Nearform_

View online

Download PDF

Download PPTX

AI in the Browser
Am I Still Waiting for It?__

Luca Del Puppo Nearform_

N_

Hi 👋, I'm **Luca**!_



Nearform_ AI Native Engineer

GDE Web Technologies

delpuppo.net

puppo

puppo92

lucadelpuppo



We’re an independent team of engineers, designers and strategists who build digital solutions and enhanced capability at pace for ambitious enterprises seeking enduring business impact.

10+ Years experience

700+ Engagements

400+ Nearformers

380+ Customers

28 Countries



Nearform’s commitment to Open Source

15m+ Downloads	Over 15 million downloads every week
1192k+ Contributions	Thousands of contributions every year
166+ Packages	We maintain many popular packages for the community

We’re hiring! **nearform.com/careers**

Nearform_ N_

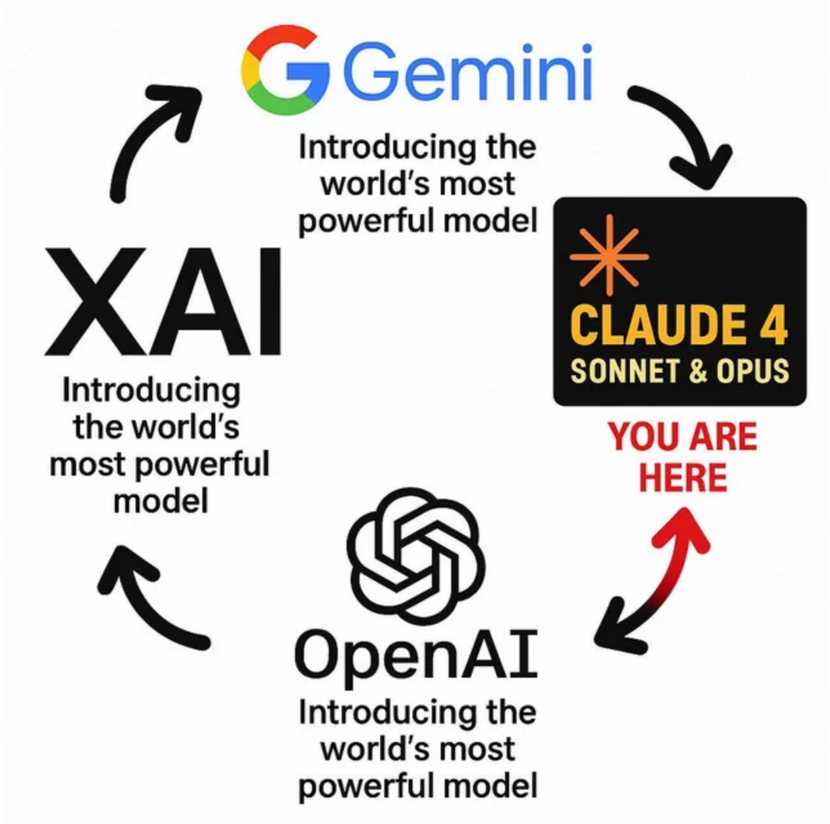
How the world looks like today_



How we are changing development products_



What are the news every day?



The Status Quo: Cloud-First AI_

1 **Most familiar LLMs run on remote servers**
GPT-4, large Gemini models

2 **This approach grants access to massive models**
Hundreds of billions of parameters

3 **Pros**
Instant scalability, access to the latest, most powerful models

4 **The Catch**
We often overlook the hidden costs and dependencies of this architecture

The Cost & Latency Bottleneck 1/2

High Cost
Operating on a "pay-per-token" model

Low usage is cheap
But high volume/long-term use can be 2x to 3x more expensive than running locally

Costs are unpredictable
Scale rapidly with adoption

The Cost & Latency Bottleneck 2/2

Network Latency
Inference requests must travel over the internet

Introduces unavoidable delays (latency)
that make true real-time interaction difficult

Stable internet connectivity
Required for every interaction

Privacy and Control Concerns

Data is Leaving the Device

Privacy Risk: Sending sensitive user data to a third-party server (API provider) is a non-starter for many compliance-sensitive applications

Loss of Control

Limited visibility into the model's execution

Vendor lock-in

Dependency on specific API providers

Zero offline capability

No internet connection means no AI assistance

The Rise of Local LLMs

The Need for Local Intelligence_

Definition

Local LLMs are models downloaded and run directly on the user's device (PC, phone, tablet)

The Enabler

Breakthroughs in Model Quantization - reducing the model's precision to drastically cut size and memory demands while preserving capability

The Local Promise: Speed and Privacy

1 **Superior Privacy**
100% client-side computation. Data never leaves the device or network perimeter

2 **Low Latency**
Inference occurs locally, eliminating network lag for near-instant responses

3 **Offline Functionality**
Works anywhere, anytime, once the model is loaded

4 **Cost-Effective**
Zero API costs everything run in the user's environment

But the Web standards evolve slowly_



WebLLM: LLMs in the Browser

WebLLM_

WebLLM
(developed by the MLC-AI team) is a high-performance, in-browser inference engine

Goal
Bring powerful, capable LLMs directly into the standard web browser

Vision
Run LLMs locally just by opening a webpage—zero environment setup

The WebLLM Technology Stack_

WebGPU
A new graphics API that grants web code efficient access to the local GPU for tensor operations (matrix math)

WebAssembly (WASM)
Provides a near-native execution environment for model computations, ensuring high speed

Caching
The model weights (quantized) and WASM modules are downloaded once and cached locally

WebLLM for Developers_

1 **Familiarity**

Provides an OpenAI-API style interface for easy integration into existing chat and generation apps

2 **Performance**

Uses Web Workers to run the heavy inference engine on a separate thread, keeping the main UI fast and responsive

3 **Key Takeaway**

WebLLM proved that full LLM inference is technically feasible within the standard, open-source web stack

How the interface looks likes 1/2_

```
import { CreateMLCEngine, MLCEngine } from "@mlc-ai/web-llm";

// Initialize with a progress callback
const initProgressCallback = (progress) => {
    console.log("Model loading progress:", progress);
};

// Using CreateMLCEngine
const engine = await CreateMLCEngine("Llama-3.1-8B-Instruct", { initProgressCallback });

// Direct instantiation
const engineInstance = new MLCEngine({ initProgressCallback });
await engineInstance.reload("Llama-3.1-8B-Instruct");
```



How the interface looks likes 2/2_

```
const messages = [
    { role: "system", content: "You are a helpful AI assistant." },
    { role: "user", content: "Hello!" }
];

const reply = await engine.chat.completions.create({
    messages,
});

console.log(reply.choices[0].message);
console.log(reply.usage);
```

Links and Videos

Hands-on Chat Demo
https://chat.webllm.ai/

WebLLM: A high-performance in-browser LLM Inference engine
https://www.youtube.com/watch?v=MhTCzq7iTy0

Built-in AI in Chrome: Gemini Nano

Native AI: Built-in to the Browser_

Chrome introduced Built-in AI features
Powered by Gemini Nano, a lightweight LLM designed for on-device use

Gemini Nano
can plug in various models based on the use case

Developers
can request download of a specific model only when needed

Goals
Seamless, deeply integrated, focused on the browsing context

Features Powered by Gemini Nano 1/2

1 **Language Detection API**
Automatically identifies the language of the webpage for tailored assistance

2 **Translator API**
Translate webpage content in real-time

3 **Summarizer API**
Get the gist of an article or webpage immediately

Features Powered by Gemini Nano 2/2

4 **Writer API**
Assist with composing emails, messages, and documents

5 **Rewriter API**
Rewrite text for clarity, tone, or style

6 **Prompt API**
General-purpose interface for custom AI interactions within the browser

7 **Proofreader API**
Check grammar and spelling in real-time

How the interface looks for developers 1/3

```
if ('LanguageDetector' in self) {
  const detector = await LanguageDetector.create({
    monitor(m) {
      m.addEventListener('downloadprogress', (e) => {
        console.log(`Downloaded ${e.loaded * 100}%`);
      });
    },
  });
  const someUserText = 'Hallo und herzlich willkommen!';
  const results = await detector.detect(someUserText);
  for (const result of results) {
    console.log(result.detectedLanguage, result.confidence);
  }
}
```

How the interface looks for developers 2/3

```
if ('Translator' in self) {
  const translator = await Translator.create({
    sourceLanguage: 'es',
    targetLanguage: 'fr',
    monitor(m) {
      m.addEventListener('downloadprogress', (e) => {
        console.log(`Downloaded ${e.loaded * 100}%`);
      });
    },
  });
  const translation = await translator.translate('Where is the next bus stop, please?');
  const stream = translator.translateStreaming(longText);
  for await (const chunk of stream) {
    console.log(chunk);
  }
}
```

How the interface looks for developers 3/3

```
if ('Summarizer' in self) {
  const summarizer = await Summarizer.create({
    monitor(m) {
      m.addEventListener('downloadprogress', (e) => {
        console.log(`Downloaded ${e.loaded * 100}%`);
      });
    }
  });
  const summary = await summarizer.summarize(longText, {
    context: 'This article is intended for a tech-savvy audience.',
  });
  const stream = summarizer.summarizeStreaming(longText, {
    context: 'This article is intended for junior developers.',
  });
  for await (const chunk of stream) {
    console.log(chunk);
  }
}
```

Typescript types_

```
npm install -D @types/dom-chromium-ai
```



WebMCP: The Agentic Web

What is WebMCP?

A proposed browser standard
Allows web developers to expose their app functionality as "tools" that AI agents and assistive technologies can invoke

MCP-inspired, browser-native
Web pages become like MCP servers, but tools run in client-side JavaScript — no backend server needed

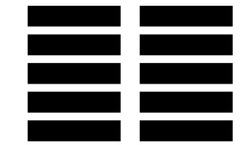
Two APIs
Declarative API for standard HTML form actions, and Imperative API for complex dynamic interactions

Human-in-the-loop
Designed for cooperative workflows where users and agents collaborate within the same web interface

WebMCP for Developers_

```
if ("modelContext" in window.navigator) {
  window.navigator.modelContext.registerTool({
    name: "add-todo",
    description: "Add a new todo item to the list",
    inputSchema: {
      type: "object",
      properties: {
        text: { type: "string", description: "The text of the todo item" }
      },
      required: ["text"]
    },
    execute({ text }, agent) {
      addTodo(text);
      return {
        content: [{ type: "text", text: `Todo "${text}" added!` }]
      };
    }
  });
}
```



Demo of Gemini Nano in Action 

Links and Videos

Built-in AI Documentation
https://developer.chrome.com/docs/ai/built-in

Build a guessing game with the Prompt API
https://developer.chrome.com/blog/ai-guessing-game?hl=en

Practical built-in AI with Gemini Nano in Chrome
https://www.youtube.com/watch?v=CjpZCWYrSxM

Lessons learned from being customer zero of Chrome's built-in APIs
https://www.youtube.com/watch?v=surh_D8CU9A

Conclusion & The Future

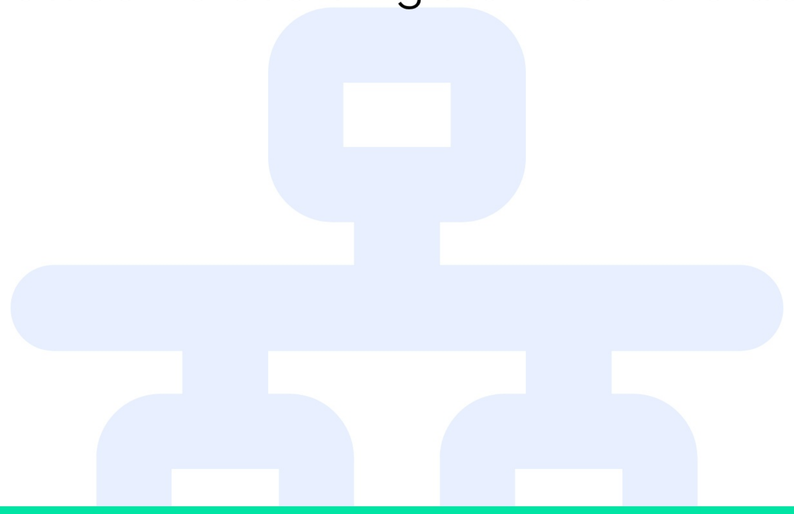
The Hybrid Future: Cloud + Client_

The future of web AI is not **Cloud vs. Local**, but **Cloud + Client**

Cloud: Handles massive, general-purpose tasks

Client: Handles fast, private, contextual tasks

The browser is evolving from a mere document viewer into the primary **AI Agent platform**



But with slow adoption_





Thank you!

Luca Del Puppo @puppo92
Nearform_ luca@delpuppo.net

Nearform_

GitHub Repos_

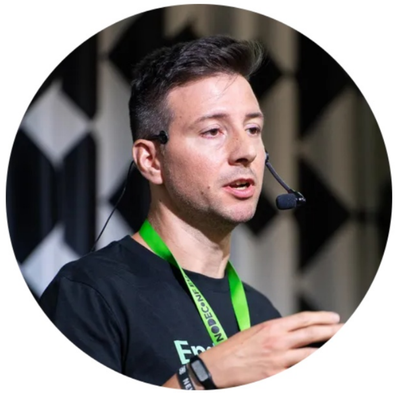
AI in the Browser

Explore the code and examples

Recipe Radar with AI

Explore the code and examples

Bye Bye , I'm **Luca**!_



Nearform_ AI Native Engineer

GDE Web Technologies

delpuppo.net puppo puppo92 lucadelpuppo



Nearform_

View online

Download PDF

Download PPTX

AI in the Browser Am I Still Waiting for It? _

Luca Del Puppo Nearform_

N_