

# AI in Your Browser with WebGPU

**Commit Your Code**

September 2025

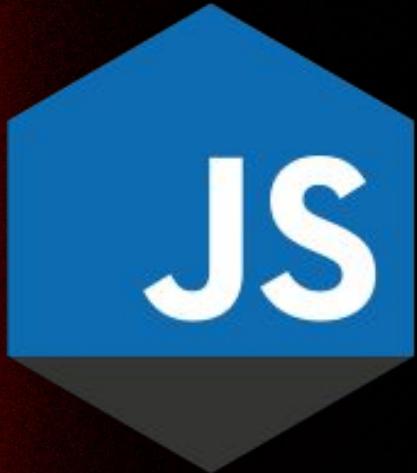


[bit.ly/CYC25-WebGPU](https://bit.ly/CYC25-WebGPU)

**Patrick Hulce**

patrick.hulce@gmail.com

# MY INSPIRATION...

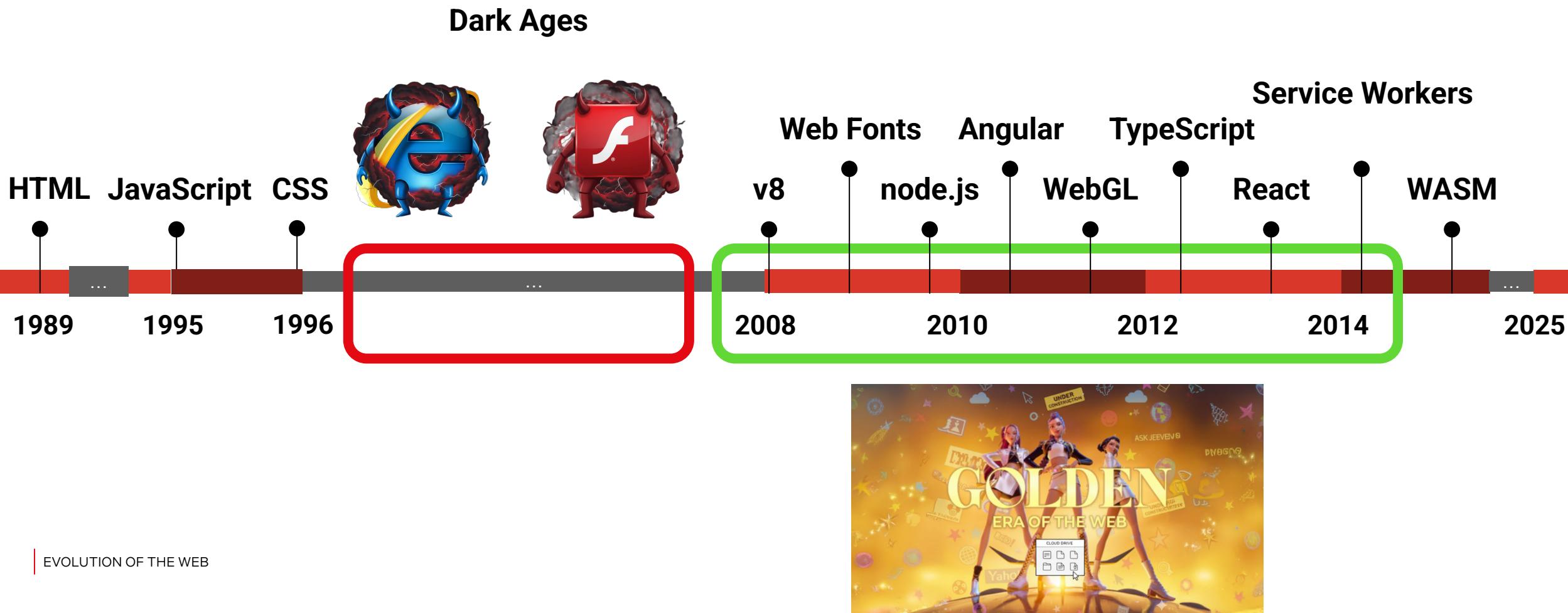


# AGENDA



- Evolution of the Web
  - How far have we come?
- WebAssembly
  - How did we get to WebGPU?
- WebGPU
  - What is it?
  - How do you use with it?
- Video Captioning System
  - How do you use WebGPU *practically*?
- Live Demo
  - Does it really work?
- The Future
  - What's next?

# Evolution of the Web



# WebAssembly

**How did we get to WebGPU?**



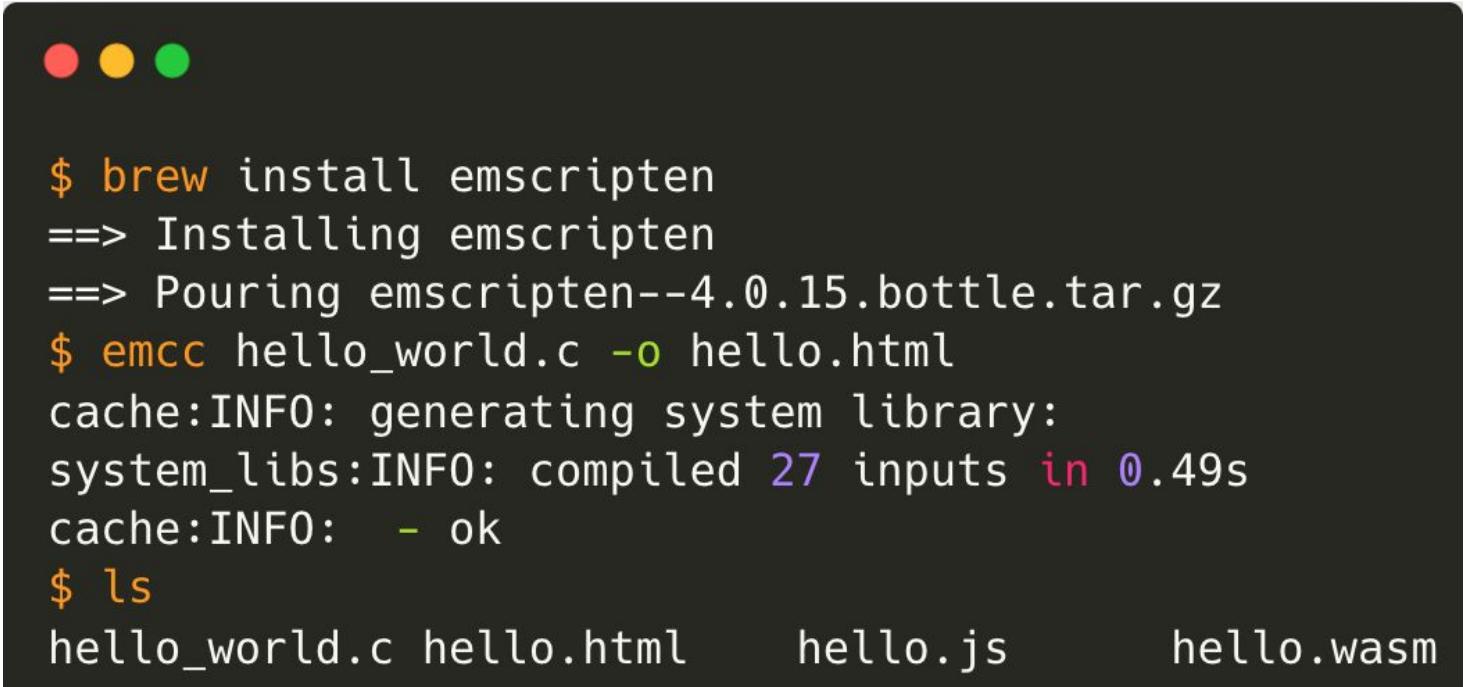
# WebAssembly



# WebAssembly - Hello World

```
#include <stdio.h>

int main() {
    printf("Hello Web!\n");
    return 0;
}
```



A terminal window showing the compilation of a C program to WebAssembly using emscripten. The terminal has three colored window control buttons (red, yellow, green) at the top.

```
$ brew install emscripten
==> Installing emscripten
==> Pouring emscripten--4.0.15.bottle.tar.gz
$ emcc hello_world.c -o hello.html
cache:INFO: generating system library:
system_libs:INFO: compiled 27 inputs in 0.49s
cache:INFO: - ok
$ ls
hello_world.c hello.html    hello.js      hello.wasm
```

# WebAssembly - Hello World



Hello Web!

A screenshot of a browser developer tools Network tab. The timeline shows a single request labeled "hello.wasm" with a duration of 150 ms. The response tab shows the raw JavaScript code for the "hello.wasm" file. The code includes a try-catch block for stdio streams and a "run()" function. A large red circle highlights the number "2208" in the response body, with a red arrow pointing from it to a thought bubble containing a sad face emoji. Another red circle highlights the text "CRITICAL FILE!" above the file list. A red arrow points from this circle to the "114 KB?! IS THIS TYPO?! IT'S WHAT JUST TEXT, THIS MADNESS?" text below. A large red arrow points from the "2208" circle to a graphic of two eyes with blood streaming down. The bottom of the slide features bold red text: "THE FUTURE OF FRONTEND IS HERE! AND IT HATES YOU".

powered by emscripten

CRITICAL FILE!

2208 LINES?! FOR HELLO WORLD?! MY EYES BLEED!

114 KB?! IS THIS TYPO?! IT'S WHAT JUST TEXT, THIS MADNESS?

THE FUTURE OF FRONTEND IS HERE!  
AND IT HATES YOU

5 / 8 requests | 114 kB / 1,160 kB transferred | Line 208, Column 1

What's new AI assistance Sensors

The image displays two software interfaces side-by-side. On the left is the Adobe Photoshop interface, showing a close-up photograph of an elephant's head emerging from water. The Photoshop toolbar is visible on the left, and a large blue 'Ps' logo is overlaid on the bottom-left corner of the image area. A context menu is open, listing selection tools like Lasso, Quick selection, Magic wand, etc., with 'Quick selection' checked. The top menu bar shows 'Elephant' and '100%'. On the right is the FFmpeg transcoding interface, featuring a large 'FFmpeg' logo at the top. It includes a file system browser, an editor window with command-line arguments, a console output window, and a progress bar. The FFmpeg logo is also overlaid on the top-right corner of the image area.

Ps Adobe Photoshop

Elephant 100%

Sample all layers Enhance edges Use pressure for size

Open in Photoshop app

Layers

Background

Selection tools

- Lasso
- Quick selection
- Magic wand
- Rectangular marquee
- Elliptical marquee

Actions

- Select subject
- Remove background

File System:

Path: /

- ..
- tmp
- home
- dev
- proc
- video.webm
- video.mp4

LOAD SAMPLE FILES

Editor:

Edit arguments below to update command:

```
1 -i  
2 "video.webm",  
3 "video.mp4"  
4  
5
```

// equivalent ffmpeg.wasm API call  
ffmpeg.exec(["-i", "video.webm", "video.mp4"]);

// equivalent ffmpeg command line  
ffmpeg -i video.webm video.mp4

Console Output:

```
70 [libx264 @ 0xdf6070] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu  
71 [libx264 @ 0xdf6070] i8c dc,h,v,p: 50% 22% 15% 13%  
72 [libx264 @ 0xdf6070] Weighted P-Frames: Y:0.0% UV  
73 [libx264 @ 0xdf6070] ref P L0: 64.1% 18.8% 10.0%  
74 [libx264 @ 0xdf6070] ref B L0: 90.4% 8.2% 1.4%  
75 [libx264 @ 0xdf6070] ref B L1: 96.4% 3.6%  
76 [libx264 @ 0xdf6070] kb/s:217.13  
77 Aborted()
```

Transcoding Progress:

100%

Time Elapsed: 2.82 s

RUN

WEBASSEMBLY

# WebGPU

**What can it do?**

# WebGPU

- *Direct* access to GPUs' parallel compute
- Not just for graphics, general purpose parallel computing API
- **2023** - Shipped in Chrome
- **2025** - Shipped across browsers (Firefox in July, Safari in macOS Tahoe)



# WebGPU

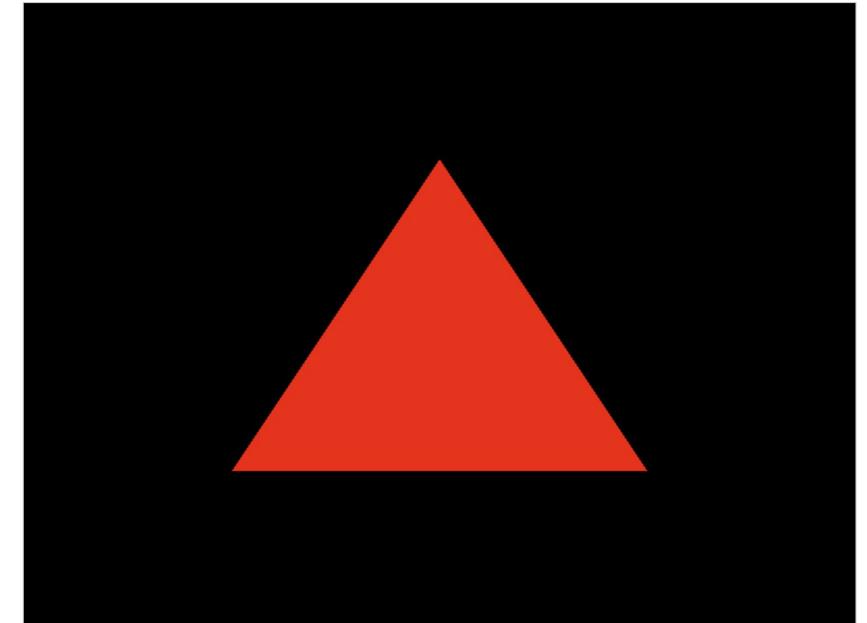
# WebGPU - Hello Triangle

```
const adapter = await navigator.gpu.requestAdapter();
const device = await adapter.requestDevice();

const vertexShader = device.createShaderModule({
  code: `
    @vertex fn main(@builtin(vertex_index) vertexIndex: u32)
      -> @builtin(position) vec4<f32> {
        // Triangle vertices
        var pos = array<vec2<f32>, 3>(
          vec2(0.0, 0.5),
          vec2(-0.5, -0.5),
          vec2(0.5, -0.5)
        );
        return vec4<f32>(pos[vertexIndex], 0.0, 1.0);
    }
  });

// CHEATING: ~100 more lines of render pipeline boilerplate...
```

Hello WebGPU Triangle



**SCORE: 2000**

**LIVES: 2**

WebGPU

Stats - fps: 60  
Rendering Options  
renderLightSprites  
renderEnvironment  
environmentLights  
metaballLights  
metaballMethod  
metaballStyle  
metaballResolution

WEBGPU

+ New Chat

# Chat GPT

How can I help you today?



Examples

"Explain quantum computing in simple terms" →



Capabilities

Remembers what user said earlier in the conversation



Limitations

May occasionally generate incorrect information

"Get any creative ideas for a 10 year old's birthday?" →

Allows user to provide follow-up corrections

May occasionally produce harmful instructions or biased content

"How do I make an HTTP request in Javascript?" →

Trained to decline inappropriate requests

Limited knowledge of world and events after 2021

Light Mode

OpenAI Discord

Updates & FAQ

Log out

ChatGPT Dec.15 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help.

## Diffusion

UNet Step 1

UNet Step 2

...

UNet Step 4

UNet Step 5

...

UNet Step 10

...

UNet Step 30

...



Image Information Creator

Image Decoder  
(Autoencoder  
decoder)

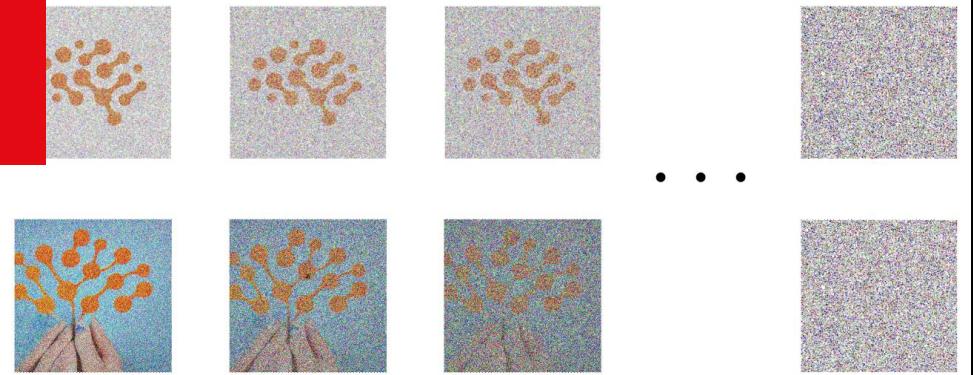


AI



Midjourney

Forward (train)



$x_0$

$x_1$

$x_2$

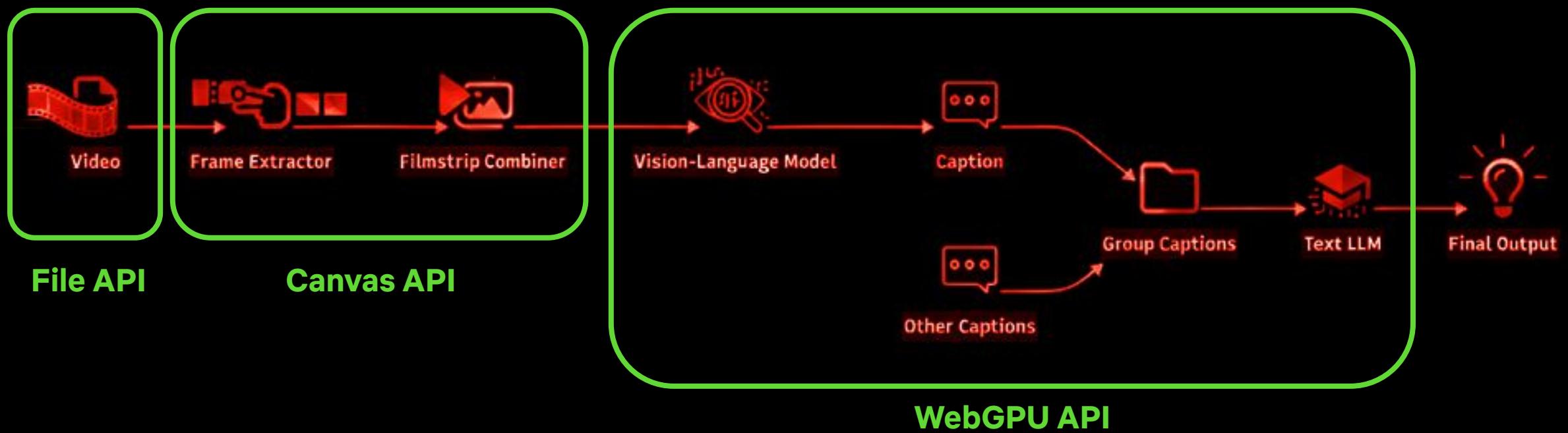
$x_3$

...

$x_{1000}$

Reverse (generate)

# MEDIA SEARCH: VIDEO CAPTIONING



# File API

## HTML

```
<input id="da-video" type="file" />
```

## JavaScript

```
const fileInput = document.getElementById('da-video');

fileInput.addEventListener('change', async (event) => {
    const file = event.target.files[0];
    const video = document.createElement('video');
    video.src = URL.createObjectURL(file);
    document.body.append(video);
    video.play();
});
```

# Canvas API

```
function extractFrames(video: HTMLVideoElement) {  
  const canvas = document.createElement('canvas');  
  const ctx = canvas.getContext('2d');  
  
  canvas.width = video.videoWidth;  
  canvas.height = video.videoHeight;  
  
  const frames = [];  
  const timestamps = [0, video.duration/2, video.duration]; // Start, middle, end  
  
  timestamps.forEach(time => {  
    video.currentTime = time;  
    ctx.drawImage(video, 0, 0);  
    frames.push(canvas.toDataURL());  
  });  
  
  return frames;  
}
```

# A Pythonic Diversion - Transformers

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Create a text generation pipeline
tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-medium")
model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-medium")

# Encode the message
msg_tokens = tokenizer.encode(
    "Hello, how are you?" + tokenizer.eos_token,
    return_tensors='pt'
)

# Generate a response
output = model.generate(msg_tokens, max_length=100, num_return_sequences=1)
print(tokenizer.decode(output[0], skip_special_tokens=True))
```

# WebGPU Transformers.js

```
import { pipeline } from '@huggingface/transformers';

// Create a text generation pipeline
const generator = await pipeline(
  'text-generation',
  'microsoft/DialoGPT-medium',
  { device: 'webgpu' }
);

// Generate a response
const output = await generator('Hello, how are you?', {
  max_length: 50,
  num_return_sequences: 1,
});

console.log(output[0].generated_text);
```

# **VISION LANGUAGE MODEL (VLM)**

a **multimodal** large language model capable of  
processing **both images and text**

# **FastVLM**

(Apple, CVPR 2025)

a small and fast vision language model that's 10x faster  
on mobile devices than prior state-of-the-art

# FastVLM Transformers.js

```
import { pipeline } from '@huggingface/transformers';

// Create a text generation pipeline
const generator = await pipeline(
  'image-to-text',
  'apple/FastVLM-7B',
  { device: 'webgpu' }
);

// Generate a response
const output = await generator(imageElement, {
  prompt: 'Describe what's happening in this video frame',
  num_return_sequences: 1,
});

console.log(output[0].generated_text);
```

# Filmstrip Method



# LIVE DEMO

**bit.ly/CYC25-WebGPU-Demo**



# The Future

**What's on the horizon?**

# WebNN to the Rescue

- **Web Neural Network API** already landed in Chrome/Edge
- Leverages any available hardware including **Neural Processing Units (NPUs)**
- The marketing fluff “AI PC” will finally become useful!



# WebNN to the Rescue

1. Go to chrome://flags
2. Search for “WebNN”



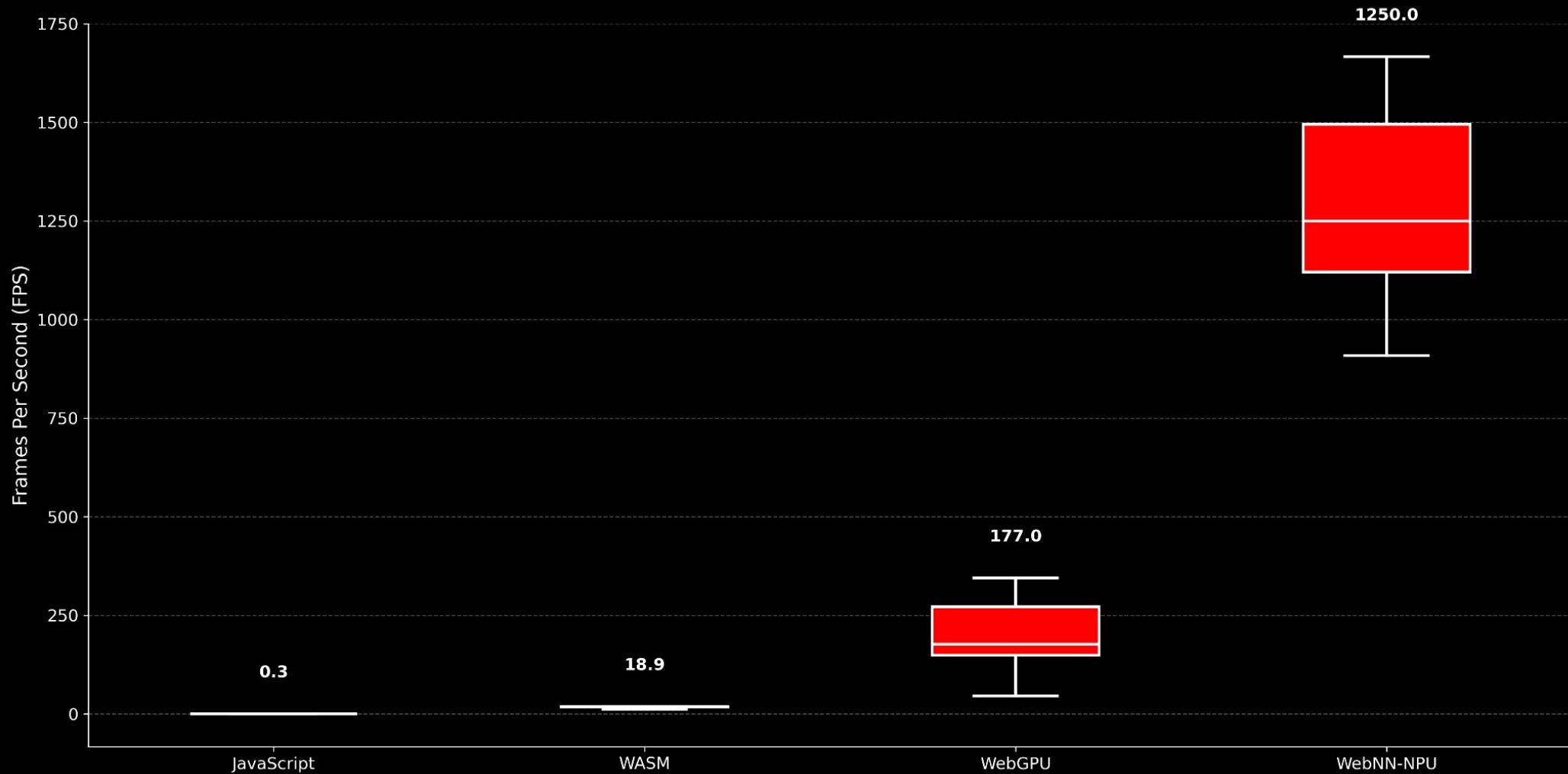
WEBNN

A screenshot of a Mac OS X desktop showing a Chrome browser window. The window title is "Experiments" and the URL is "chrome://flags". A search bar at the top contains the text "webnn". Below the search bar, the word "Experiments" is displayed. The page lists several experimental flags under the "Available" tab. The first flag is "Enables WebNN API", which is described as enabling the Web Machine Learning Neural Network (WebNN) API. It includes a link to the specification at <https://www.w3.org/TR/webnn/>. The status for this flag is "Enabled". The second flag is "Enables experimental WebNN API features", which enables additional experimental features in the WebNN API. It also includes a link to the specification at [#experimental-web-machine-learning-neural-network](#). Its status is also "Enabled". The third flag is "Core ML backend for WebNN", which enables using Core ML for GPU and NPU inference with the WebNN API. It includes a link to the specification at [#webnn-coreml](#). Its status is "Enabled". At the bottom of the list, there is a note about instructing Core ML to use GPU or Neural Engine explicitly, with a link to the specification at [#webnn-coreml-explicit-gpu-or-npu](#). The status for this note is "Default".

Flag	Status
Enables WebNN API	Enabled
Enables experimental WebNN API features	Enabled
Core ML backend for WebNN	Enabled
Instruct Core ML to use GPU or Neural Engine explicitly	Default

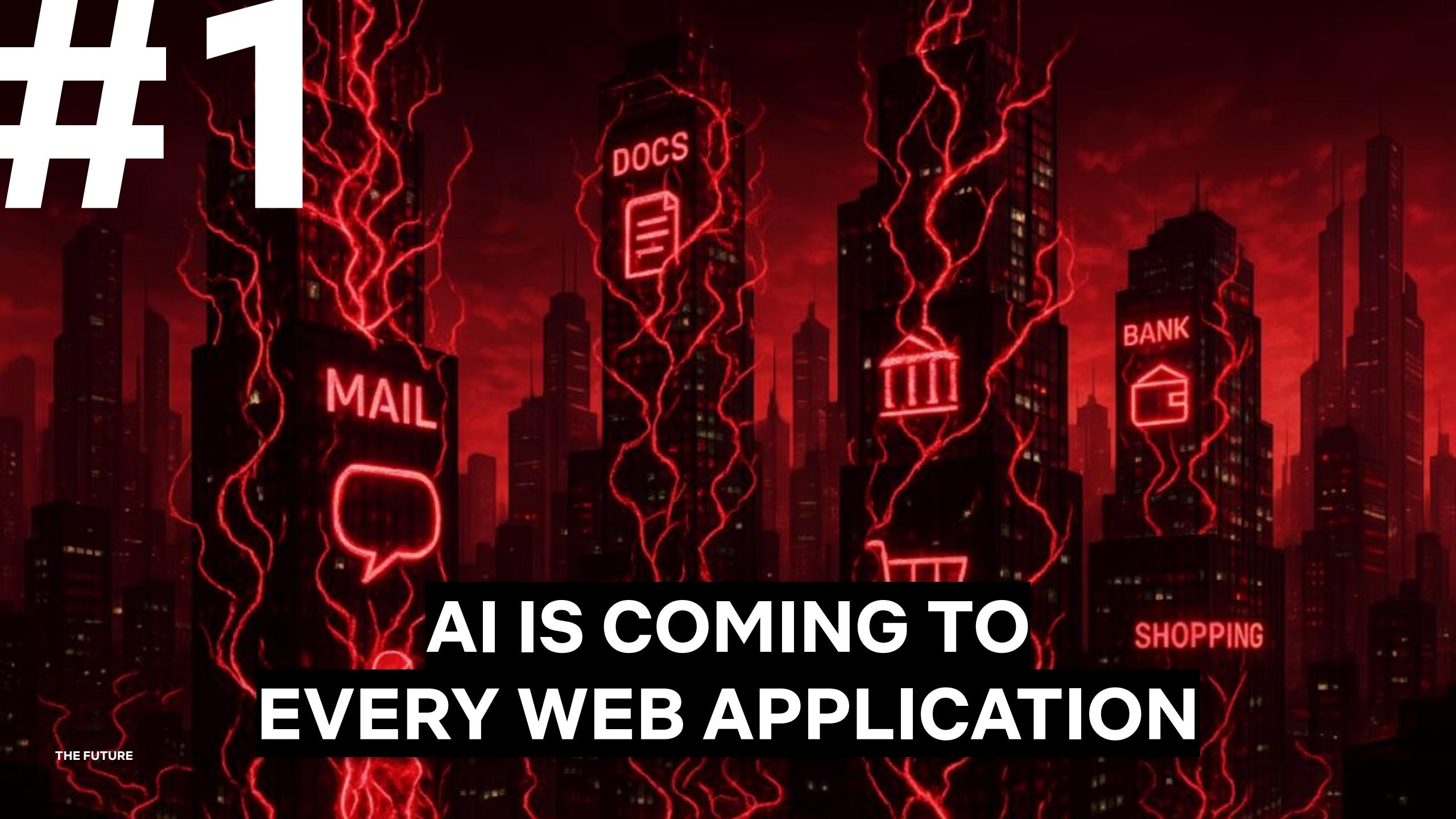
# WebNN vs. WebGPU

Image Classification Frames per Second, Macbook M3 Pro



# Limitations

- Models are BIG
  - “Tiny” FastVLM is a ~300MB download
- Accuracy leaves a lot to be desired
  - Not close to frontier models (GPT-5, Gemini, Claude)
- Choose the appropriate performance-privacy-cost tradeoff for you!



#1  
**AI IS COMING TO  
EVERY WEB APPLICATION**

# #2



## PRIVACY PRESERVING AI

# #3

YOU get AI

YOU get AI

## ACTUAL DEMOCRATIZATION

# Thank You.

 @ patrickhulce

 patrick.hulce@gmail.com

 /in/patrickhulce

