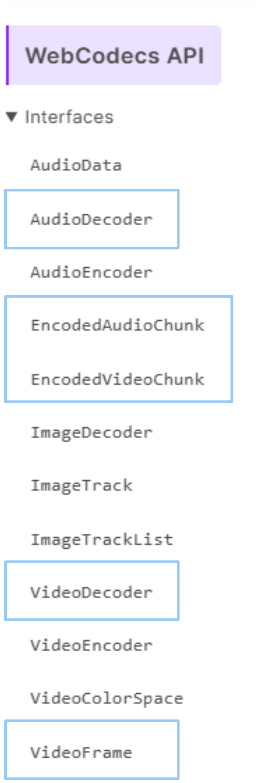
高效率地於瀏覽器中轉換音視頻格式，或編輯多媒體內容。

WebCodecs API 官方範例

這個範例展示了如何透過呼叫WebCodecs API來處理音視頻的解碼以及同步：

1. AudioDecoder
2. VideoDecoder
3. 從demuxer取出Chunks
4. 操作由Decoder解碼出來的Frame
5. 使用SharedArrayBuffer和Worker共享記憶體

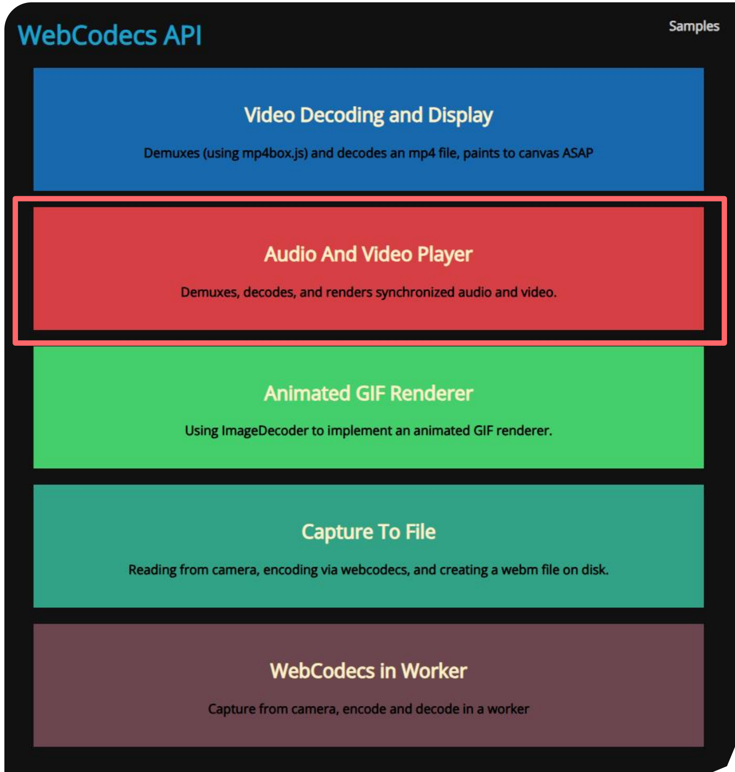
範例請掃我



WebCodecs API

▼ Interfaces

- AudioData
- AudioDecoder
- AudioEncoder
- EncodedAudioChunk
- EncodedVideoChunk
- ImageDecoder
- ImageTrack
- ImageTrackList
- VideoDecoder
- VideoEncoder
- VideoColorSpace
- VideoFrame



WebCodecs API

Samples

Video Decoding and Display

Demuxes (using mp4box.js) and decodes an mp4 file, paints to canvas ASAP

Audio And Video Player

Demuxes, decodes, and renders synchronized audio and video.

Animated GIF Renderer

Using ImageDecoder to implement an animated GIF renderer.

Capture To File

Reading from camera, encoding via webcodecs, and creating a webm file on disk.

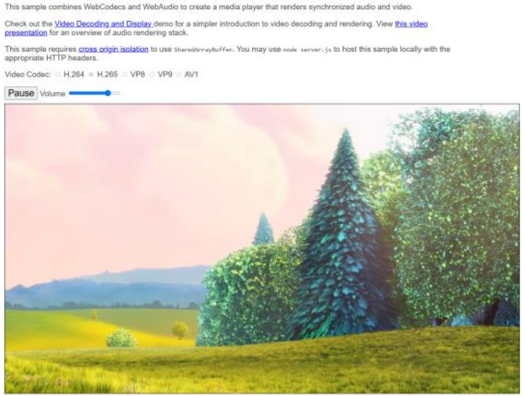
WebCodecs in Worker

Capture from camera, encode and decode in a worker

Audio And Video Player

範例展示如何利用 **Web Worker** 和 **WebCodecs API**，實現音頻與視頻的解碼、同步和播放的流程。

Video Codec: ● H.264 ○ H.265 ○ VP8 ○ VP9 ○ AV1

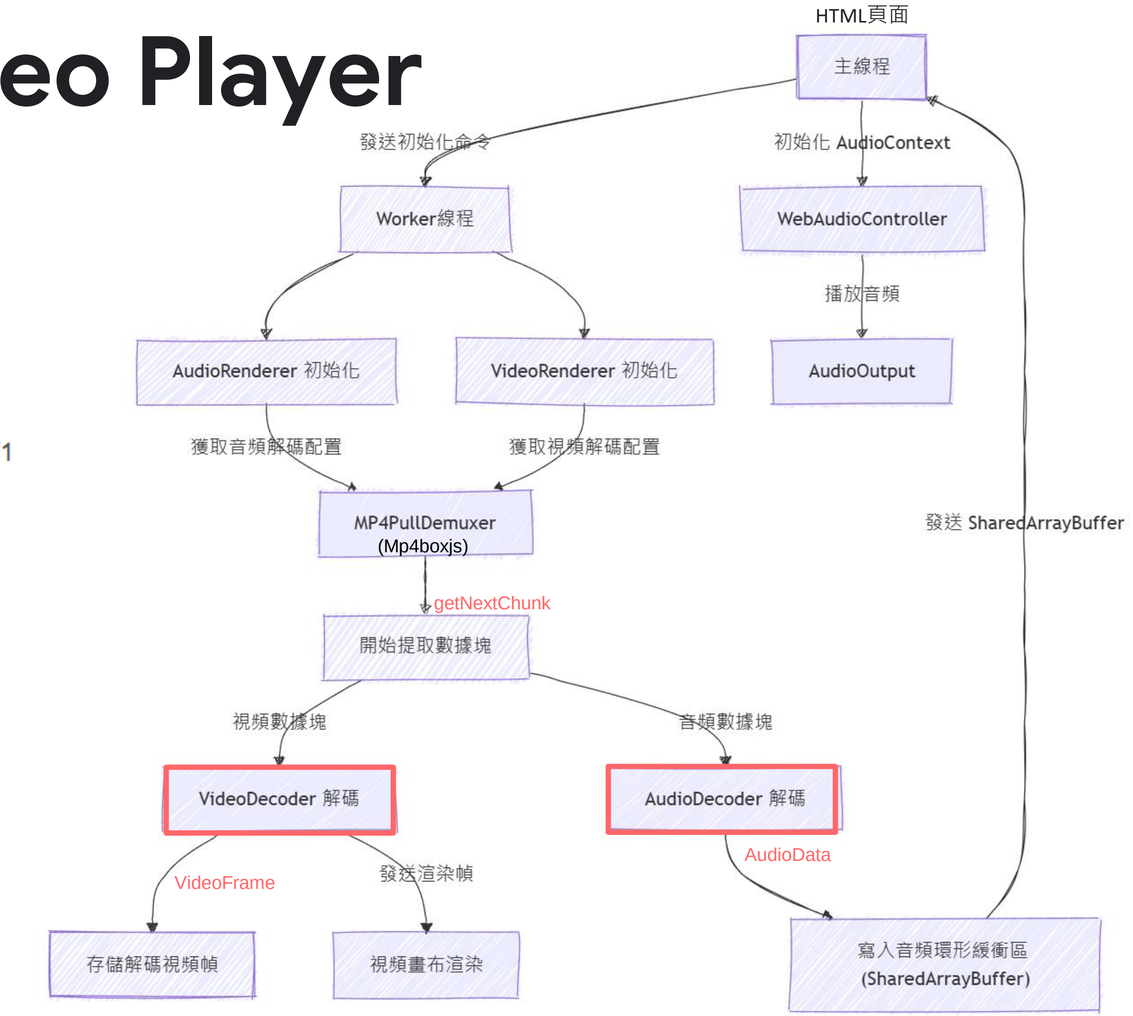


This sample combines WebCodecs and WebAudio to create a media player that renders synchronized audio and video.

Check out the Video Decoding and Display demo for a simpler introduction to video decoding and rendering. View this video presentation for an overview of audio rendering stack.

This sample requires cross origin isolation to use `SharedArrayBuffer`. You may use `node server.js` to host this sample locally with the appropriate HTTP headers.

Video Codec: ○ H.264 ● H.265 ○ VP8 ○ VP9 ○ AV1



HTML頁面

主線程

發送初始化命令

初始化 AudioContext

Worker線程

WebAudioController

播放音頻

AudioRenderer 初始化

VideoRenderer 初始化

AudioOutput

獲取音頻解碼配置

獲取視頻解碼配置

MP4PullDemuxer (Mp4boxjs)

發送 SharedArrayBuffer

getNextChunk

開始提取數據塊

視頻數據塊

音頻數據塊

VideoDecoder 解碼

AudioDecoder 解碼

VideoFrame

發送渲染幀

AudioData

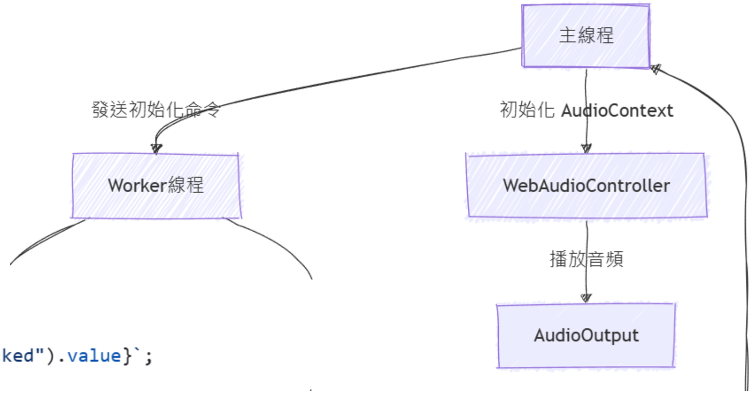
存儲解碼視頻幀

視頻畫布渲染

寫入音頻環形緩衝區 (SharedArrayBuffer)

Women Techmakers
Taipei

關鍵程式碼片段



主線程

發送初始化命令

初始化 AudioContext

Worker線程

WebAudioController

播放音頻

AudioOutput

audio_video_player.html

```
// Wait for worker initialization. Use metadata to init the WebAudioController.
await new Promise(resolve => {
  const videoCodec = `${document.querySelector("input[name=\"video_codec\"]:checked").value}`;
  mediaWorker.postMessage(
    {
      command: 'initialize',
      audioFile: '../data/bbb_audio_aac_frag.mp4',
      videoFile: `../data/bbb_video_${videoCodec}_frag.mp4`,
      canvas: offscreenCanvas
    },
    { transfer: [offscreenCanvas] }
  );

  mediaWorker.addEventListener('message', (e) => {
    console.assert(e.data.command == 'initialize-done');
    audioController.initialize(e.data.sampleRate, e.data.channelCount, e.data.sharedArrayBuffer);
    initDone = true;
    resolve();
  });
});
```

初始化線程中的音視頻來源，並設定最終視頻的繪製目標

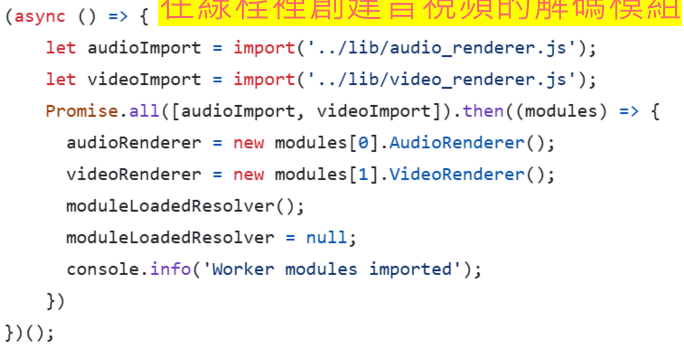
從sharedArrayBuffer(用來傳遞共享記憶體)取出音頻丟進WebAudioController播放聲音

https://github.com/w3c/webcodecs/tree/main/samples/audio-video-player

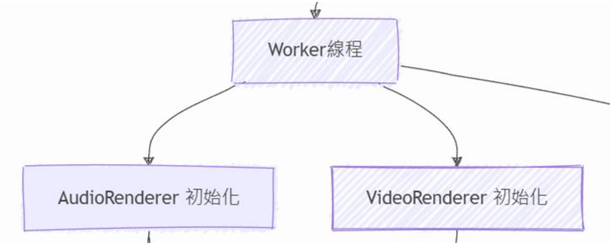
關鍵程式碼片段

media_worker.js

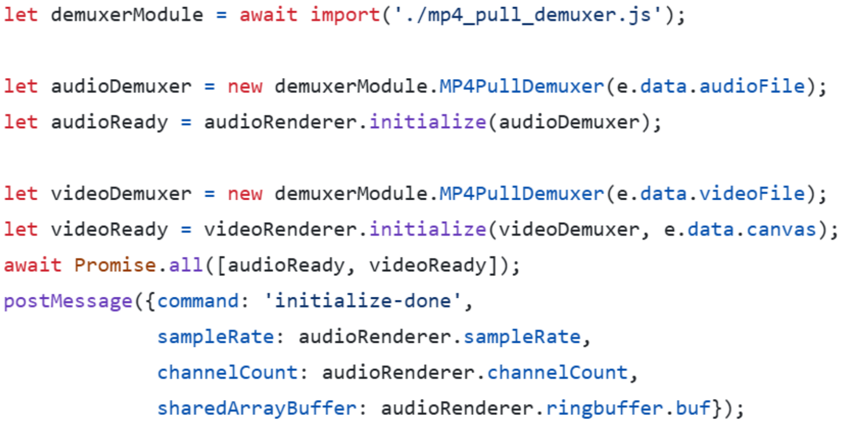
在線程裡創建音視頻的解碼模組



```
(async () => {
    let audioImport = import('../lib/audio_renderer.js');
    let videoImport = import('../lib/video_renderer.js');
    Promise.all([audioImport, videoImport]).then((modules) => {
      audioRenderer = new modules[0].AudioRenderer();
      videoRenderer = new modules[1].VideoRenderer();
      moduleLoadedResolver();
      moduleLoadedResolver = null;
      console.info('Worker modules imported');
    })
})();
```

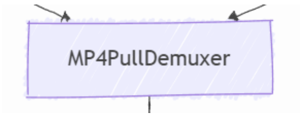


解封裝影片並且設定要呈現的Canvas目標



```
let demuxerModule = await import('./mp4_pull_demuxer.js');

let audioDemuxer = new demuxerModule.MP4PullDemuxer(e.data.audioFile);
let audioReady = audioRenderer.initialize(audioDemuxer);

let videoDemuxer = new demuxerModule.MP4PullDemuxer(e.data.videoFile);
let videoReady = videoRenderer.initialize(videoDemuxer, e.data.canvas);
await Promise.all([audioReady, videoReady]);
postMessage({command: 'initialize-done',
             sampleRate: audioRenderer.sampleRate,
             channelCount: audioRenderer.channelCount,
             sharedArrayBuffer: audioRenderer.ringbuffer.buf});
```

解封裝音頻並且將儲存的sharedArrayBuffer傳給主線程



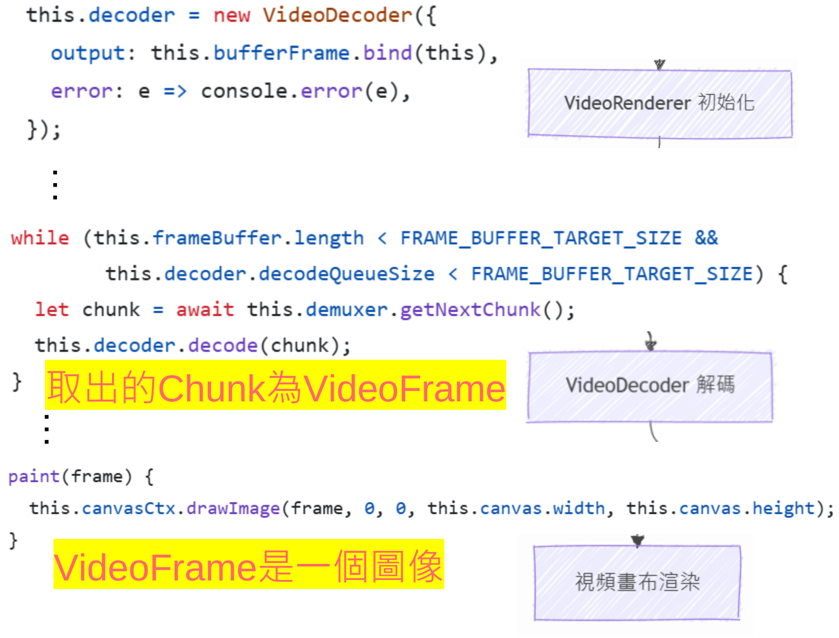
Women Techmakers Taipei

https://github.com/w3c/webcodecs/tree/main/samples/audio-video-player

關鍵程式碼片段

lib/video_renderer.js

VideoDecoder是WebCodecs的Web API



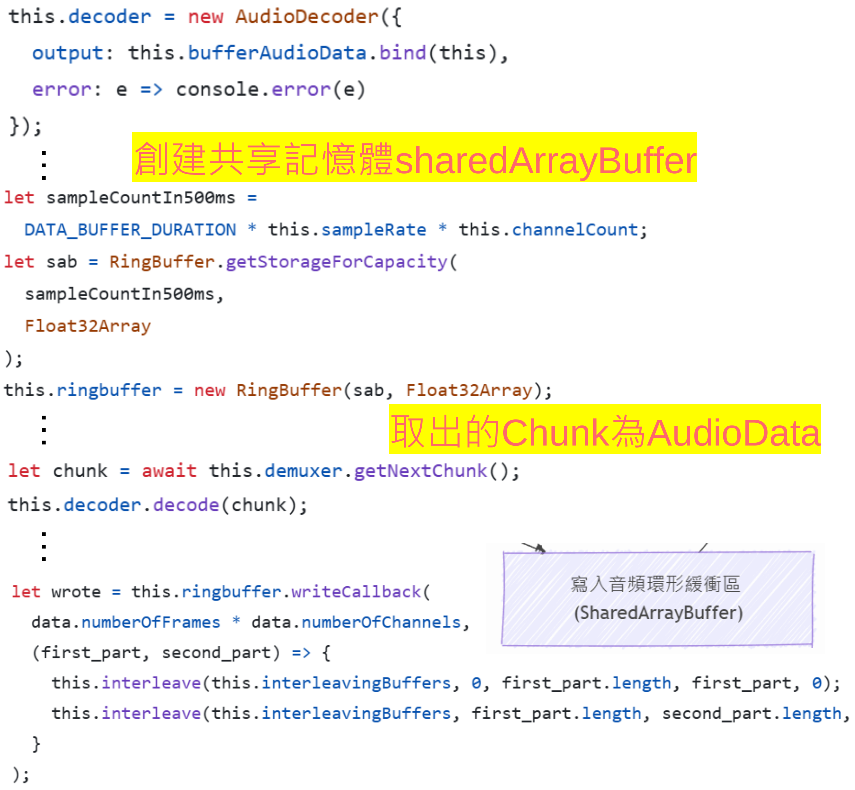
```
this.decoder = new VideoDecoder({
  output: this.bufferFrame.bind(this),
  error: e => console.error(e),
});
 ⋮

while (this.frameBuffer.length < FRAME_BUFFER_TARGET_SIZE &&
        this.decoder.decodeQueueSize < FRAME_BUFFER_TARGET_SIZE) {
  let chunk = await this.demuxer.getNextChunk();
  this.decoder.decode(chunk);
}
 ⋮

paint(frame) {
  this.canvasCtx.drawImage(frame, 0, 0, this.canvas.width, this.canvas.height);
}
```

取出的Chunk為VideoFrame

VideoFrame是一個圖像

lib/audio_renderer.js

AudioDecoder是WebCodecs的Web API



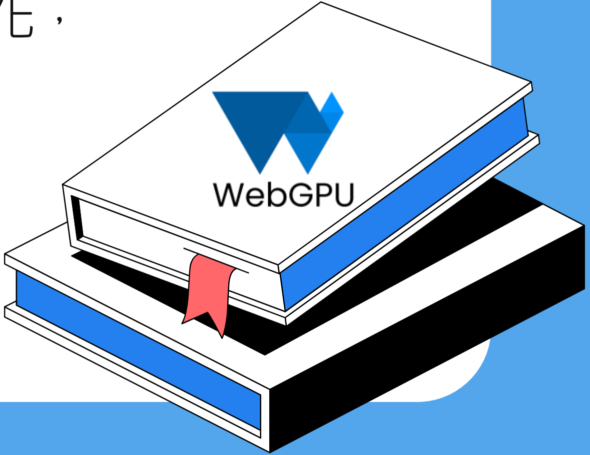
```
this.decoder = new AudioDecoder({
  output: this.bufferAudioData.bind(this),
  error: e => console.error(e)
});
  ⋮
let sampleCountIn500ms =
  DATA_BUFFER_DURATION * this.sampleRate * this.channelCount;
let sab = RingBuffer.getStorageForCapacity(
  sampleCountIn500ms,
  Float32Array
);
this.ringbuffer = new RingBuffer(sab, Float32Array);
  ⋮
let chunk = await this.demuxer.getNextChunk();
this.decoder.decode(chunk);
  ⋮
let wrote = this.ringbuffer.writeCallback(
  data.numberOfFrames * data.numberOfChannels,
  (first_part, second_part) => {
    this.interleave(this.interleavingBuffers, 0, first_part.length, first_part, 0);
    this.interleave(this.interleavingBuffers, first_part.length, second_part.length,
  }
);
```

創建共享記憶體sharedArrayBuffer

取出的Chunk為AudioData

WebGPU介紹

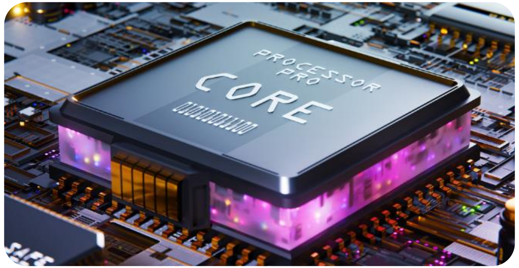
WebGPU讓瀏覽器端的AI運算成為可能。
其特別針對高耗能的計算工作負載進行優化，
例如機器學習和 AI 推理



WebGPU：WebGL 的現代繼任者

WebAssembly 的進步

SIMD是一種並行處理技術，可一次性對多個數據執行相同的指令。Relaxed SIMD則可以在不需要高精度的情境下降低精度以加速處理。支援硬體級別的FP16半經度浮點數。



支持大規模並行運算

針對大規模數據處理進行了優化。充分利用 GPU 的平行處理能力，可同時運算數千甚至數百萬個數據點。對於需要高效處理的工作負載（如深度學習推理、圖像渲染）尤為重要，能提升計算性能、減少運行時間。

Packed Integer Dot Products

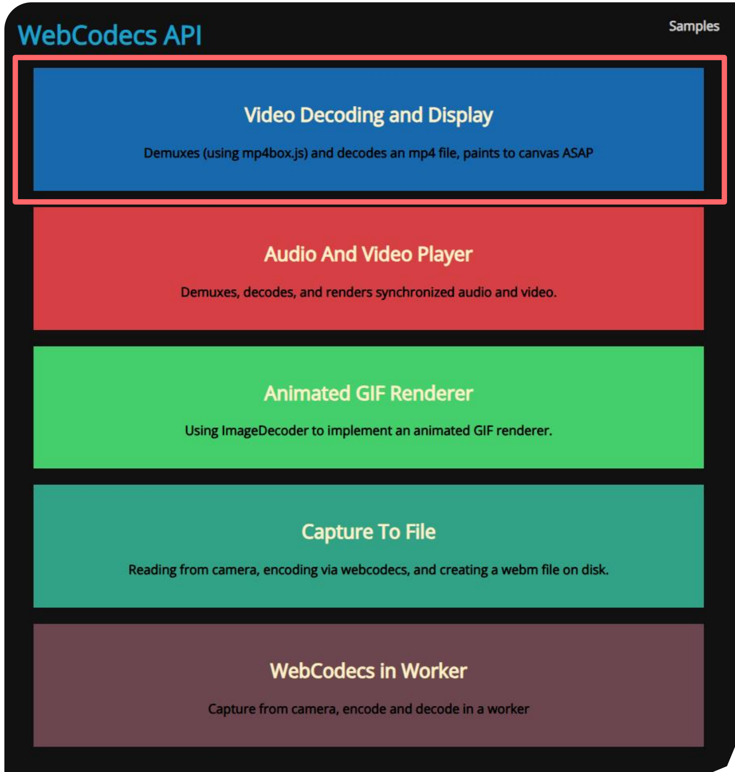
壓縮整數點積可將高精度的 32 位元浮點數（FP32）或 16 位元浮點數（FP16）轉換為更低精度的 8 位元整數表示，提升運算效率、同時顯著減少記憶體和計算資源的使用，提升性能。

結合WebCodecs API及WebGPU

這個範例展示了如何將WebCodecs API與WebGPU結合，實現高效率的影片處理：

1. 數據從解碼到渲染僅需一次傳遞
2. WebGPU 通過直接操作 GPU 記憶體，在渲染過程中執行高效的並行計算
3. WebCodecs 和 WebGPU 都支持非同步操作，解碼與渲染可以在獨立的執行緒中進行，充分利用多核資源。

範例請掃我


WebCodecs API

Samples

Video Decoding and Display

Demuxes (using mp4box.js) and decodes an mp4 file, paints to canvas ASAP

Audio And Video Player

Demuxes, decodes, and renders synchronized audio and video.

Animated GIF Renderer

Using ImageDecoder to implement an animated GIF renderer.

Capture To File

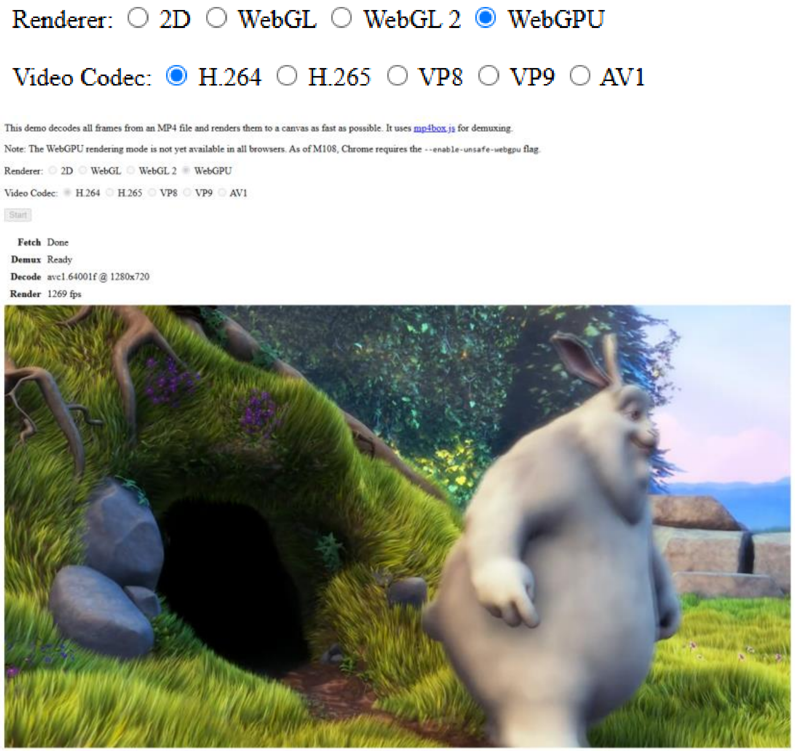
Reading from camera, encoding via webcodecs, and creating a webm file on disk.

WebCodecs in Worker

Capture from camera, encode and decode in a worker

Video Decoding and Display

此範例中，WebGPU透過importExternalTexture將VideoFrame 作為外部紋理匯入。
並透過WGSL將外部紋理直接渲染至Canvas。



Renderer: ○ 2D ○ WebGL ○ WebGL 2 ● WebGPU

Video Codec: ● H.264 ○ H.265 ○ VP8 ○ VP9 ○ AV1

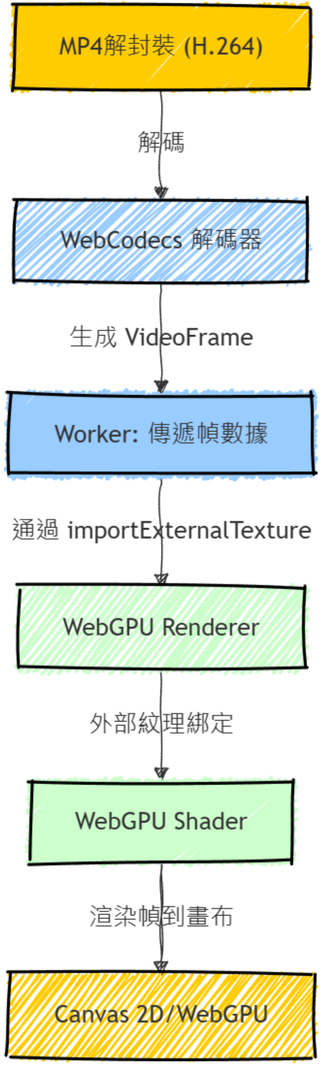
This demo decodes all frames from an MP4 file and renders them to a canvas as fast as possible. It uses mp4box.js for demuxing.

Note: The WebGPU rendering mode is not yet available in all browsers. As of M108, Chrome requires the `--enable-unsafe-webgpu` flag.

Renderer: ○ 2D ○ WebGL ○ WebGL 2 ● WebGPU

Video Codec: ● H.264 ○ H.265 ○ VP8 ○ VP9 ○ AV1

Fetch Done
Demux Ready
Decode avc1.64001f @ 1280x720
Render 1269 fps



MP4解封裝 (H.264)

解碼

WebCodecs 解碼器

生成 VideoFrame

Worker: 傳遞幀數據

通過 importExternalTexture

WebGPU Renderer

外部紋理綁定

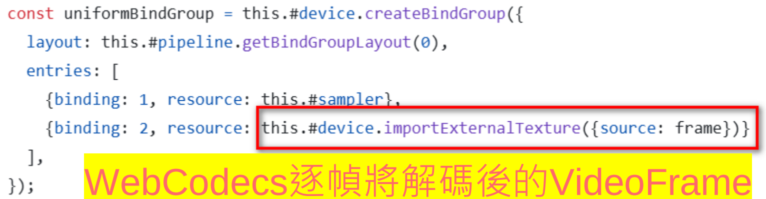
WebGPU Shader

渲染幀到畫布

Canvas 2D/WebGPU

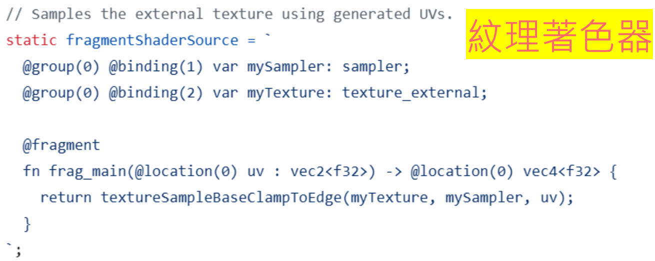
關鍵程式碼片段

renderer_webgpu.js



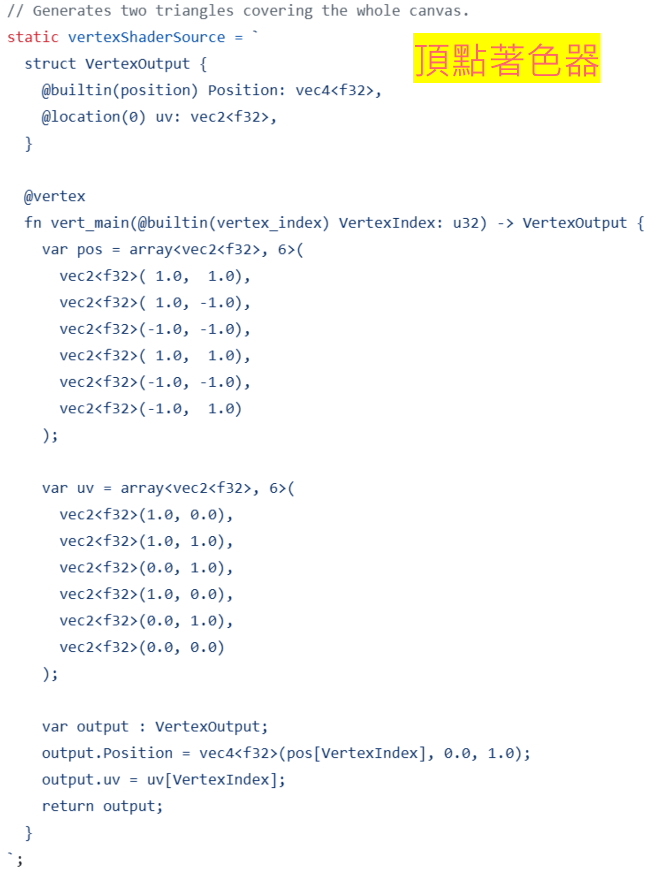
```
const uniformBindGroup = this.#device.createBindGroup({
  layout: this.#pipeline.getBindGroupLayout(0),
  entries: [
    {binding: 1, resource: this.#sampler},
    {binding: 2, resource: this.#device.importExternalTexture({source: frame})}
  ],
});
```

WebCodecs逐幀將解碼後的VideoFrame傳送至WebGPU



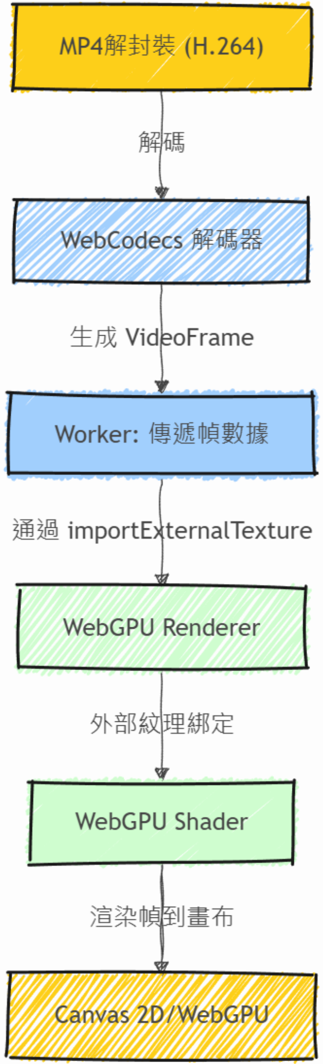
紋理著色器

```
// Samples the external texture using generated UVs.
static fragmentShaderSource = `
  @group(0) @binding(1) var mySampler: sampler;
  @group(0) @binding(2) var myTexture: texture_external;

  @fragment
  fn frag_main(@location(0) uv : vec2<f32>) -> @location(0) vec4<f32> {
    return textureSampleBaseClampToEdge(myTexture, mySampler, uv);
  }
`;
```



頂點著色器

```
// Generates two triangles covering the whole canvas.
static vertexShaderSource = `
  struct VertexOutput {
    @builtin(position) Position: vec4<f32>,
    @location(0) uv: vec2<f32>,
  }

  @vertex
  fn vert_main(@builtin(vertex_index) VertexIndex: u32) -> VertexOutput {
    var pos = array<vec2<f32>, 6>(
      vec2<f32>( 1.0,  1.0),
      vec2<f32>( 1.0, -1.0),
      vec2<f32>(-1.0, -1.0),
      vec2<f32>( 1.0,  1.0),
      vec2<f32>(-1.0, -1.0),
      vec2<f32>(-1.0,  1.0)
    );

    var uv = array<vec2<f32>, 6>(
      vec2<f32>(1.0, 0.0),
      vec2<f32>(1.0, 1.0),
      vec2<f32>(0.0, 1.0),
      vec2<f32>(1.0, 0.0),
      vec2<f32>(0.0, 1.0),
      vec2<f32>(0.0, 0.0)
    );

    var output : VertexOutput;
    output.Position = vec4<f32>(pos[VertexIndex], 0.0, 1.0);
    output.uv = uv[VertexIndex];
    return output;
  }
`;
```

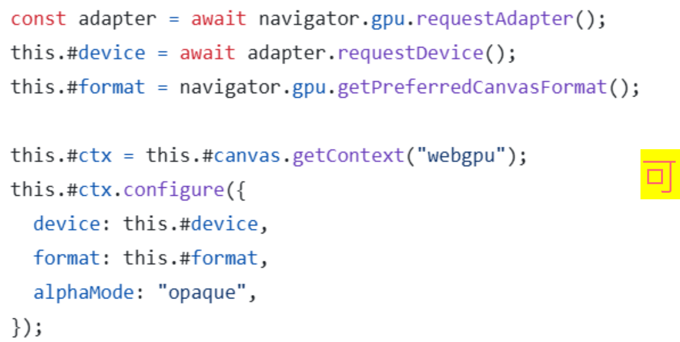


Women Techmakers Taipei

https://github.com/w3c/webcodecs/tree/main/samples/video-decode-display

關鍵程式碼片段

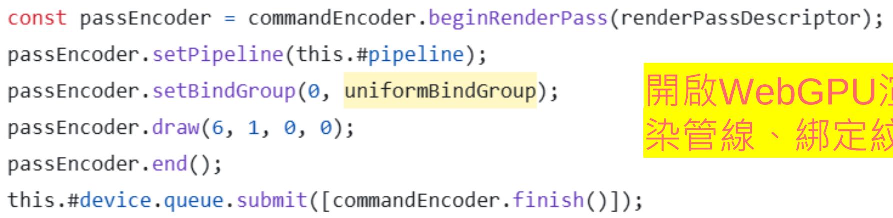
renderer_webgpu.js



```
const adapter = await navigator.gpu.requestAdapter();
this.#device = await adapter.requestDevice();
this.#format = navigator.gpu.getPreferredCanvasFormat();

this.#ctx = this.#canvas.getContext("webgpu");
this.#ctx.configure({
  device: this.#device,
  format: this.#format,
  alphaMode: "opaque",
});
```

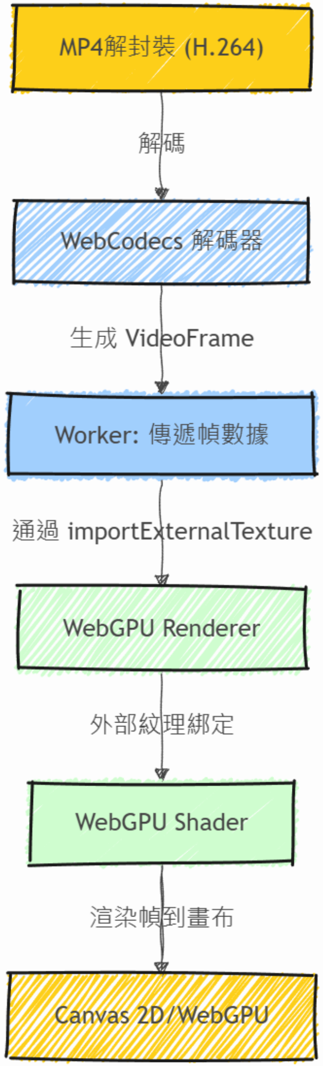
可直接在WebGPU渲染圖像至Canvas

⋮



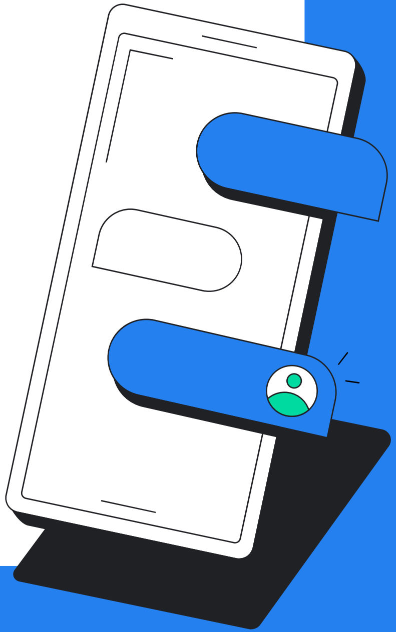
```
const passEncoder = commandEncoder.beginRenderPass(renderPassDescriptor);
passEncoder.setPipeline(this.#pipeline);
passEncoder.setBindGroup(0, uniformBindGroup);
passEncoder.draw(6, 1, 0, 0);
passEncoder.end();
this.#device.queue.submit([commandEncoder.finish()]);
```

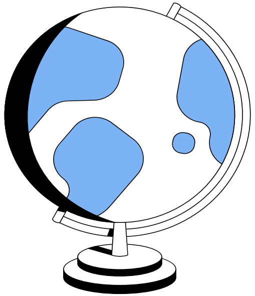
開啟WebGPU渲染通道、設置渲染管線、綁定紋理和頂點並繪圖



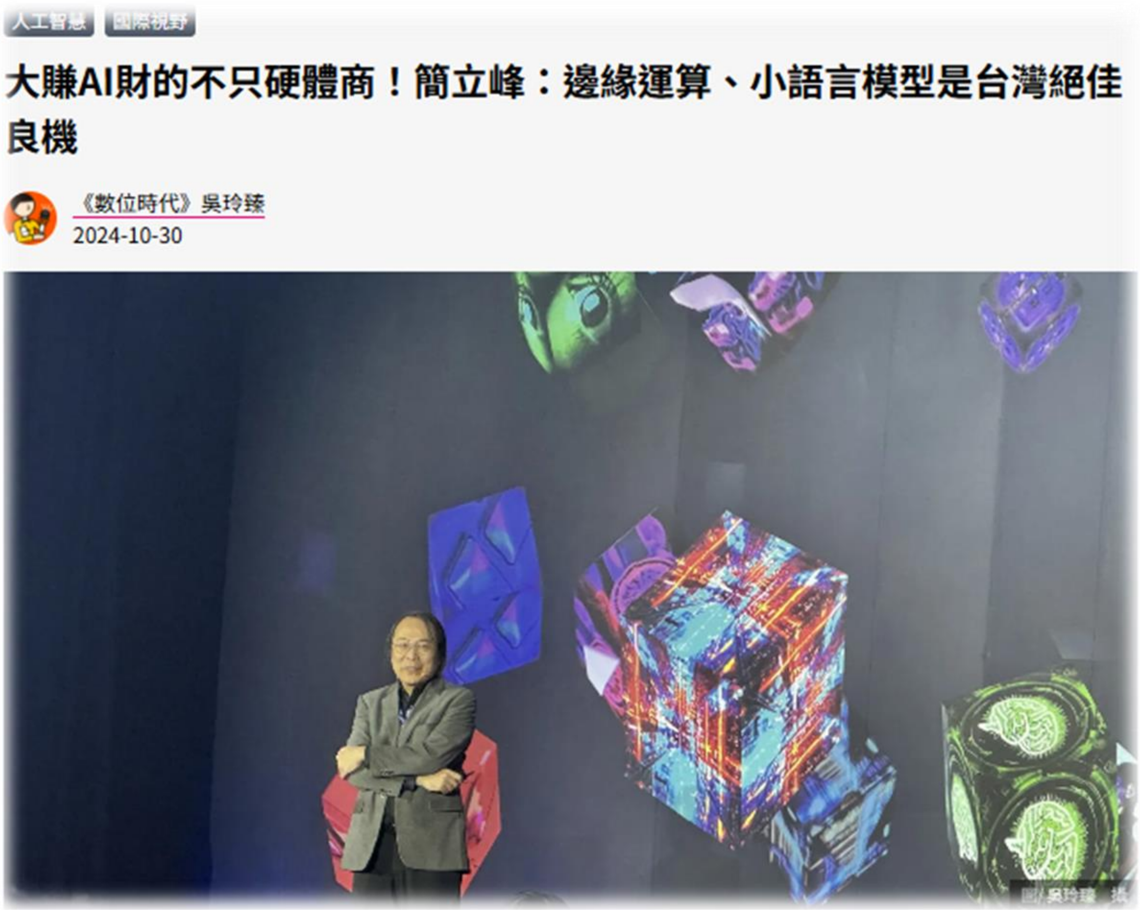
https://github.com/w3c/webcodecs/tree/main/samples/video-decode-display

應用場景與
未來展望





- 邊緣運算是AI未來發展的必然趨勢。
- 網頁具有跨平台、免安裝、易於使用、即時更新、不受硬體限制等優點，可有效降低使用者的學習門檻與開發者的部署成本。
- 隨著短影音、直播、線上會議、虛擬實境（VR）、擴增實境（AR）等技術的快速進步，**結合邊緣運算與網頁技術的應用，將能更有效地滿足即時性、高效能與高互動性的需求**。



人工智慧 國際視野

大賺AI財的不只硬體商！簡立峰：邊緣運算、小語言模型是台灣絕佳良機

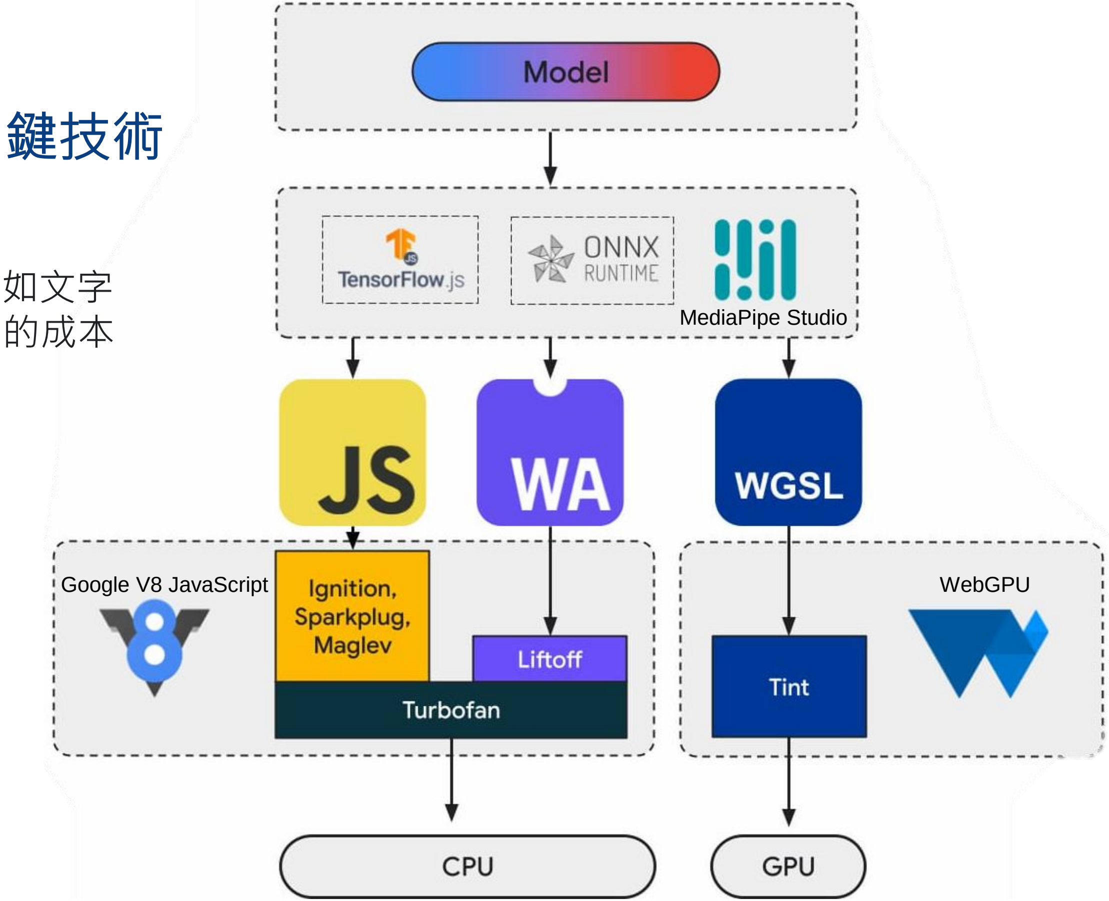
《數位時代》吳玲臻

2024-10-30

圖/吳玲臻 攝

瀏覽器運行AI的關鍵技術

對於較小的工作負載 (例如文字或音訊工作負載)，GPU 的成本會很高



應用場景與未來展望

擴增實境（AR）與虛擬實境（VR）

WebCodecs 解碼全景視頻（360 度視頻），WebGPU 渲染到 AR/VR 設備的視圖中。範例： 虛擬旅遊、沉浸式培訓。

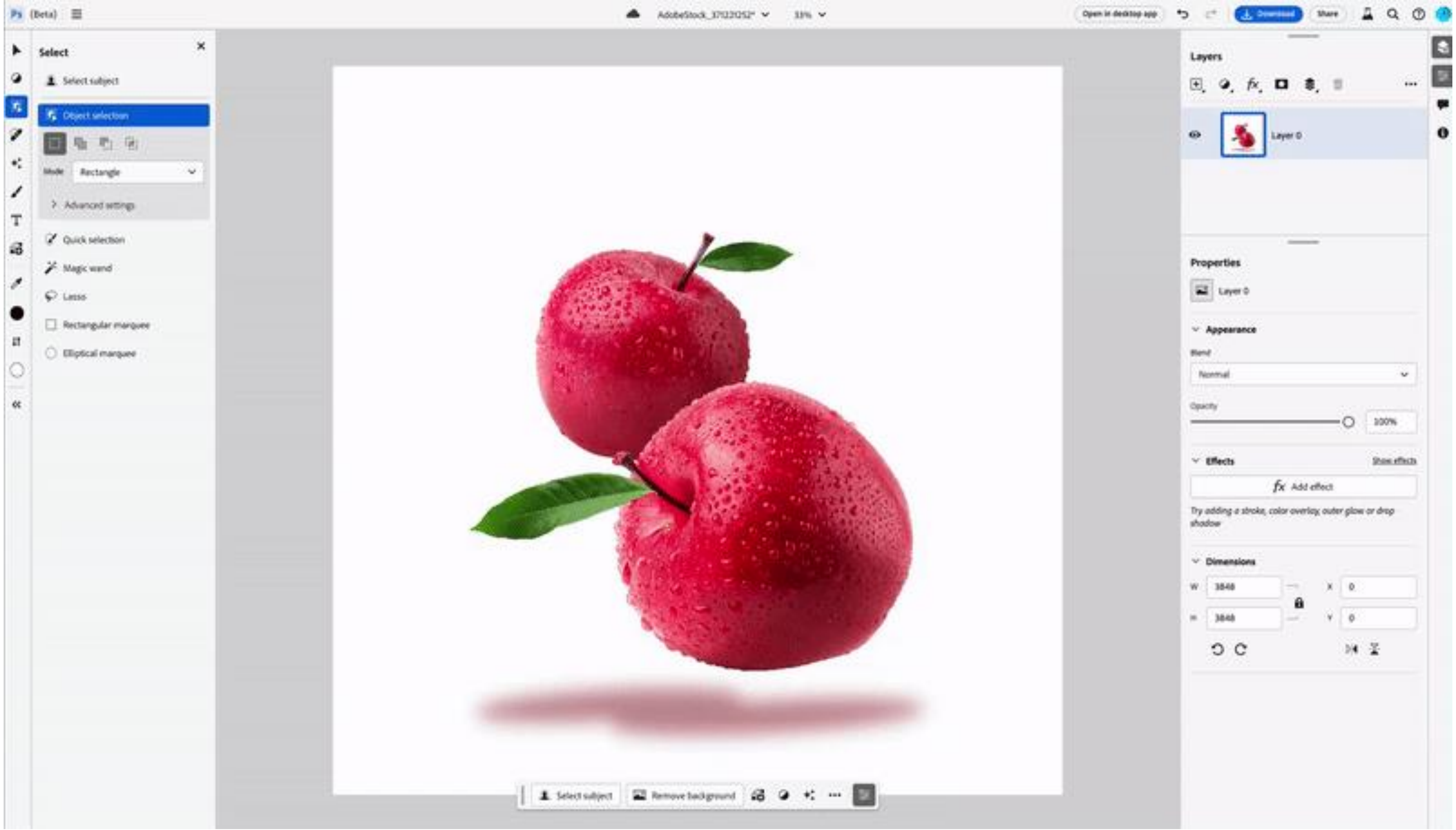
機器學習和影像處理

WebCodecs 解碼視頻，WebGPU 支援使用機器學習模型（如降噪、畫質增強）進行處理，然後高效渲染結果。範例： 即時畫質增強、即時的人臉識別。

遊戲和互動媒體

WebCodecs 負責解碼串流視頻內容，WebGPU 渲染遊戲場景或視頻中的交互內容。 範例：雲遊戲平台（如 Stadia）、交互式劇情視頻。

Adobe 使用 Tensorflow.js 強化 Photoshop 網頁版



Mode: Rectangle

Blend: Normal

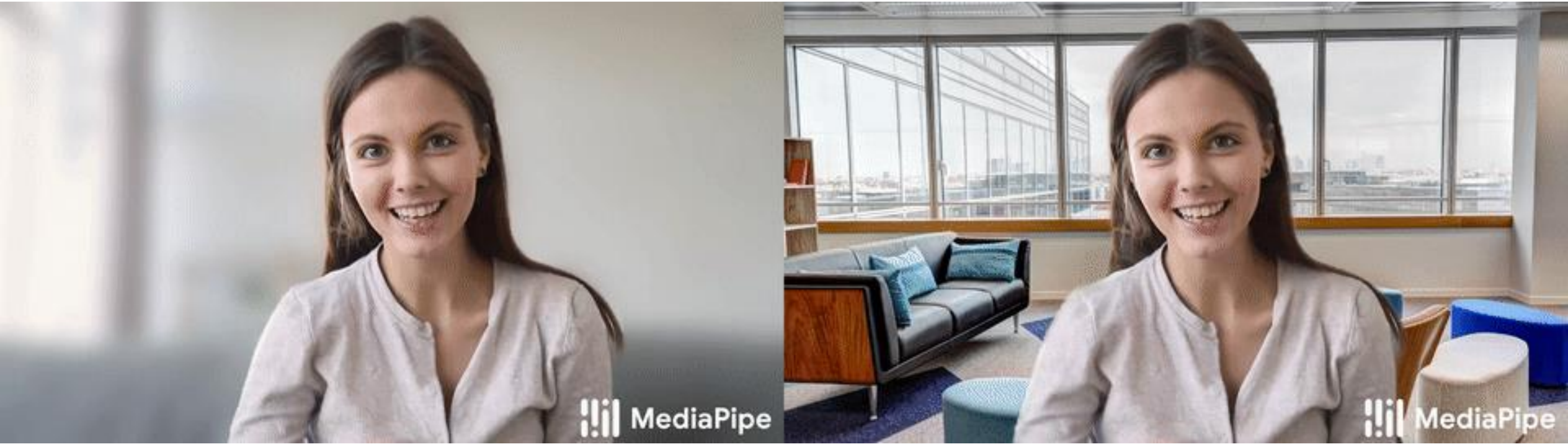
Opacity: 100%

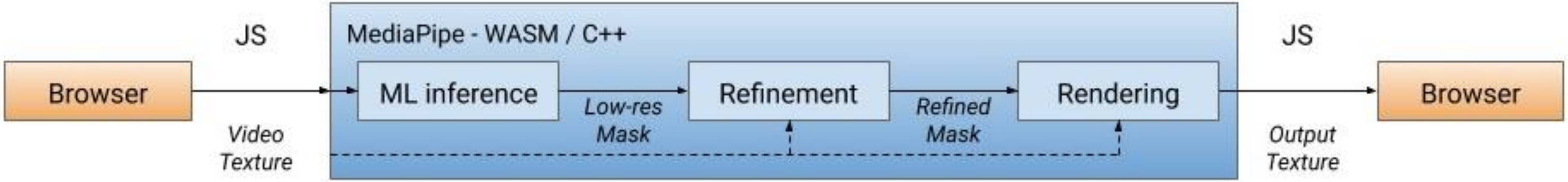
Try adding a stroke, color overlay, outer glow or drop shadow

W: 3848 X: 0

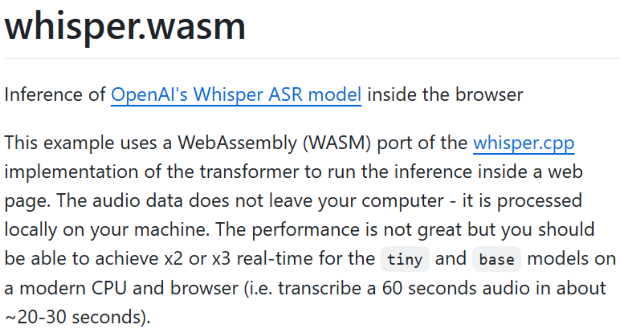
H: 3848 Y: 0

Google Meet 新增背景模糊效果





在網頁中使用OpenAI Whisper功能

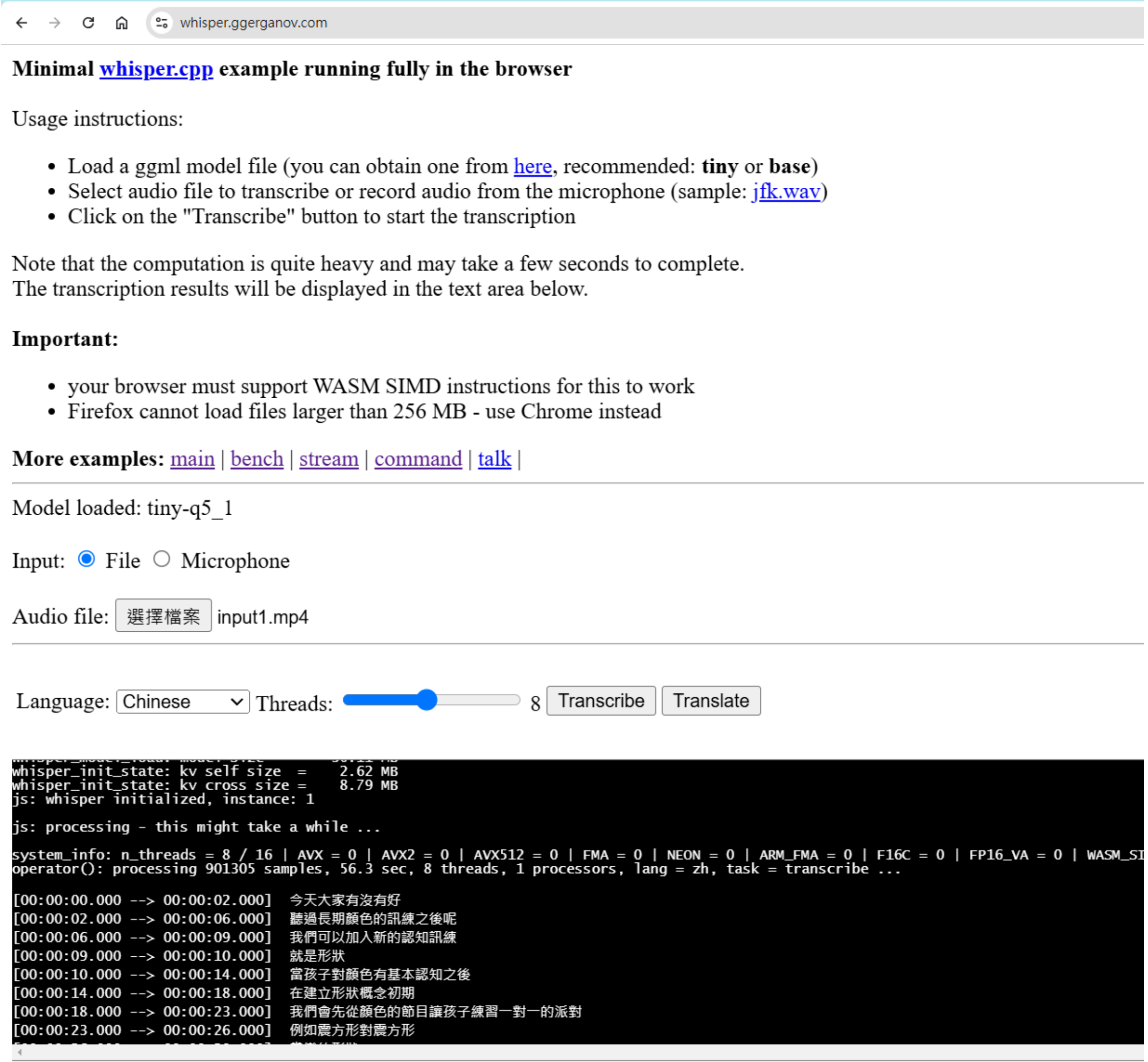


whisper.wasm

Inference of OpenAI's Whisper ASR model inside the browser

This example uses a WebAssembly (WASM) port of the whisper.cpp implementation of the transformer to run the inference inside a web page. The audio data does not leave your computer - it is processed locally on your machine. The performance is not great but you should be able to achieve x2 or x3 real-time for the `tiny` and `base` models on a modern CPU and browser (i.e. transcribe a 60 seconds audio in about ~20-30 seconds).

掃我試玩！



Minimal whisper.cpp example running fully in the browser

Usage instructions:

- Load a ggml model file (you can obtain one from here, recommended: **tiny** or **base**)
- Select audio file to transcribe or record audio from the microphone (sample: jfk.wav)
- Click on the "Transcribe" button to start the transcription

Note that the computation is quite heavy and may take a few seconds to complete.
The transcription results will be displayed in the text area below.

Important:

- your browser must support WASM SIMD instructions for this to work
- Firefox cannot load files larger than 256 MB - use Chrome instead

More examples: main | bench | stream | command | talk |

Model loaded: tiny-q5_1

Input: ● File ○ Microphone

Audio file: 選擇檔案 input1.mp4

Language: Chinese Threads: 8

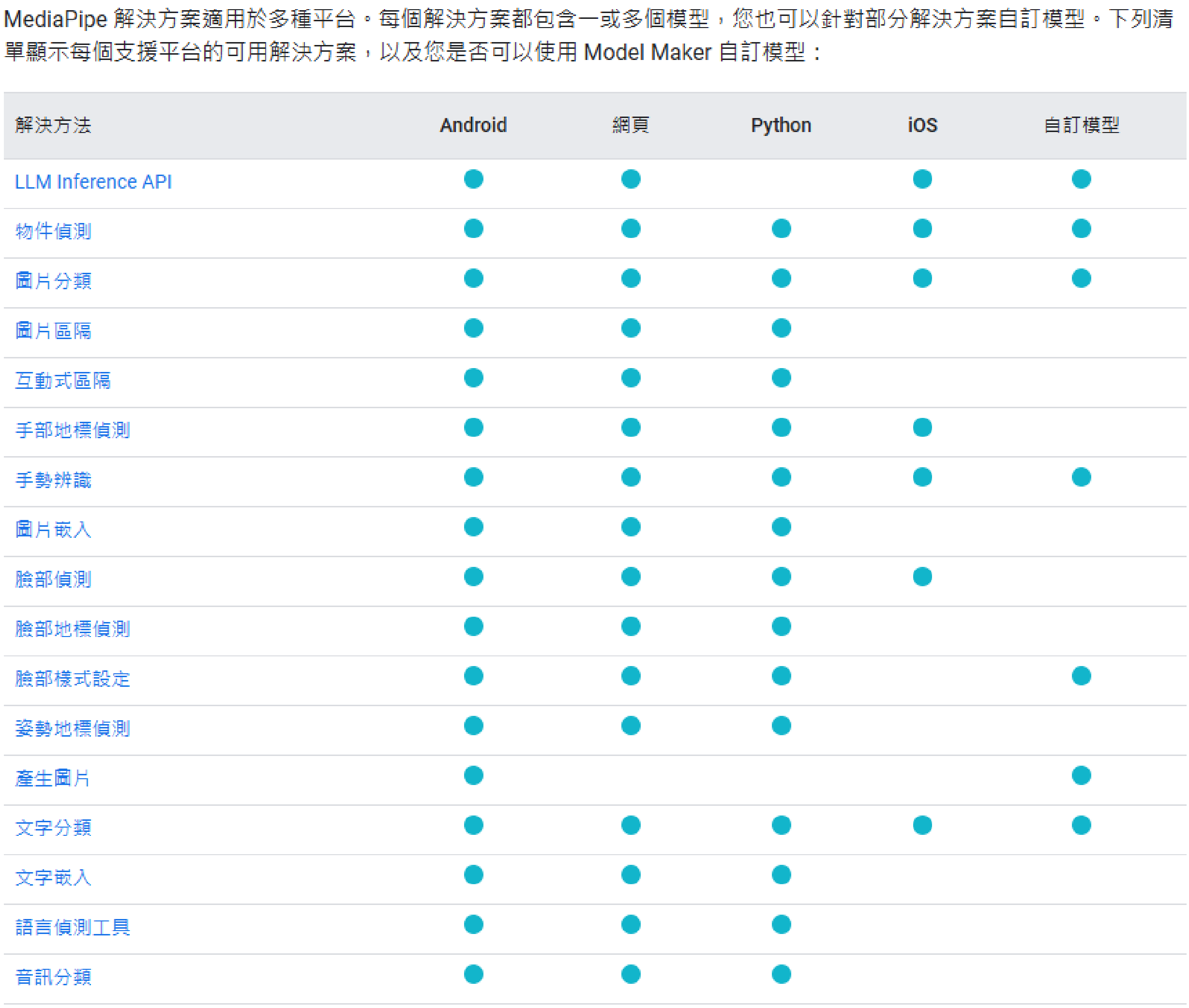
```
whisper_init_state: kv self size  =    2.62 MB
whisper_init_state: kv cross size =    8.79 MB
js: whisper initialized, instance: 1

js: processing - this might take a while ...

system_info: n_threads = 8 / 16 | AVX = 0 | AVX2 = 0 | AVX512 = 0 | FMA = 0 | NEON = 0 | ARM_FMA = 0 | F16C = 0 | FP16_VA = 0 | WASM_SI
operator(): processing 901305 samples, 56.3 sec, 8 threads, 1 processors, lang = zh, task = transcribe ...

[00:00:00.000 --> 00:00:02.000]  今天大家有沒有好
[00:00:02.000 --> 00:00:06.000]  聽過長期顏色的訊練之後呢
[00:00:06.000 --> 00:00:09.000]  我們可以加入新的認知訊練
[00:00:09.000 --> 00:00:10.000]  就是形狀
[00:00:10.000 --> 00:00:14.000]  當孩子對顏色有基本認知之後
[00:00:14.000 --> 00:00:18.000]  在建立形狀概念初期
[00:00:18.000 --> 00:00:23.000]  我們會先從顏色的節目讓孩子練習一對一的派對
[00:00:23.000 --> 00:00:26.000]  例如震方形對震方形
```

Women Techmakers
Taipei

MediaPipe 中有眾多範例

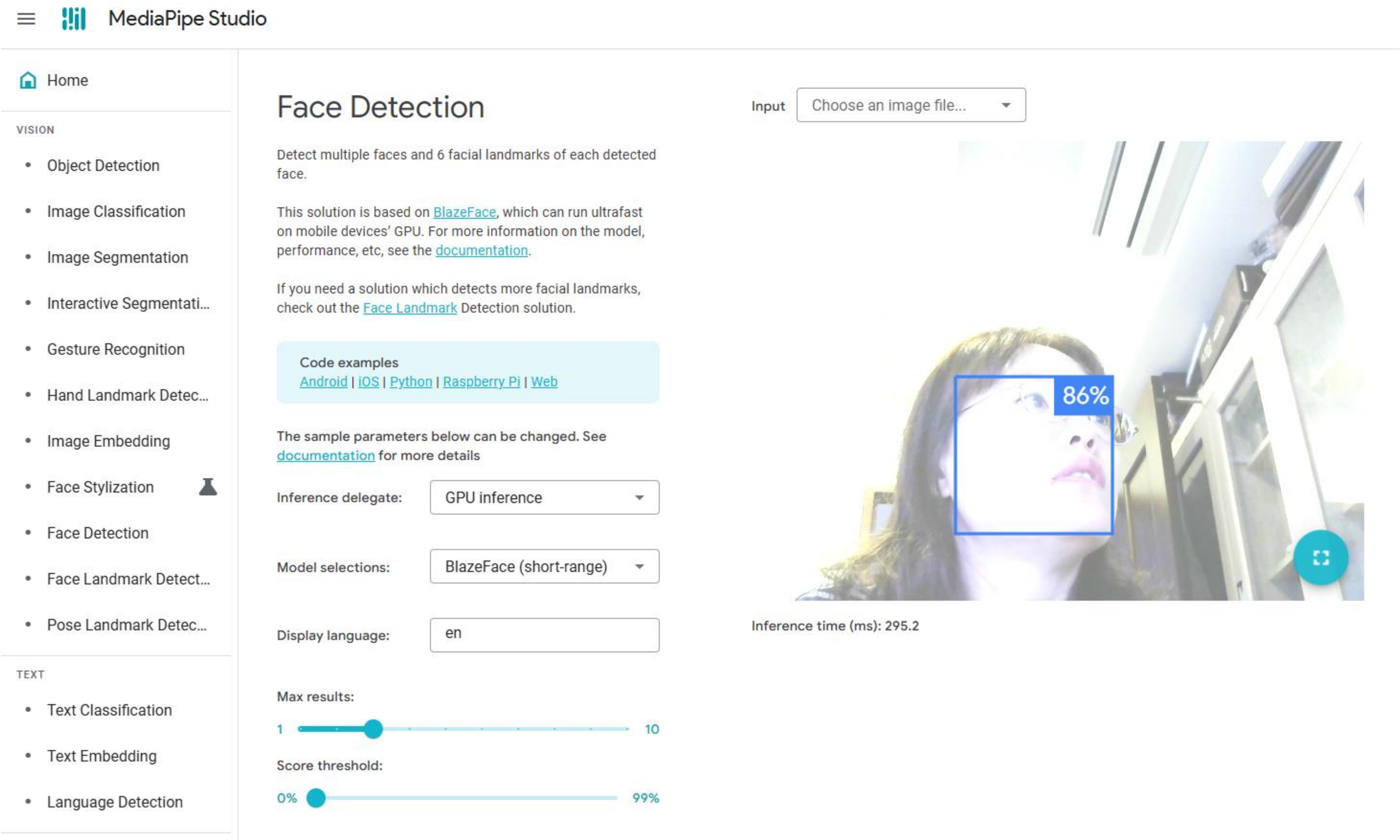
MediaPipe Solutions 提供一系列的程式庫和工具，可讓您在應用程式中快速套用人工智慧 (AI) 和機器學習 (ML) 技術。MediaPipe Solutions 屬於 MediaPipe 開放原始碼專案的一部分，因此您可以根據應用程式需求進一步自訂解決方案程式碼。

掃我試玩！



MediaPipe 解決方案適用於多種平台。每個解決方案都包含一或多個模型，您也可以針對部分解決方案自訂模型。下列清單顯示每個支援平台的可用解決方案，以及您是否可以使用 Model Maker 自訂模型：

解決方法	Android	網頁	Python	iOS	自訂模型
LLM Inference API	●	●		●	●
物件偵測	●	●	●	●	●
圖片分類	●	●	●	●	●
圖片區隔	●	●	●		
互動式區隔	●	●	●		
手部地標偵測	●	●	●	●	
手勢辨識	●	●	●	●	●
圖片嵌入	●	●	●		
臉部偵測	●	●	●	●	
臉部地標偵測	●	●	●		
臉部樣式設定	●	●	●		●
姿勢地標偵測	●	●	●		
產生圖片	●				●
文字分類	●	●	●	●	●
文字嵌入	●	●	●		
語言偵測工具	●	●	●		
音訊分類	●	●	●		



Face Detection

Detect multiple faces and 6 facial landmarks of each detected face.

This solution is based on BlazeFace, which can run ultrafast on mobile devices' GPU. For more information on the model, performance, etc, see the documentation.

If you need a solution which detects more facial landmarks, check out the Face Landmark Detection solution.

Code examples
Android | iOS | Python | Raspberry Pi | Web

The sample parameters below can be changed. See documentation for more details

Inference delegate: GPU inference

Model selections: BlazeFace (short-range)

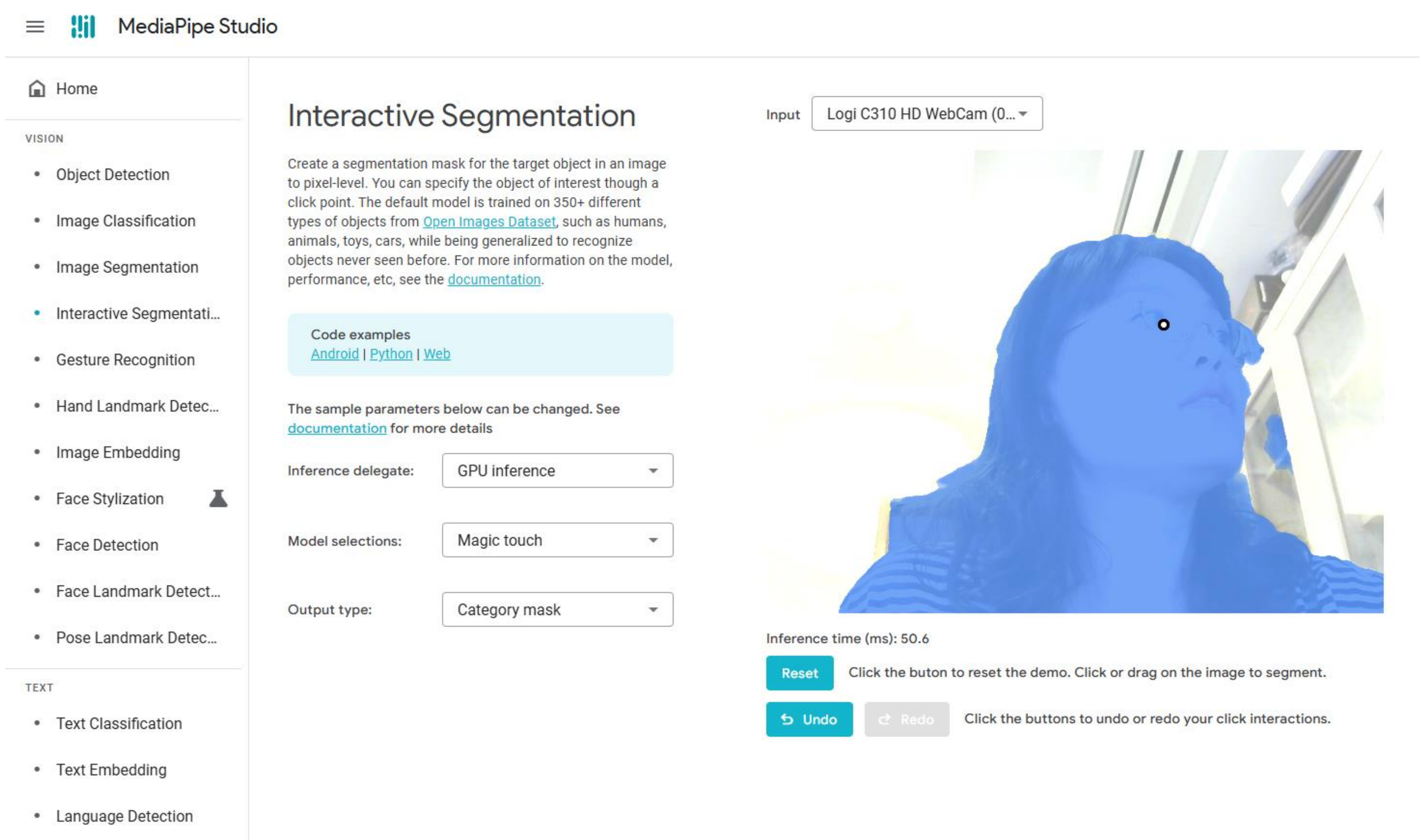
Display language: en

Max results: 1 10

Score threshold: 0% 99%

Input: Choose an image file...

Inference time (ms): 295.2



Interactive Segmentation

Create a segmentation mask for the target object in an image to pixel-level. You can specify the object of interest though a click point. The default model is trained on 350+ different types of objects from Open Images Dataset, such as humans, animals, toys, cars, while being generalized to recognize objects never seen before. For more information on the model, performance, etc, see the documentation.

Code examples
Android | Python | Web

The sample parameters below can be changed. See documentation for more details

Inference delegate: GPU inference

Model selections: Magic touch

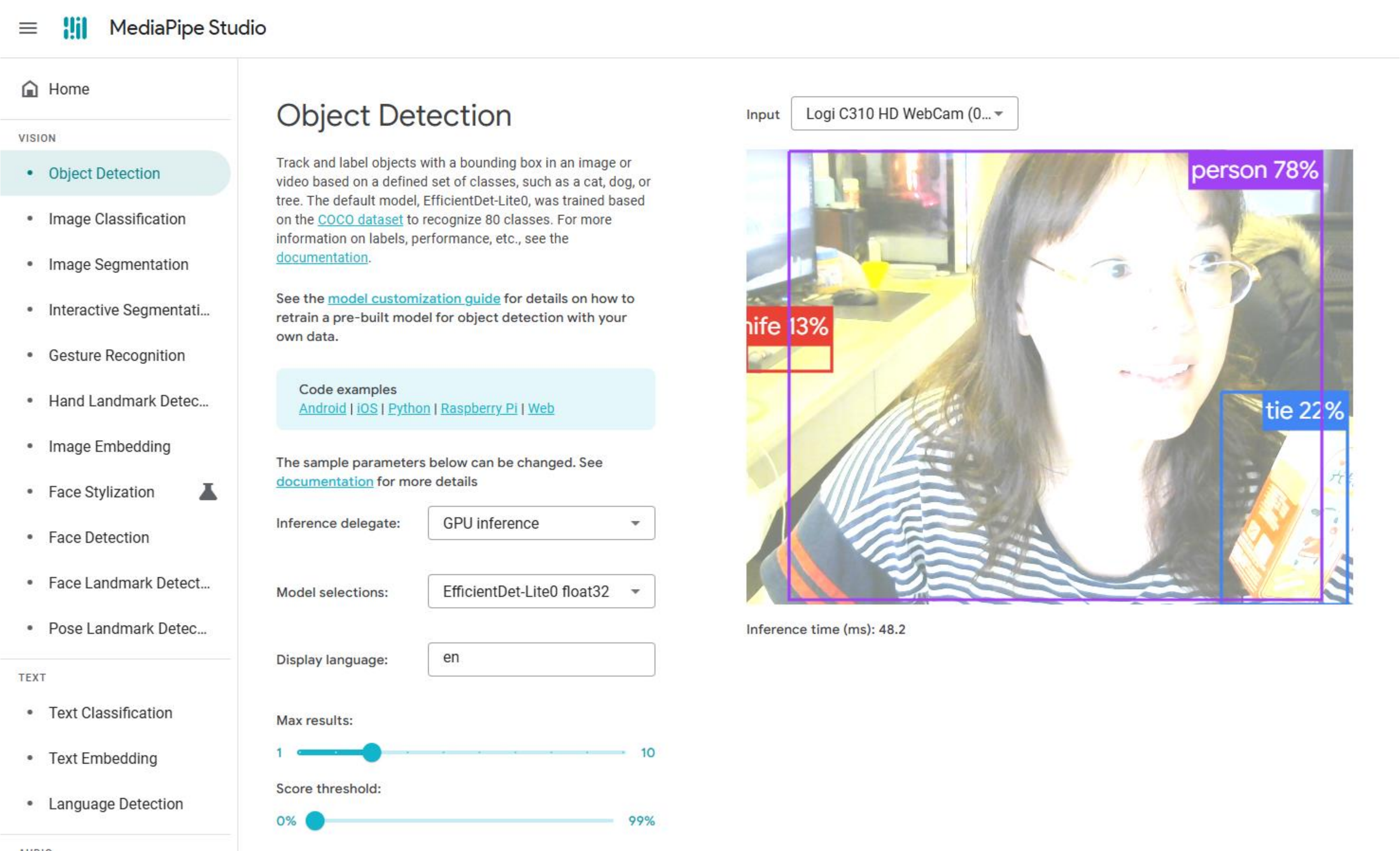
Output type: Category mask

Input: Logi C310 HD WebCam (0...

Inference time (ms): 50.6

Click the buton to reset the demo. Click or drag on the image to segment.

Click the buttons to undo or redo your click interactions.



Object Detection

Track and label objects with a bounding box in an image or video based on a defined set of classes, such as a cat, dog, or tree. The default model, EfficientDet-Lite0, was trained based on the COCO dataset to recognize 80 classes. For more information on labels, performance, etc., see the documentation.

See the model customization guide for details on how to retrain a pre-built model for object detection with your own data.

Code examples
Android | iOS | Python | Raspberry Pi | Web

The sample parameters below can be changed. See documentation for more details

Inference delegate: GPU inference

Model selections: EfficientDet-Lite0 float32

Display language: en

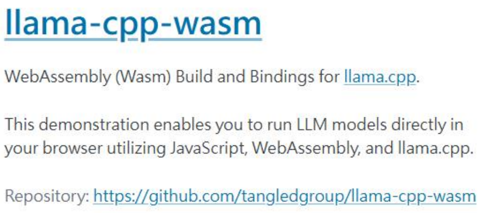
Max results: 1 — 10

Score threshold: 0% — 99%

Input: Logi C310 HD WebCam (0...

Inference time (ms): 48.2

在網頁中執行llama.cpp



llama-cpp-wasm

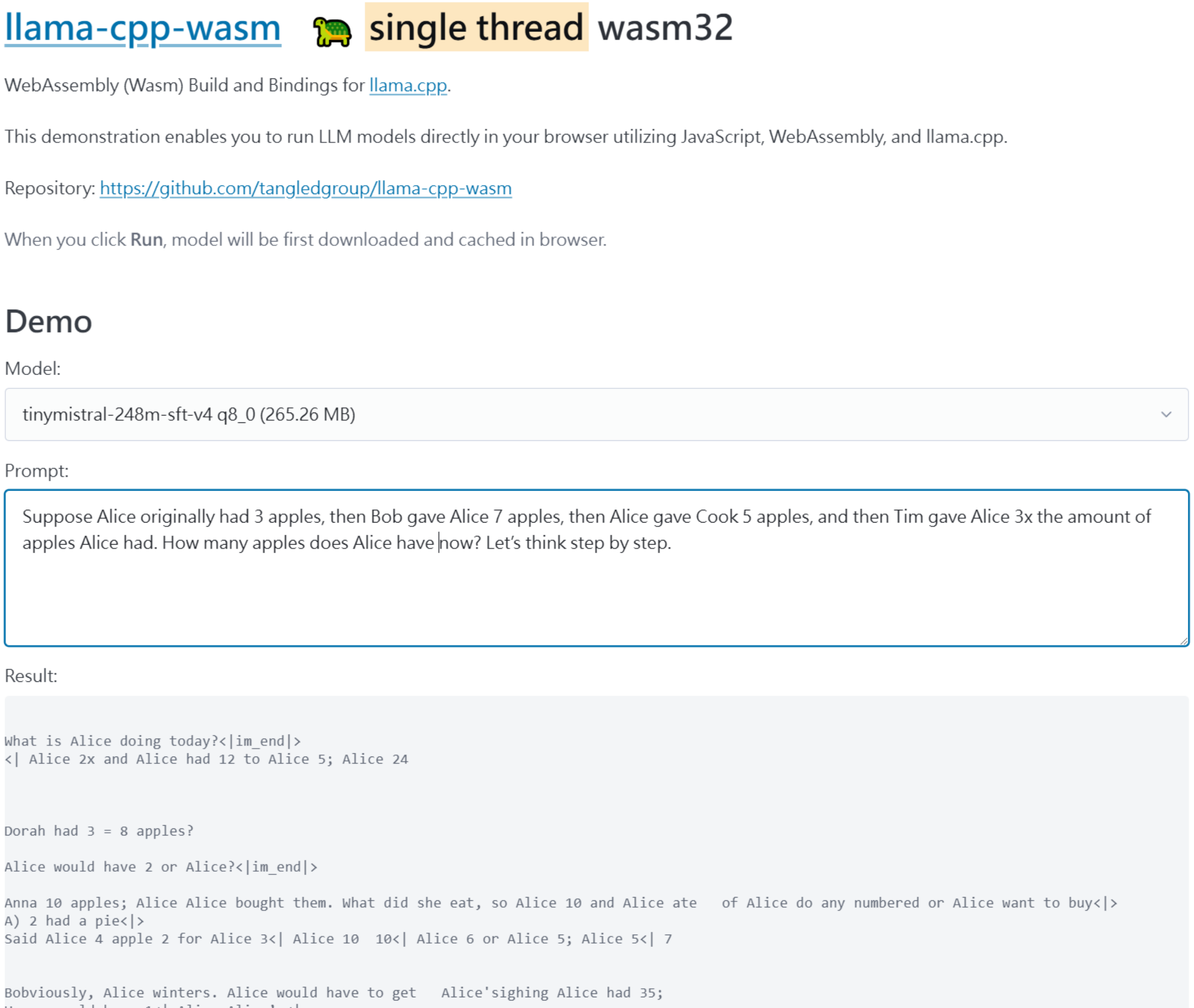
WebAssembly (Wasm) Build and Bindings for llama.cpp.

This demonstration enables you to run LLM models directly in your browser utilizing JavaScript, WebAssembly, and llama.cpp.

Repository: https://github.com/tangledgroup/llama-cpp-wasm

掃我試玩！

Women Techmakers
Taipei



llama-cpp-wasm 🐢 single thread wasm32

WebAssembly (Wasm) Build and Bindings for llama.cpp.

This demonstration enables you to run LLM models directly in your browser utilizing JavaScript, WebAssembly, and llama.cpp.

Repository: https://github.com/tangledgroup/llama-cpp-wasm

When you click **Run**, model will be first downloaded and cached in browser.

Demo

Model:

tinymistral-248m-sft-v4 q8_0 (265.26 MB)

Prompt:

Suppose Alice originally had 3 apples, then Bob gave Alice 7 apples, then Alice gave Cook 5 apples, and then Tim gave Alice 3x the amount of apples Alice had. How many apples does Alice have now? Let's think step by step.

Result:

```
What is Alice doing today?<|im_end|>
<| Alice 2x and Alice had 12 to Alice 5; Alice 24



Dorah had 3 = 8 apples?

Alice would have 2 or Alice?<|im_end|>

Anna 10 apples; Alice Alice bought them. What did she eat, so Alice 10 and Alice ate   of Alice do any numbered or Alice want to buy<|>
A) 2 had a pie<|>
Said Alice 4 apple 2 for Alice 3<| Alice 10  10<| Alice 6 or Alice 5; Alice 5<| 7


Bobviously, Alice winters. Alice would have to get   Alice'sighing Alice had 35;
```

Women Techmakers

Thank you