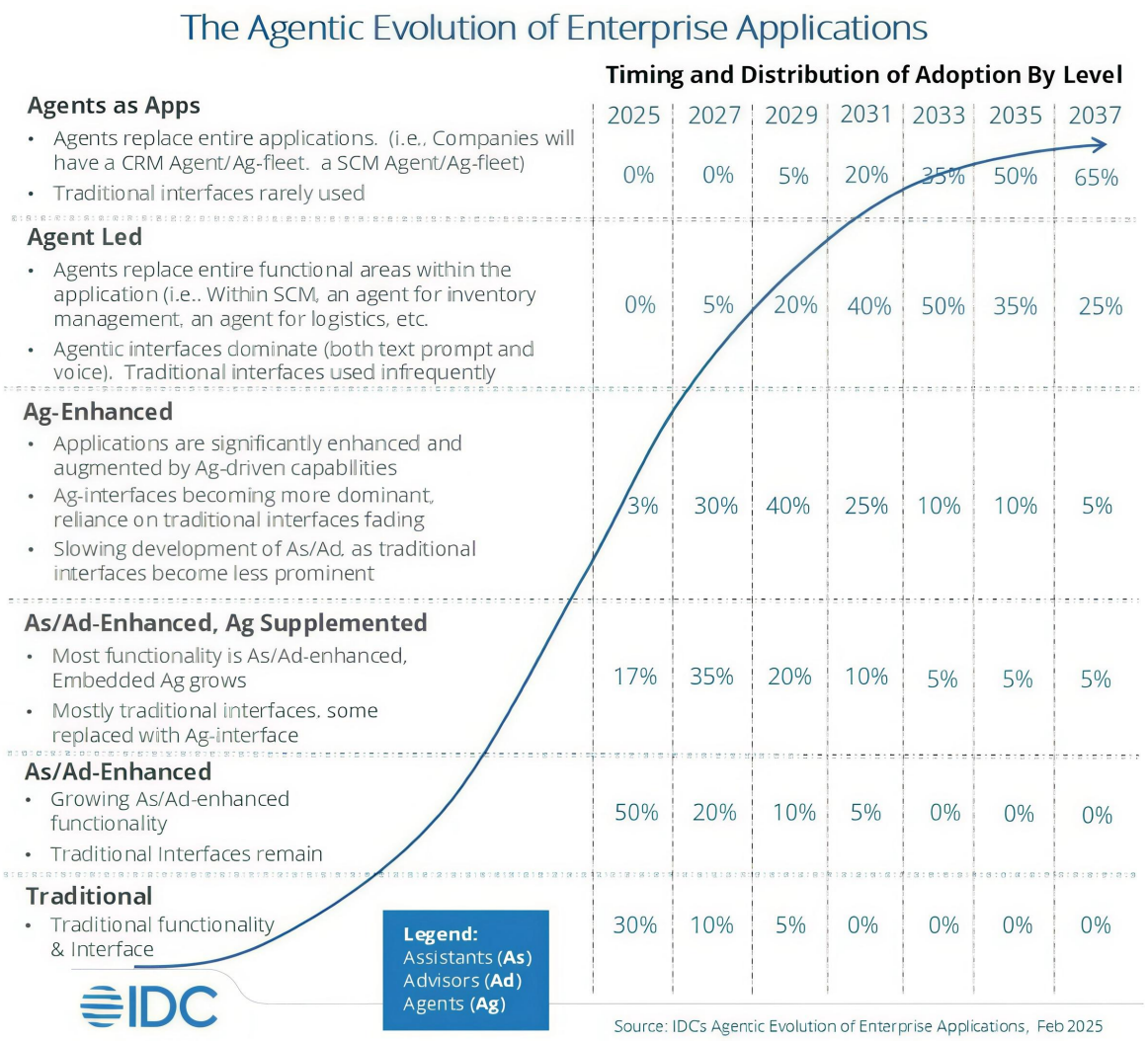
Generative UI & MCP Server
The Future of Web AI

Chunhui Mo （Huawei）

WebEvolve 2025

AI Agent & Open Web Ecosystem
（HANGZHOU）

The Agentic Evolution of Enterprise Applications



> We're at the beginning of a major transformation in which autonomous AI agents will become the new user interface.
>
> Adam Evans, EVP and GM of AI, Salesforce

Three types of agents for Multi-Agent enterprise integration

User-facing agents embedded in platforms like Slack, serve as the users direct interface, making interactions intuitive and conversational.

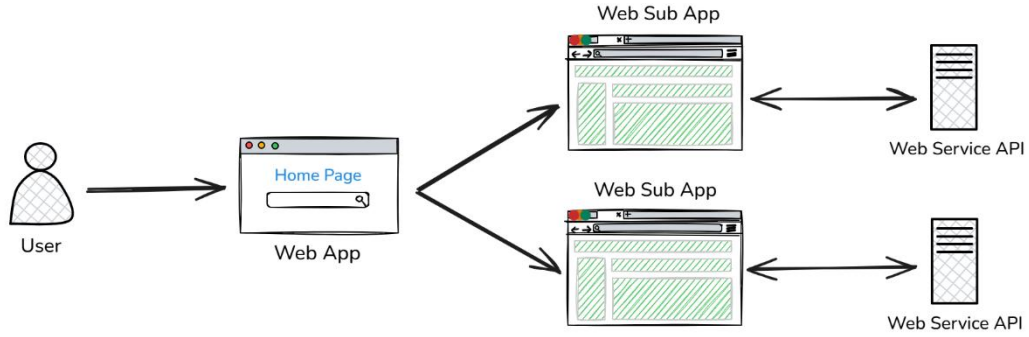
Orchestration agents work behind-the-scenes to manage task delegation and workflow coordination, ensuring that the right requests are routed to the right systems.

Specialized agents autonomously handle execution, integrating with APIs to retrieve structured and unstructured data, analyze it, and act accordingly.

The Evolution of Agentic Web Application

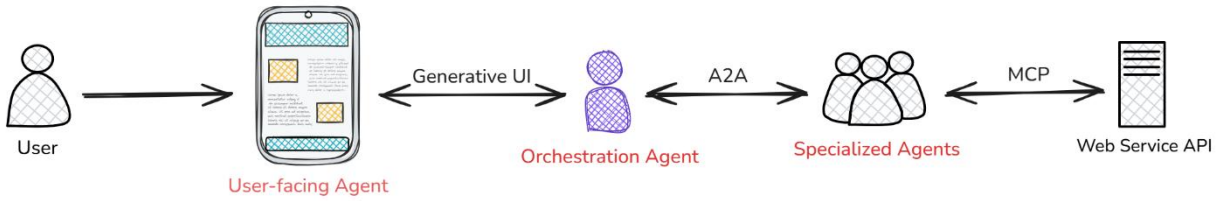
Traditional Current

Traditional application functionality and interfaces will gradually diminish, with adoption projected to fall to 0% after 2031.



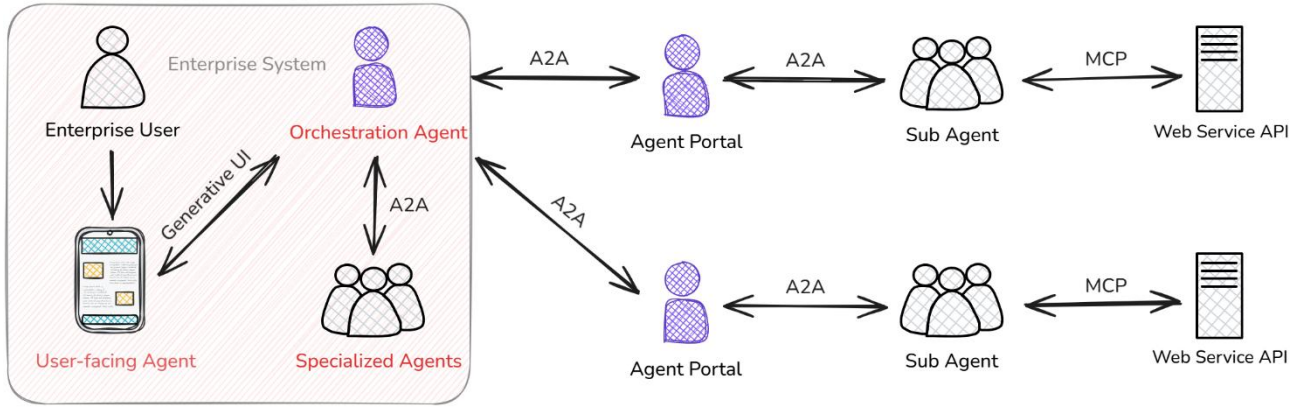
Agent Led In 3 ~ 5 years

Agents will replace entire functional areas within applications, and agent interfaces will become dominant. The use of traditional interfaces will become less frequent, with adoption projected to peak at 50% by 2031.

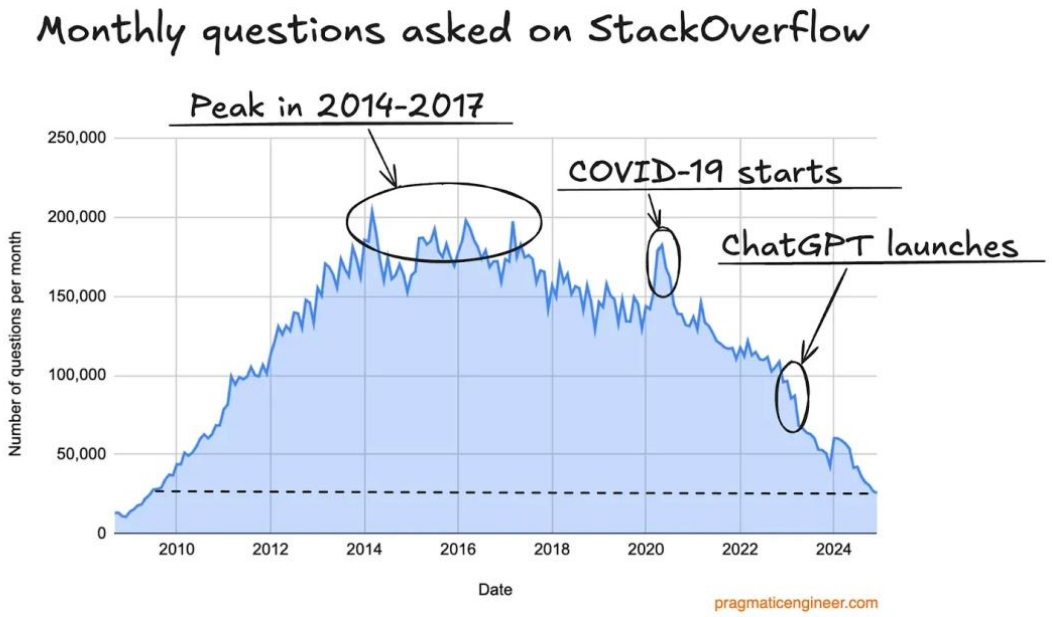


Agents as Apps In 10 years

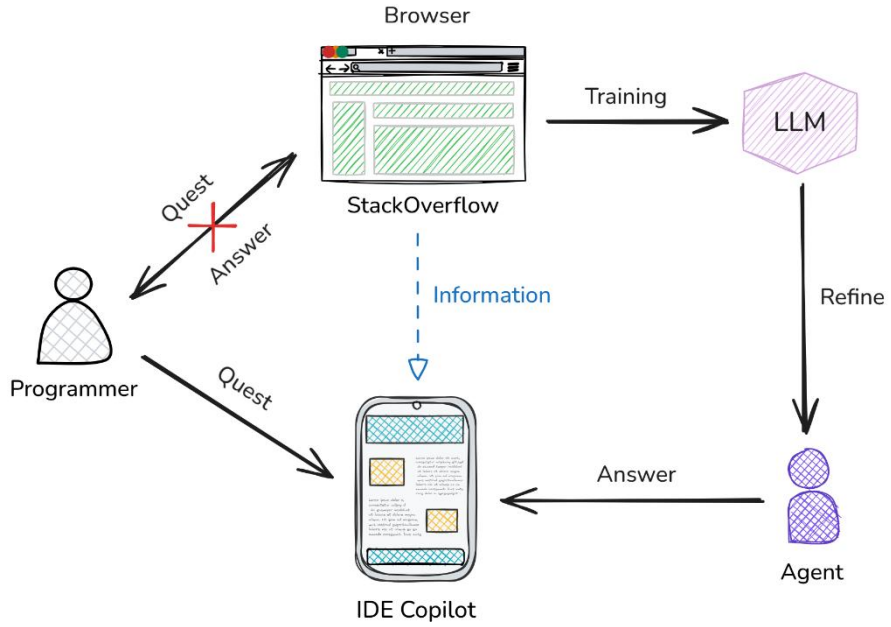
By 2037, agents will replace entire applications, and traditional interfaces will be rarely used, with adoption projected to reach 65%. Cloud providers will offer agent services directly to customers, replacing traditional service models.



Starting with the Deteriorating StackOverflow



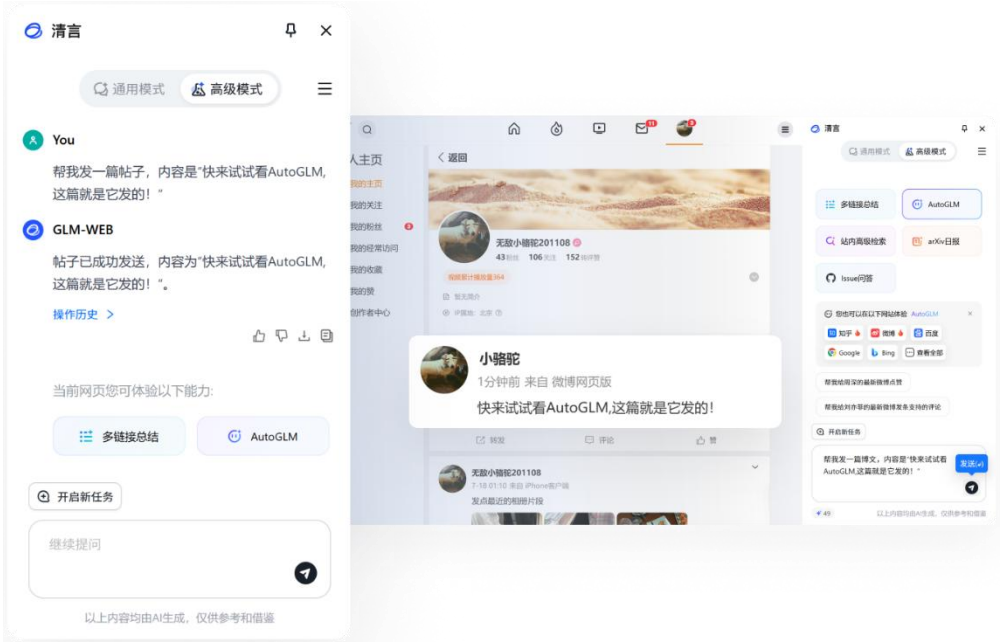
The number of questions dropped rapidly after ChatGPT was launched



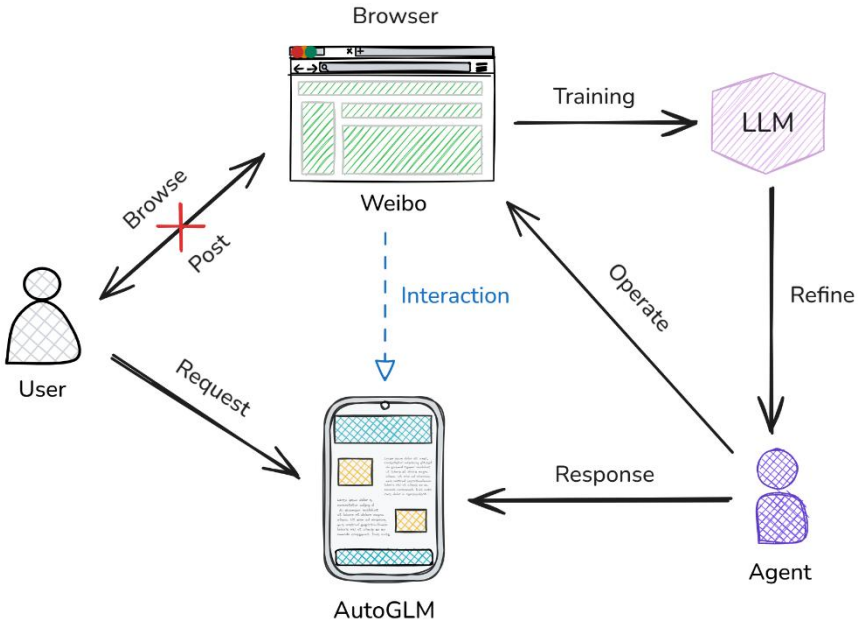
The information is being transferred from the browser to IDE Copilot

The two core functions of the browser: a channel for users to **obtain information** (information transfer), and an **interactive interface** for web applications (interaction transfer?)

Will the Interaction of Web-Based Apps Shift to AI Apps?



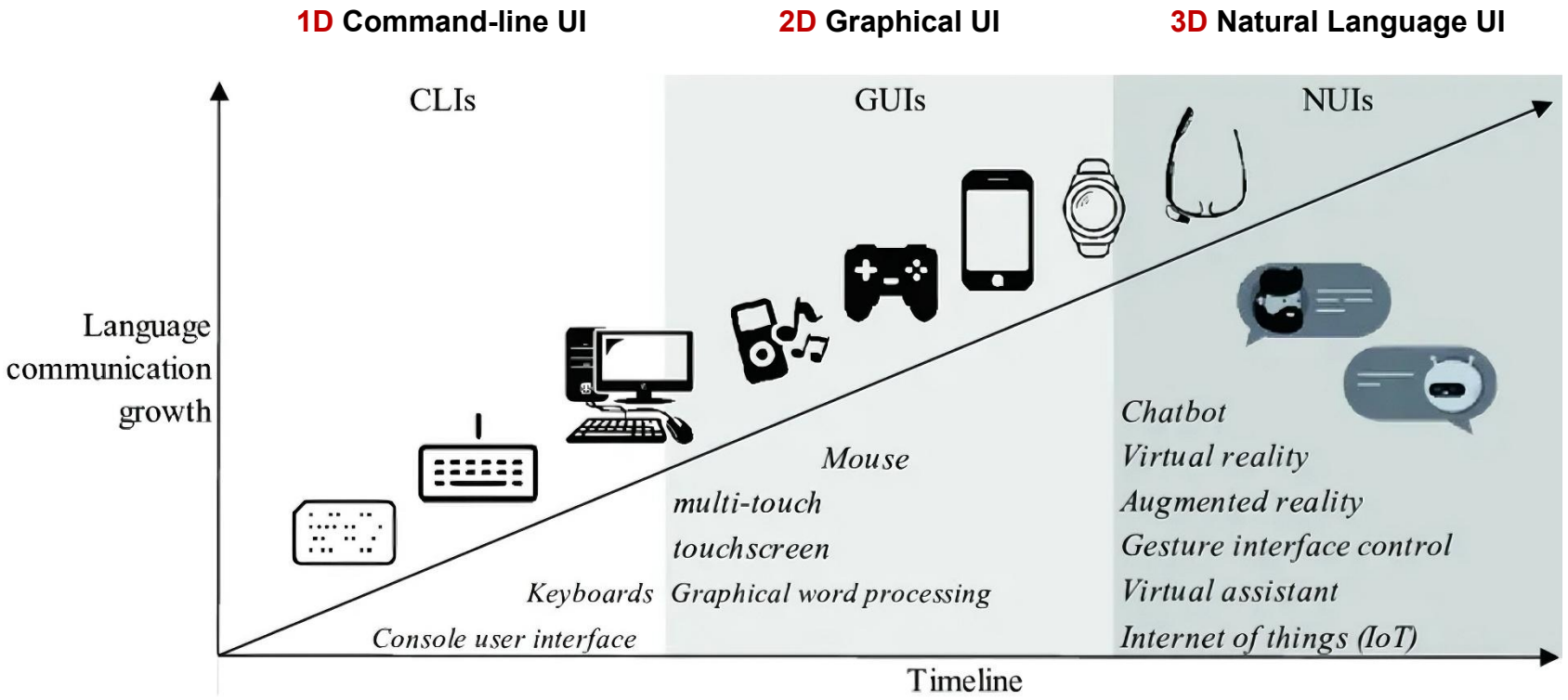
AutoGLM directly posts a tweet on Weibo by operating the browser.



The interaction on Weibo is shifting from the browser to AutoGLM

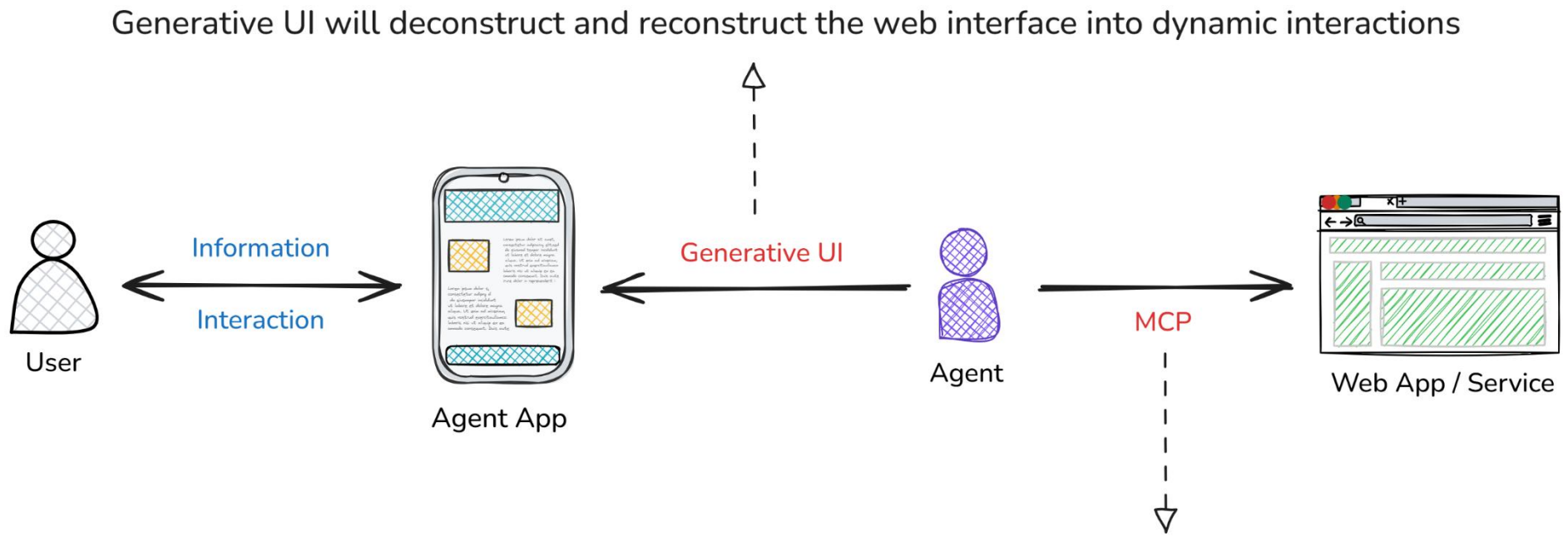
Conclusion: The traditional information and interaction handled by browsers are now **shifting to AI apps**.

From Command-line, Graphical to Natural-Language User Interface



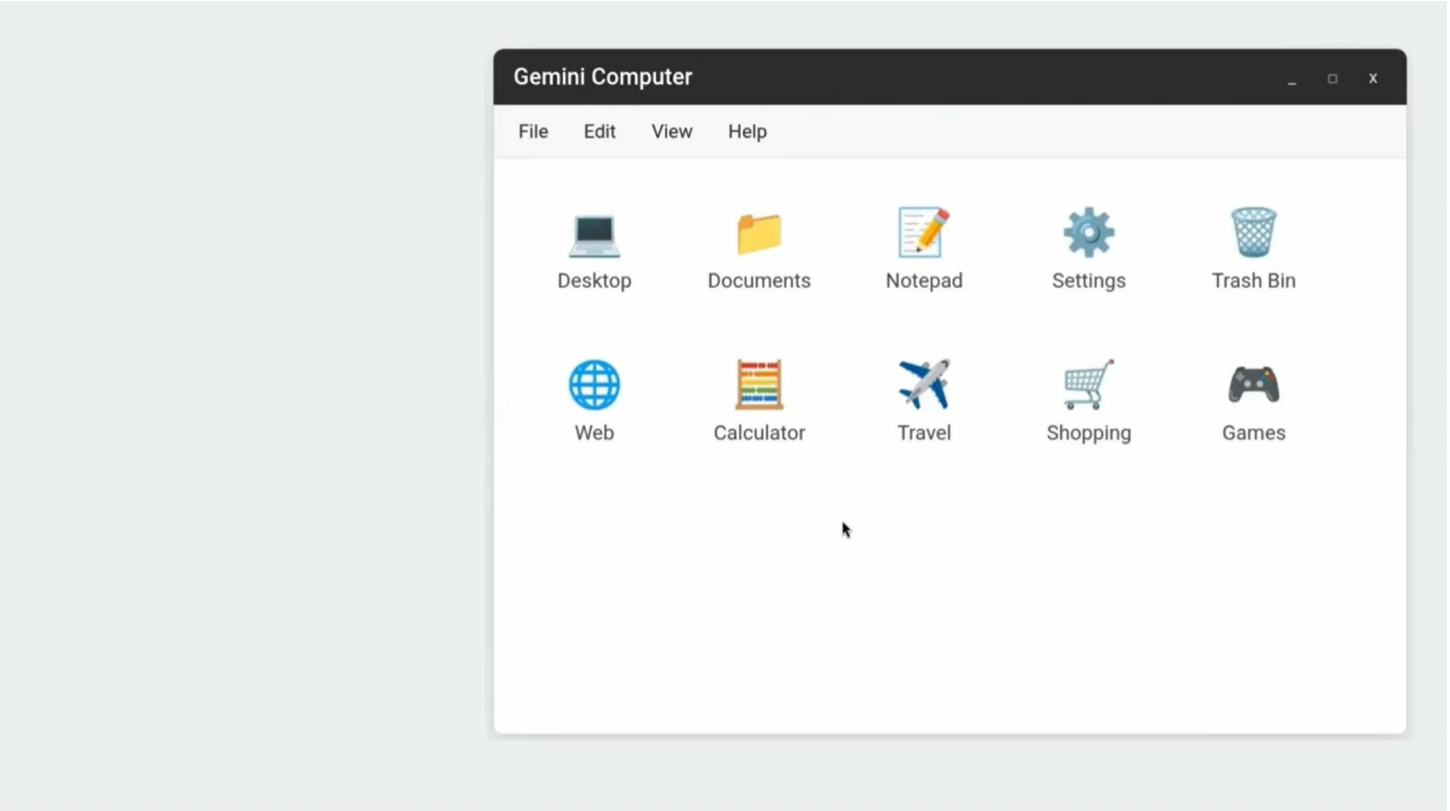
How can browsers adapt to the development of the AI era, cope with the shift of information and interaction, and embrace the user interface based on natural language?

Two Technologies for Agent App Development



Generative UI for Web

Dynamic Generative UI Technology

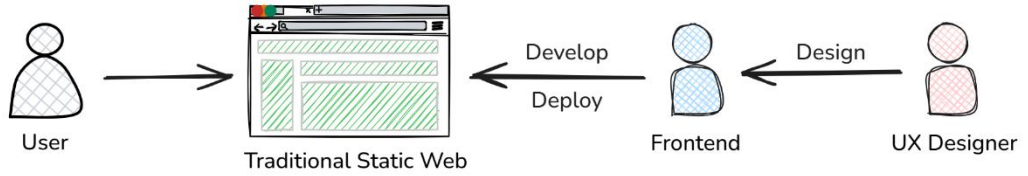


Generative AI » Generative UI

Gemini 2.5 Flash-Lite outputs UI code based only on the context of the previous screen.

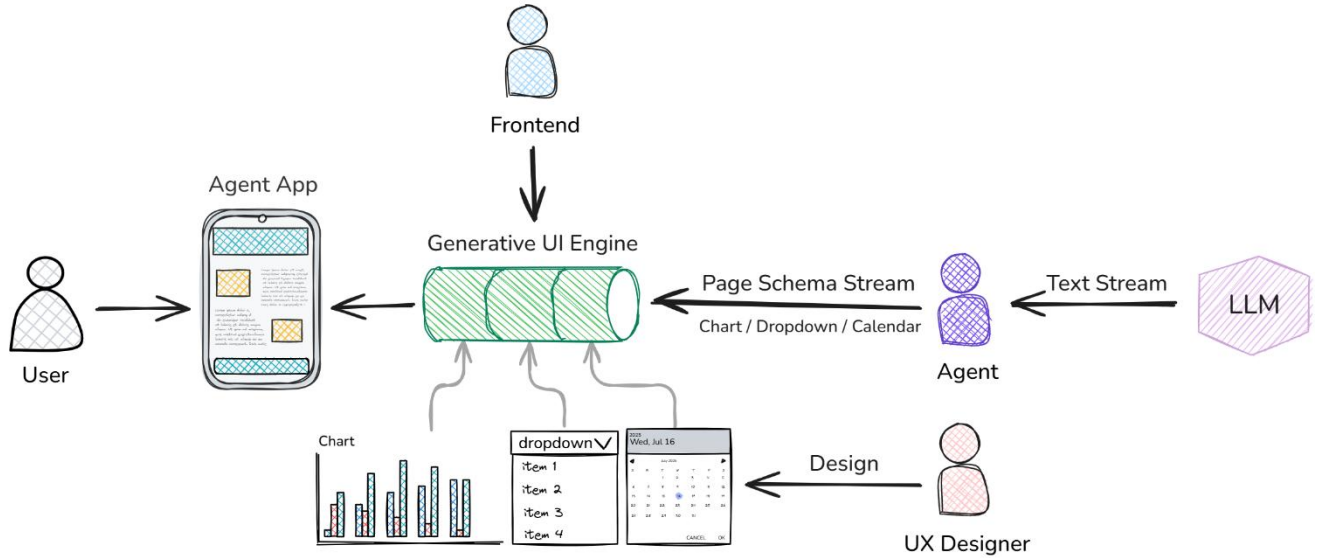
Differences from Static Web App Development

Traditional Static Web App



After a web page is designed, developed, and deployed, the interactive interface seen by the user does not change.

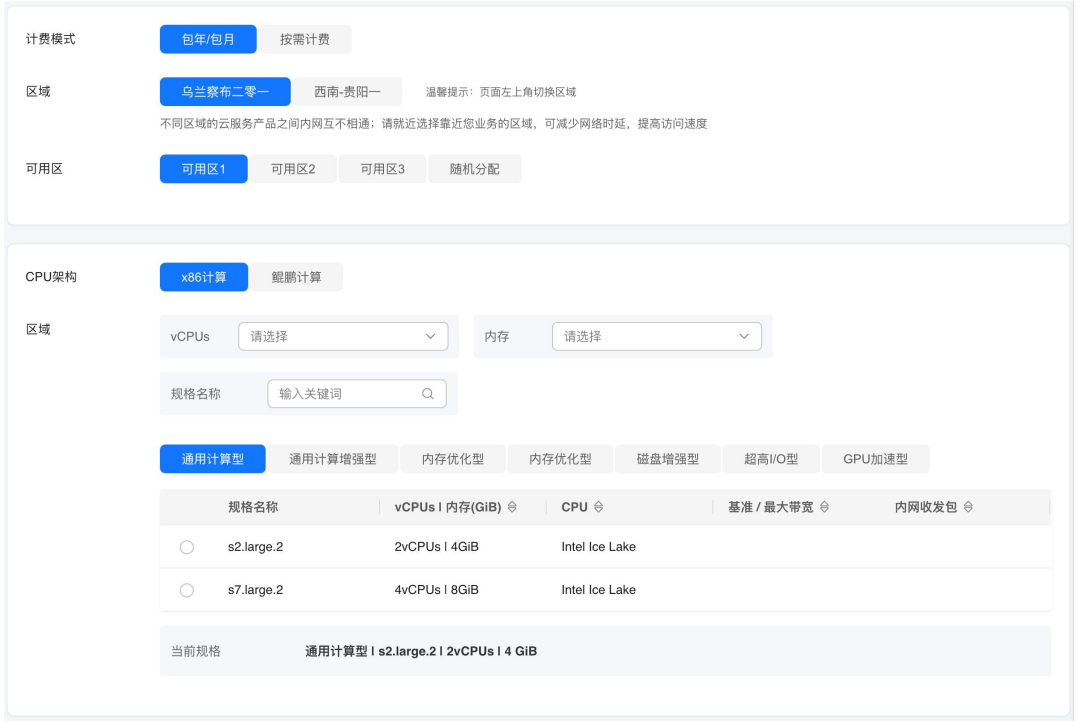
Dynamic Generative Agent App



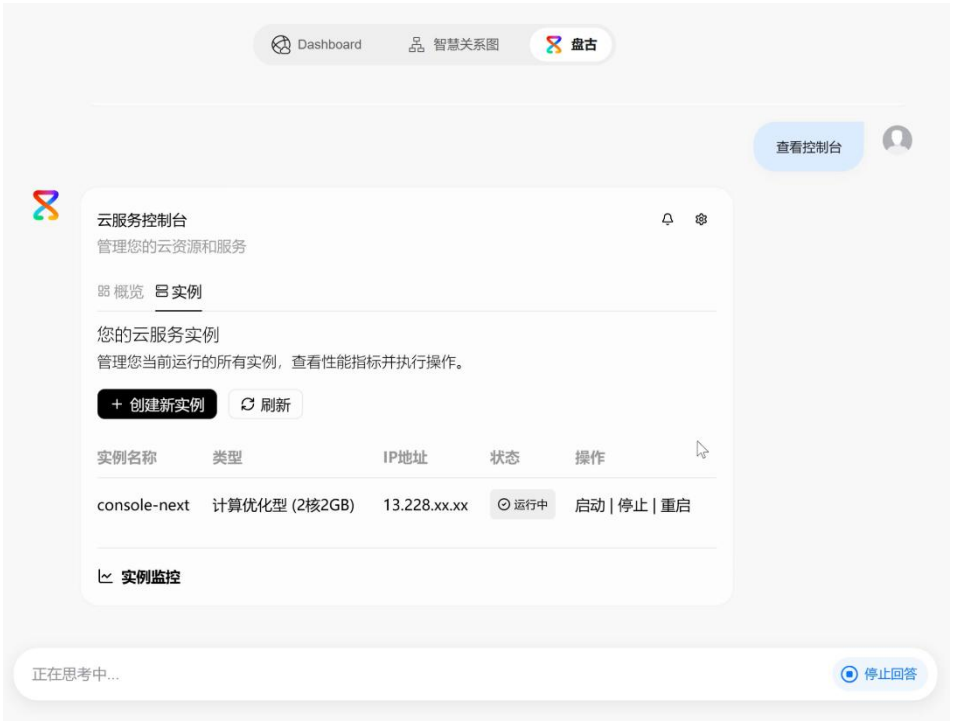
The text flow is converted into a page description schema flow, and the interactive interface seen by the user is generated dynamically in real-time.

Generative UI Deconstruct & Reconstruct Static Web Page

Complex Traditional Web Apps



Dynamic Generative AI Apps



Dynamic generative AI apps completely deconstruct & reconstruct the UI of complex traditional web apps.

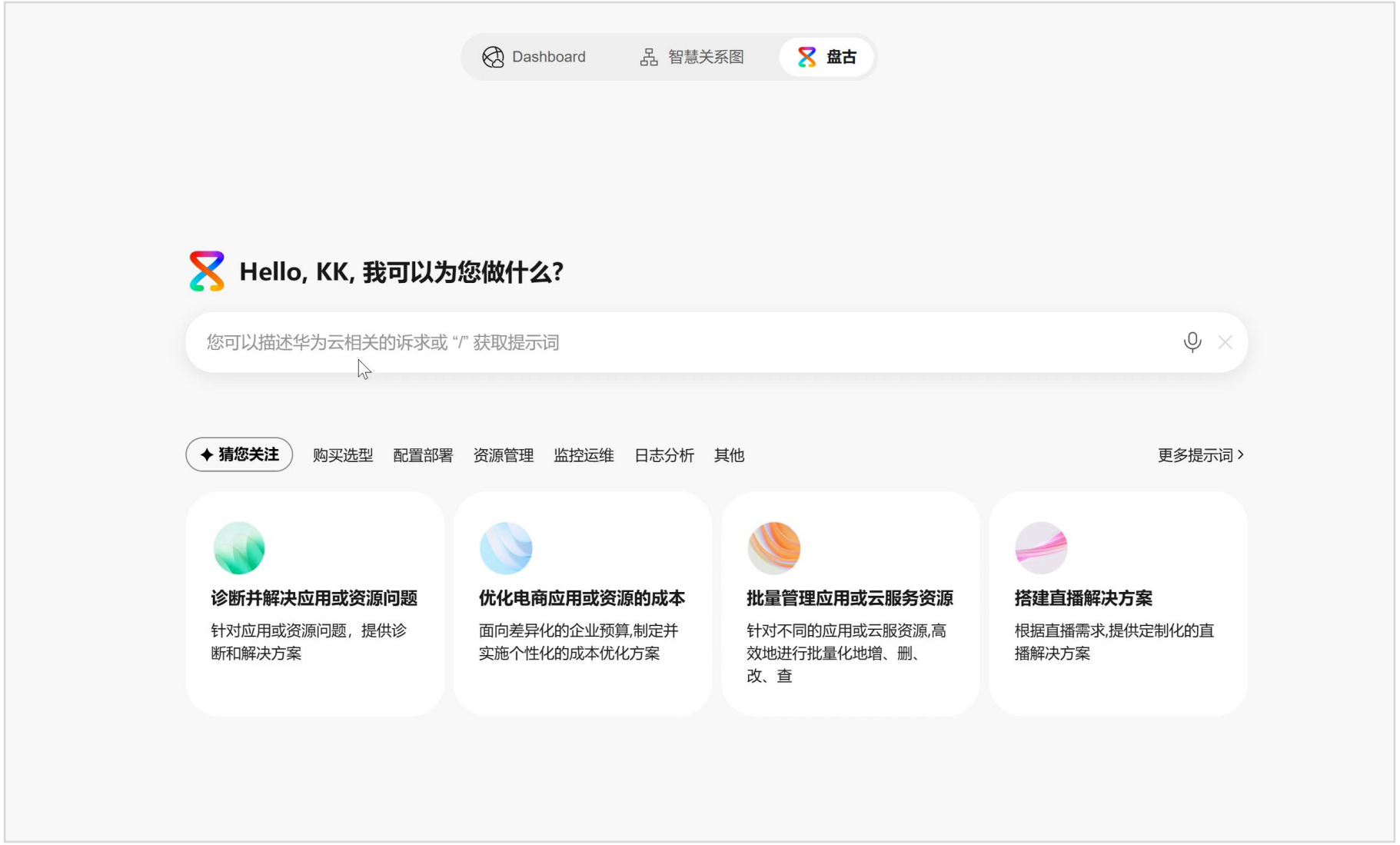
AI App Built Using Generative UI

Generative UI

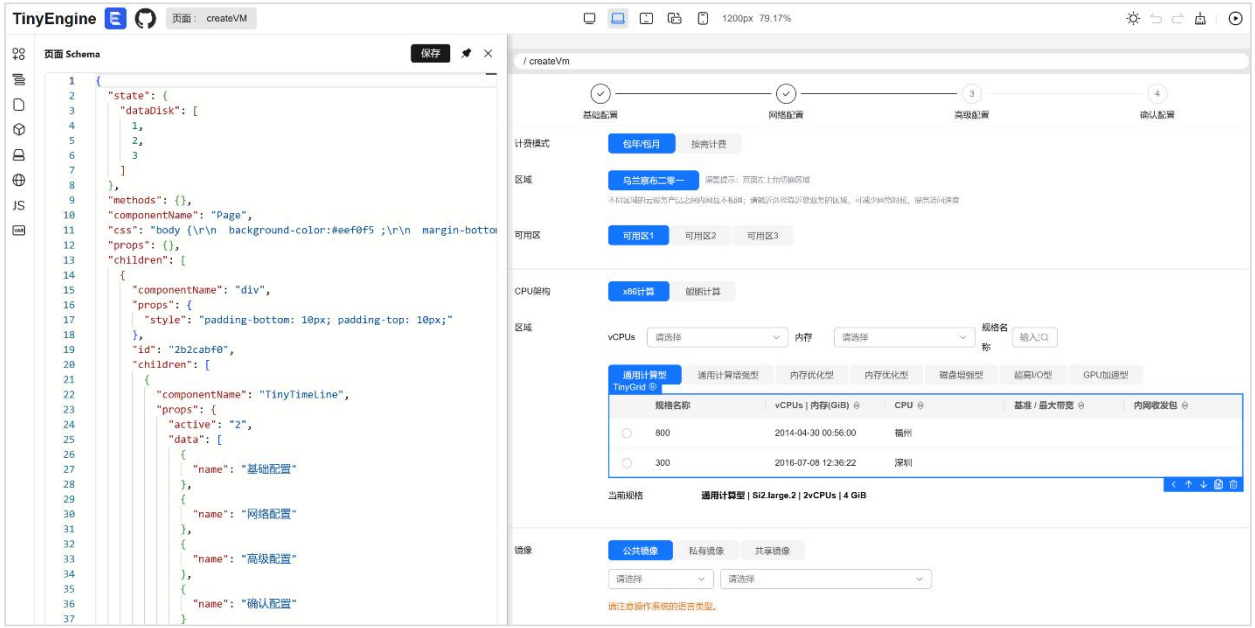
The generated interface is an embodiment of the AI's will, a user interface dynamically created on demand. It cannot be precisely designed; it can only be trained.

Web Standards Subset

The UI for AI conversations belongs to a lightweight rendering category. Dynamic generation places higher demands on web performance, and the components and styles required for Generative UI are limited.



Generative UI is Based on "Page Schema + streamObject"




```
{
  "state": {
    "dataDisk": [
      1,
      2,
      3
    ]
  },
  "methods": {},
  "componentName": "Page",
  "css": "body {\r\n  background-color:#eef0f5 ;\r\n  margin-bottom
  "props": {},
  "children": [
    {
      "componentName": "div",
      "props": {
        "style": "padding-bottom: 10px; padding-top: 10px;"
      },
      "id": "2b2cabf0",
      "children": [
        {
          "componentName": "TinyTimeLine",
          "props": {
            "active": "2",
            "data": [
              {
                "name": "基础配置"
              },
              {
                "name": "网络配置"
              },
              {
                "name": "高级配置"
              },
              {
                "name": "确认配置"
              }
```

1. Low-code Page Schema

```ts
interface IComponentSchema {
  componentName?: string;
  id: string;
  props?: {
      condition?: boolean | IBindProps;
      ref?: {
          name: string;
          type: string;
      };
      style?: string;
      className?: string;
      [prop:string]?: IEventProps | IBindProps | any;
  };
  children?: Array< IComponentSchema > | string;
}
```

```ts
export const nodeSchema: z.ZodType<Node> = z.object({
  id: z.string(),
  componentName: z.string(),
  props: z.record(z.any(), z.any()),
  children: z.array(z.any()).optional(),
  componentType: z.enum(['Block', 'PageStart', 'PageSection']).optional(),
  slot: z.union([z.string(), z.record(z.any(), z.any())]).optional(),
  params: z.array(z.string()).optional(),
  loop: z.record(z.any(), z.any()).optional(),
  loopArgs: z.array(z.string()).optional(),
  condition: z.union([z.boolean(), z.record(z.any(), z.any())]).optional(),
});
```

2. Convert to streamObject Schema

AI SDK Core > streamObject

`streamObject()`

Streams a typed, structured object for a given prompt and schema using a language model.

It can be used to force the language model to return structured data, e.g. for information extraction, synthetic data generation, or classification tasks.

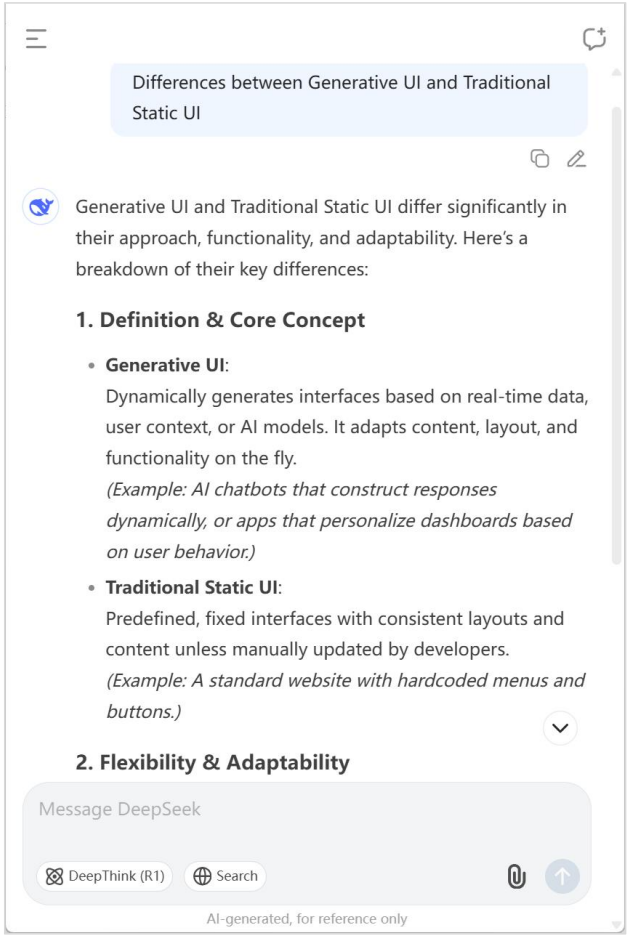
Example: stream an object using a schema

```ts
import { openai } from '@ai-sdk/openai';
import { streamObject } from 'ai';
import { z } from 'zod';

const { partialObjectStream } = streamObject({
  model: openai('gpt-4.1'),
  schema: z.object({
    recipe: z.object({
      name: z.string(),
      ingredients: z.array(z.string()),
      steps: z.array(z.string()),
    }),
  }),
  prompt: 'Generate a lasagna recipe.',
});

for await (const partialObject of partialObjectStream) {
  console.clear();
  console.log(partialObject);
}
```

3. Call the AI SDK's streamObject Method

Suggestions for Web Standards Subset for Generative UI

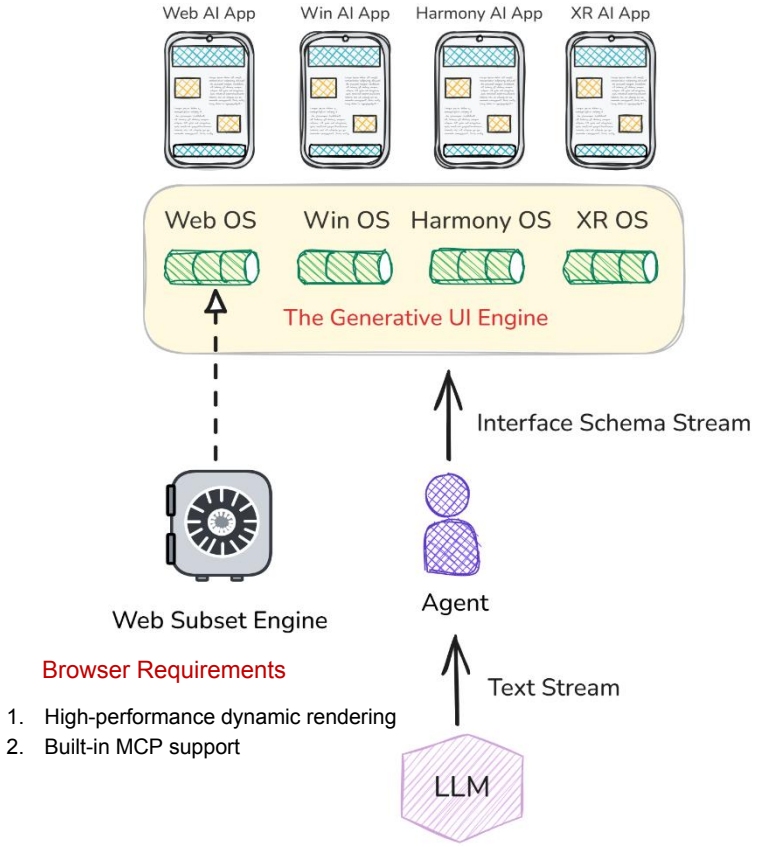
Markdown has become the de facto standard for generated text



The Interface Description Schema should be a UI protocol that is independent of any specific technology stack

```typescript
interface IPageSchema {
 fileName?: string;
 componentName?: string;
 meta: {
   id: number;
   creator: string;
   description: string;
   group: string;
   isHome: boolean;
   parentId: string;
   rootElement: string;
   router: string;
 }
 css?: string;
 props?: {
     [prop:string]?: any;
     style?: string;
     className?: string;
 };
 children?: Array< IComponentSchema > | string;
}

interface IComponentSchema {
  componentName?: string;
  id: string;
  props?: {
      condition?: boolean | IBindProps;
      ref?: {
          name: string;
          type: string;
      };
      style?: string;
      className?: string;
      [prop:string]?: IEventProps | IBindProps | any;
  };
  children?: Array< IComponentSchema > | string;
}
```

The Generative UI Engine must be adaptable to various OS for native-like experiences

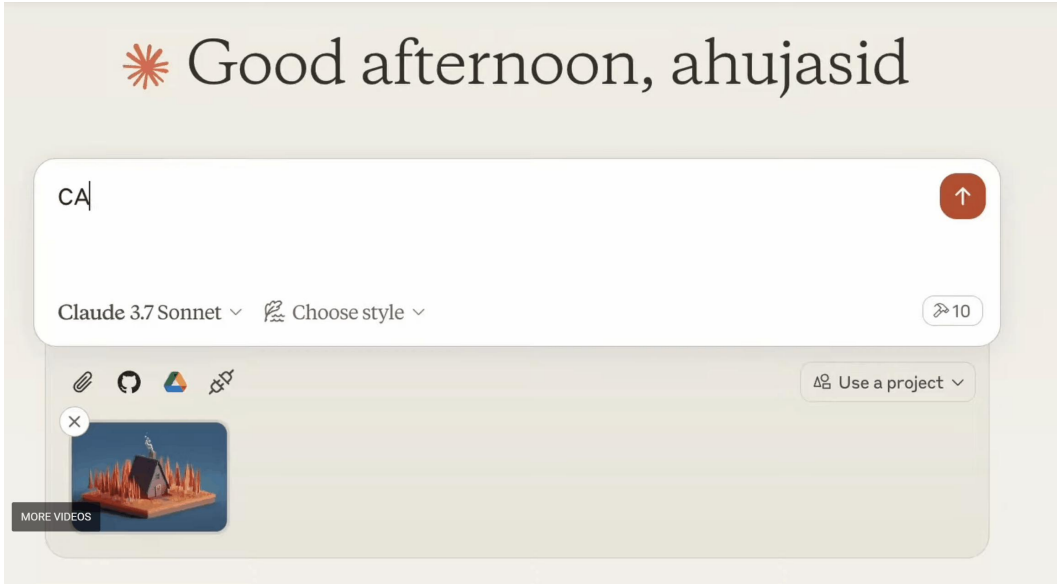


Browser Requirements

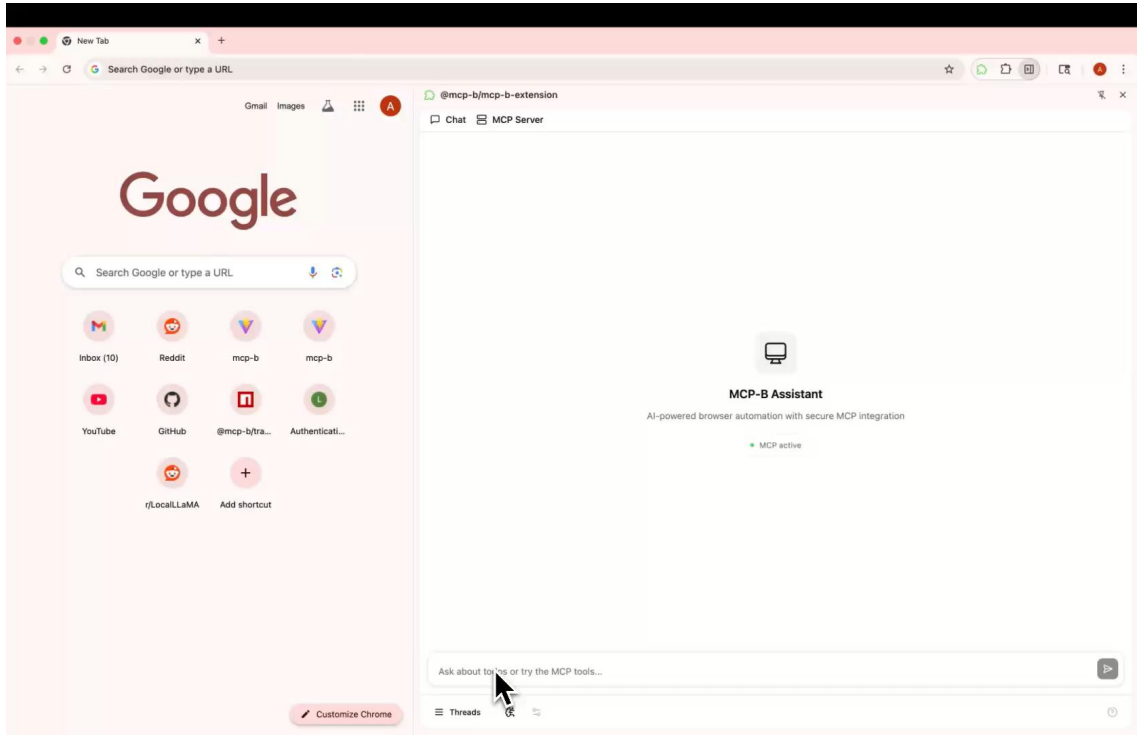
1. High-performance dynamic rendering
2. Built-in MCP support

MCP Servers for Web

MCP Servers From Native Apps to Web Apps



Blender MCP allows an LLM to directly control
the Blender modeling software via the MCP protocol



MCP-B allows a local LLM to control web apps
through the MCP protocol and a browser extension

MCP-B —— Browser-Native MCP Servers

Industry First

Why MCP-B Changes Everything

While other solutions take 10-20 seconds per action and cost $4-5 per task, MCP-B delivers instant results with zero configuration. The future of browser automation is here.

The Breakthrough

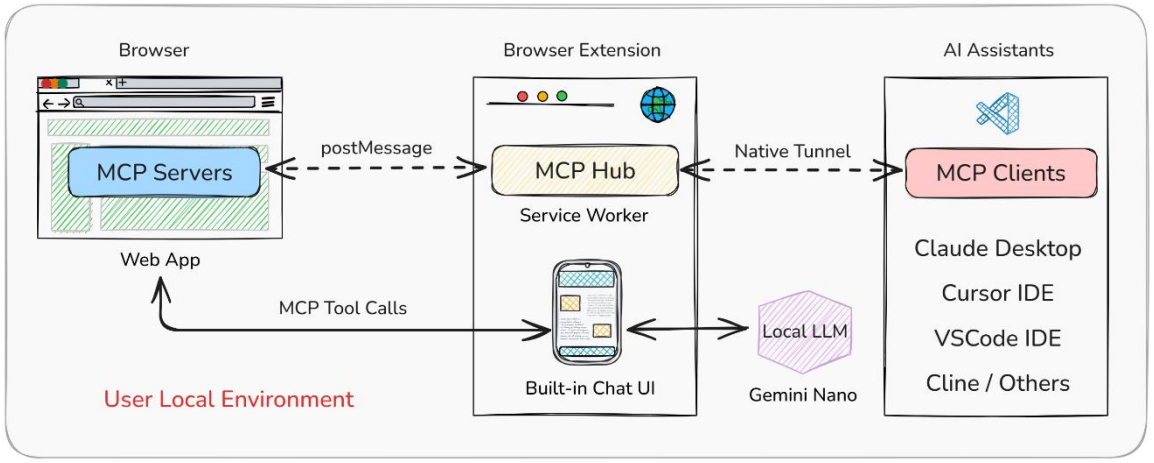
Browser-Native MCP Servers

Instead of running MCP servers as separate processes or cloud services, we embed them directly into web pages. The MCP server becomes part of your web application.

The Challenge
- Remote MCPs need complex OAuth 2.1
- Local MCPs require API keys everywhere
- White-collar work happens in browsers

Our Solution
- ✓ Run MCP servers inside web pages
- ✓ Use existing browser authentication
- ✓ Bridge to any MCP client via extension

The Result
- → Authentication just works
- → No API keys or OAuth flows
- → Works with any website

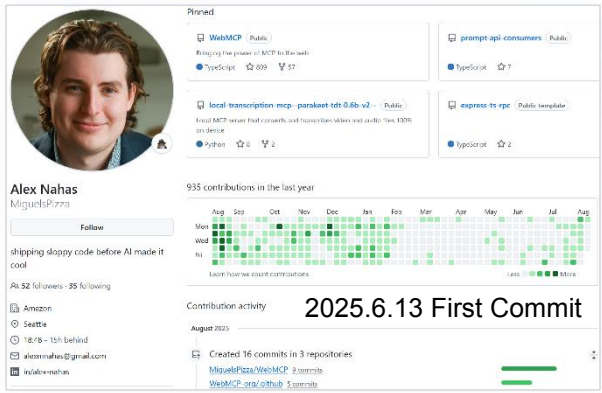


10,000x Performance Improvement

Traditional browser automation performance:
- 10-20 seconds per task
- $4-5 in API costs per simple action
- Multiple model calls for UI parsing
- Brittle, unreliable execution

MCP-B executes the same tasks in milliseconds with direct API calls.

https://github.com/MiguelsPizza/WebMCP/



How MCP-B Works: From Web Page to AI Assistant

1. Tab MCP Servers

Your Web Apps

Tab MCP Server
- TypeScript, in-memory transport
- Wraps your authenticated APIs
- Uses existing cookies/JWT

→ fetch/XHR →

Your Existing APIs
- No changes needed
- Same auth as UI

2. MCP-B Extension

Chrome Extension

Content Scripts
Connect to tab servers via postMessage

MCP Hub (Service Worker)
- Aggregates all tab tools
- Routes tool calls
- Manages connections

Built-in Chat UI
Side panel AI assistant

3. MCP Clients

AI Assistants

Native Bridge
- Native messaging tunnel
- Proxy server option

Claude Desktop

Cursor IDE

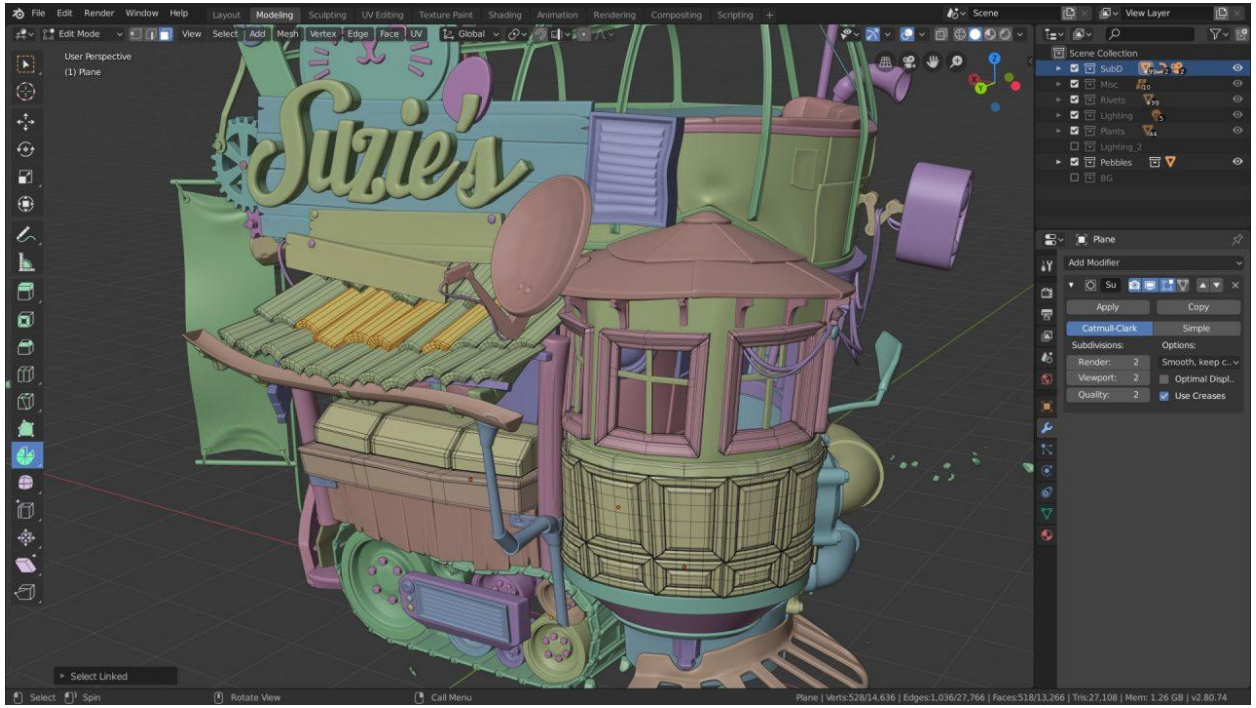
Cline / Others

Complete flow: AI requests tool → Extension routes to tab → Tab MCP executes using your auth → Results flow back to AI

MCP servers run in web pages, extension bridges to AI clients, transport layers handle the plumbing

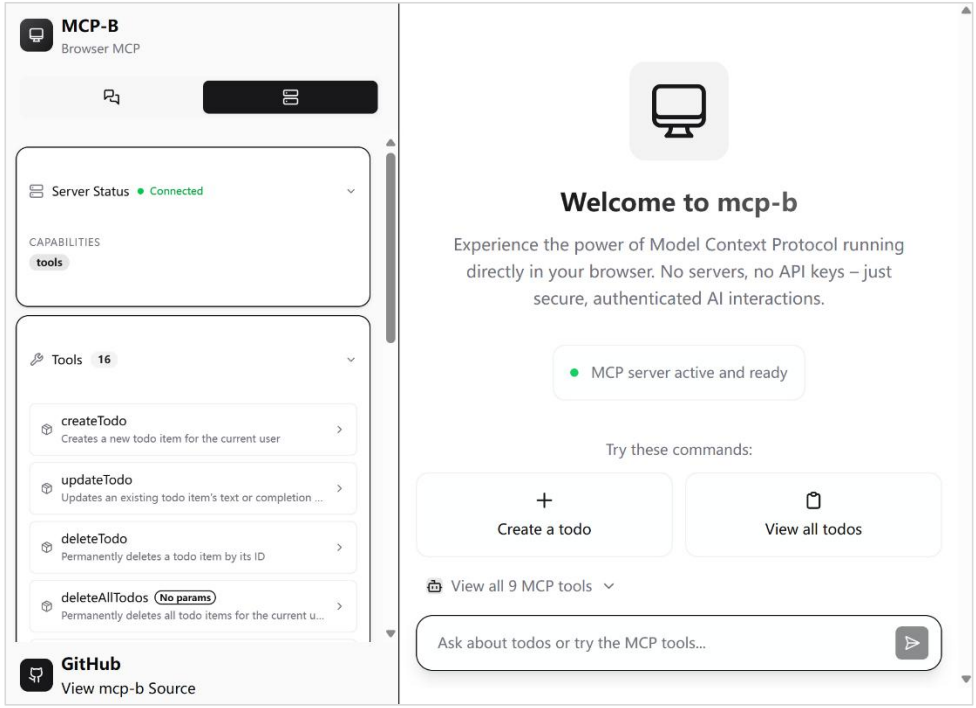
Scenarios for Web App MCP Servers

Is it suitable for web applications with complex interactions similar to Blender?



For websites that are primarily for browsing, or for web applications with simple interactions?

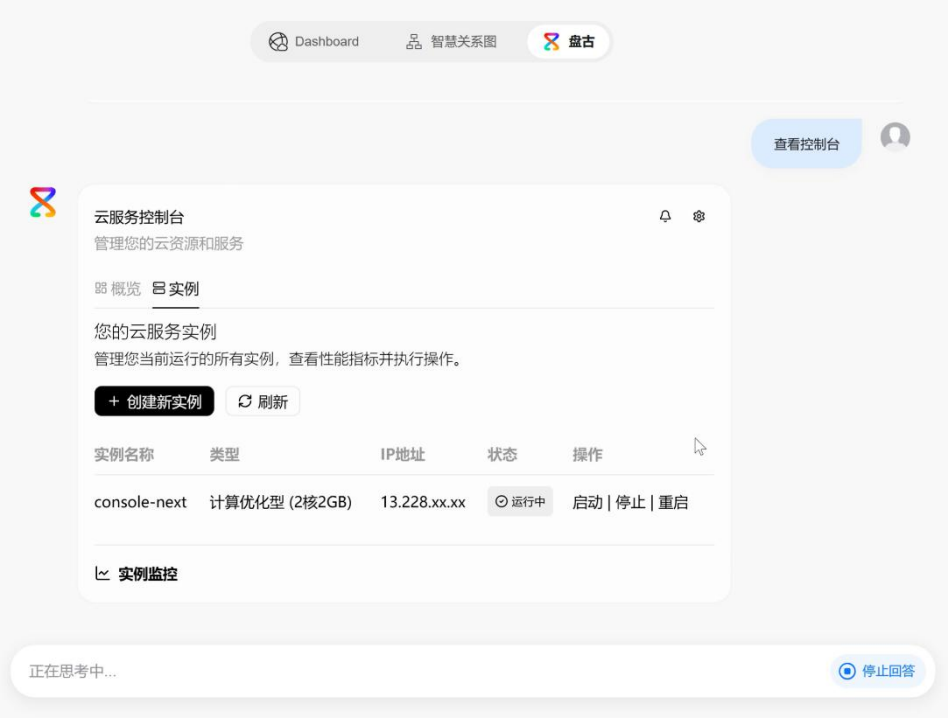
Integrating with a local LLM to protect user privacy?



With a browser extension acting as the bridge
Can a remote MCP Host directly connect to a user's local computer?

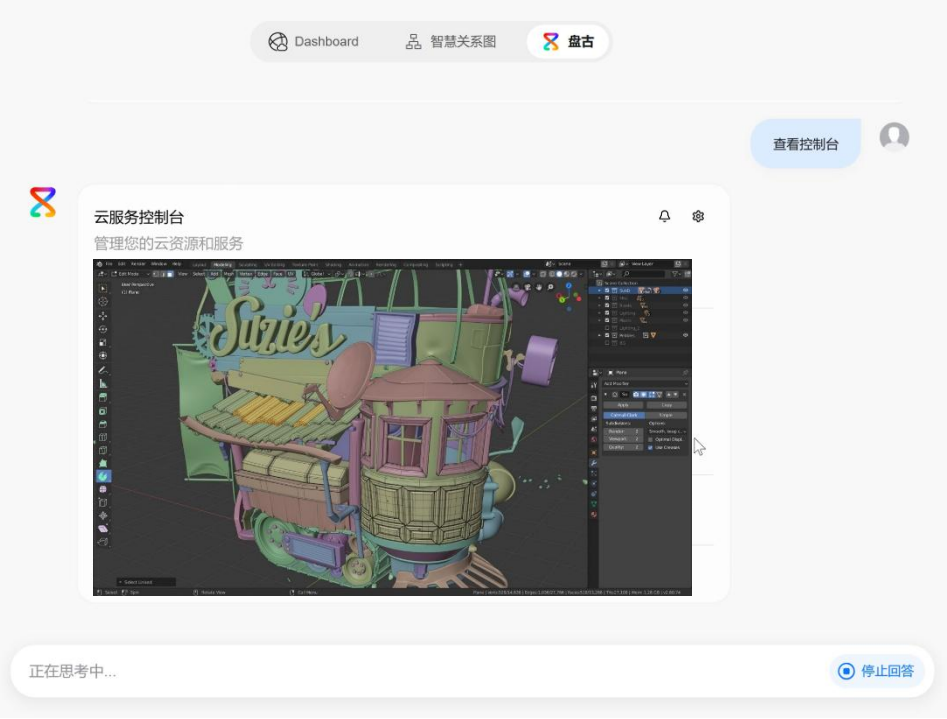
Web Apps that Replaced with Generative UI

Generative UI can render interfaces that are primarily for browsing and simple interactions



The prerequisite: the web application's backend provides the AI application with APIs to read and manipulate data.

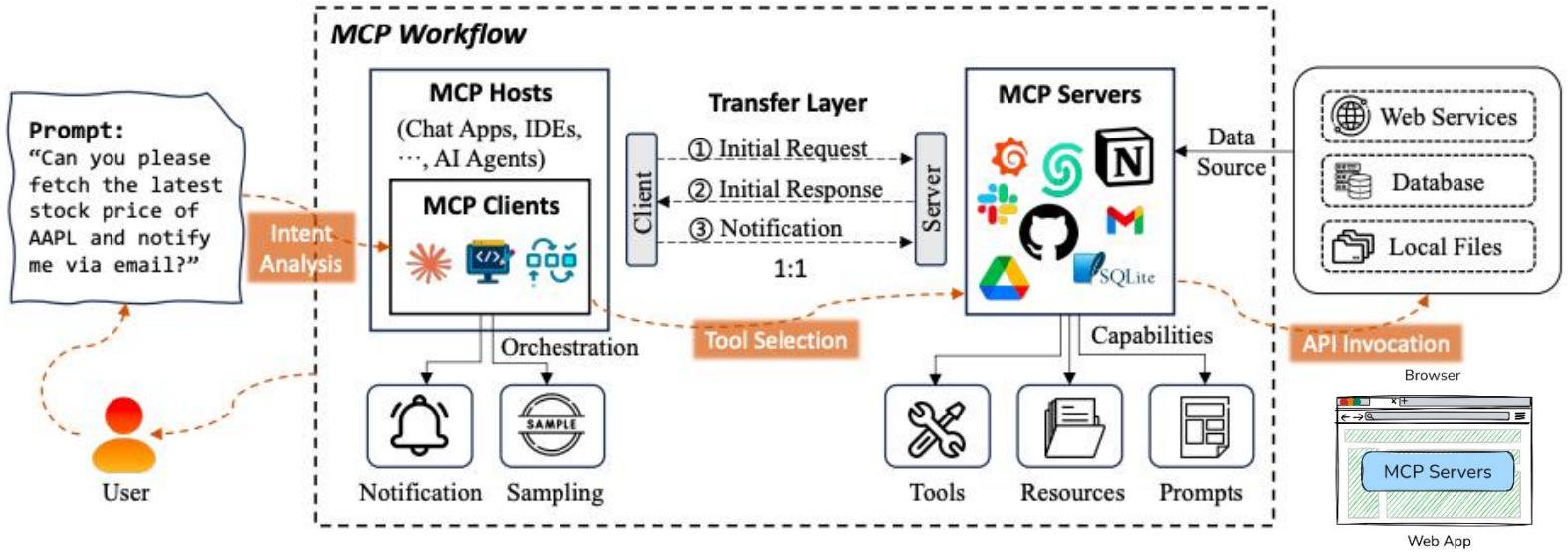
Generative UI has difficulty rendering interfaces with complex interactions, similar to those in Blender



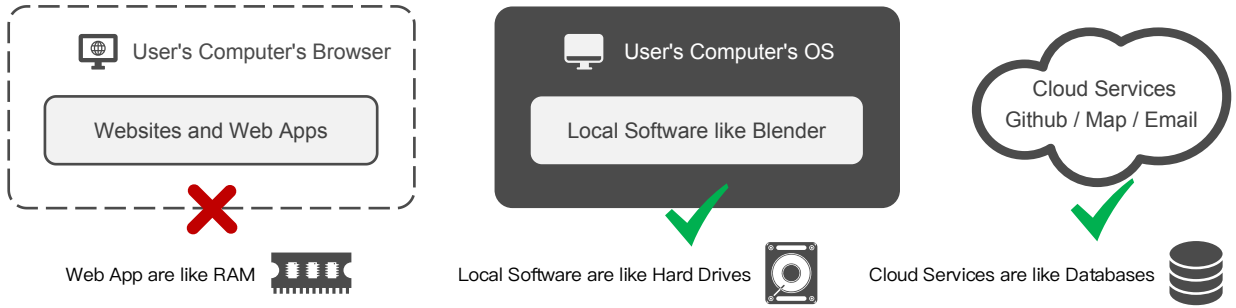
The prerequisite: AI applications shift to a 3D interaction, such as XR (Extended Reality) plus gestures

MCP Host Directly Connects to Web Apps

The operational process of an AI application using the MCP protocol

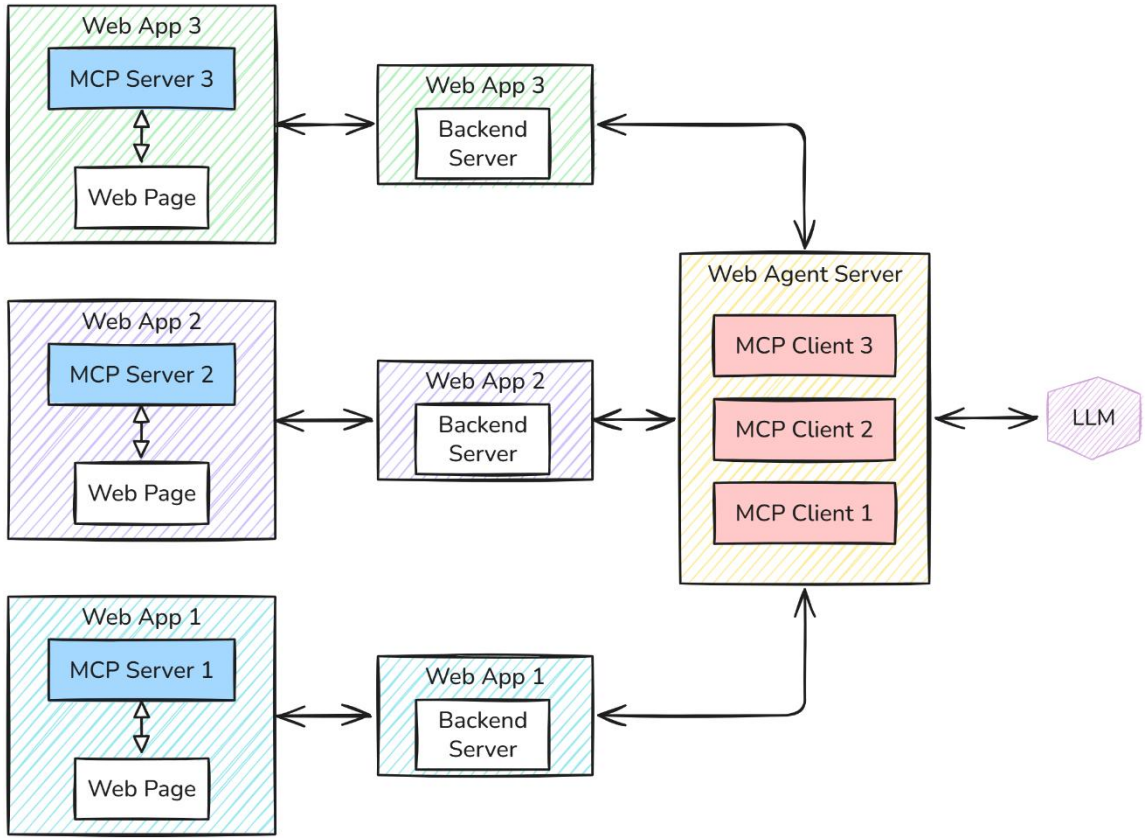


Currently, the MCP server only supports local software and cloud services

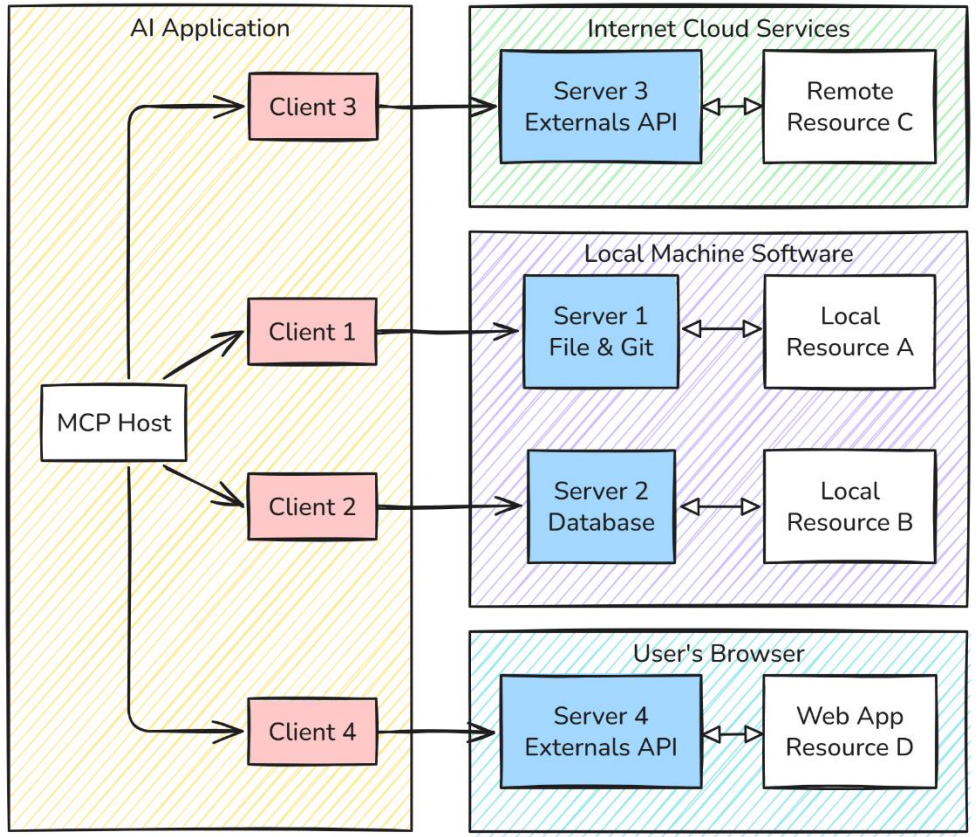


Our Solution: Web Apps Access to Web Agent Server

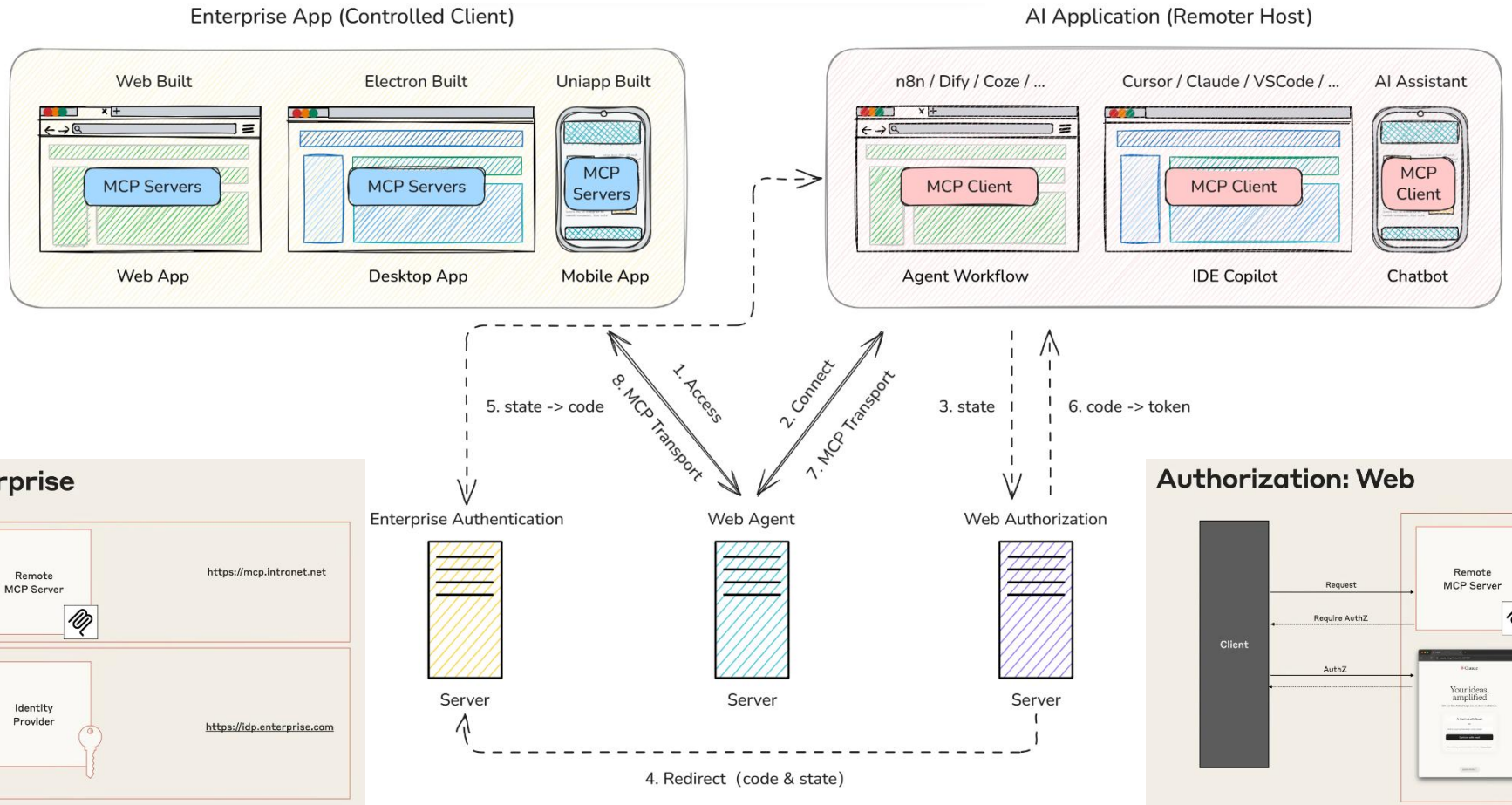
Any web application running in a browser can be connected to a Web Agent Server



Web app becomes an MCP Service option that an MCP Host can call.



Web Agent Server uses OAuth 2.1 authentication



Enterprise Authentication Mechanism for MCP-enabled Web

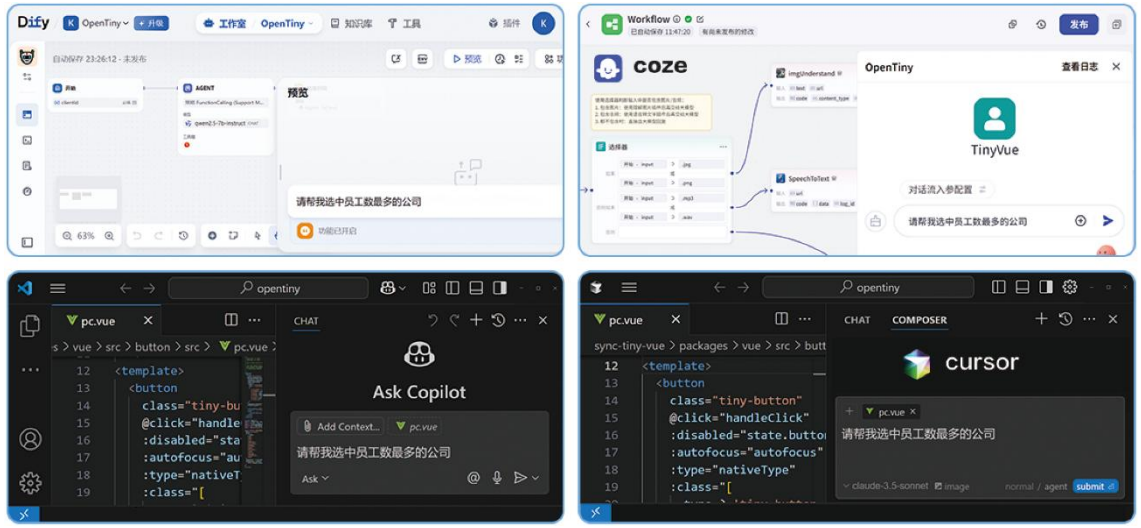
Third-party AI applications must undergo enterprise authentication to control enterprise web applications

Client Authorization Mechanism for MCP-enabled Web

A Practical Demo of the Web App MCP Servers

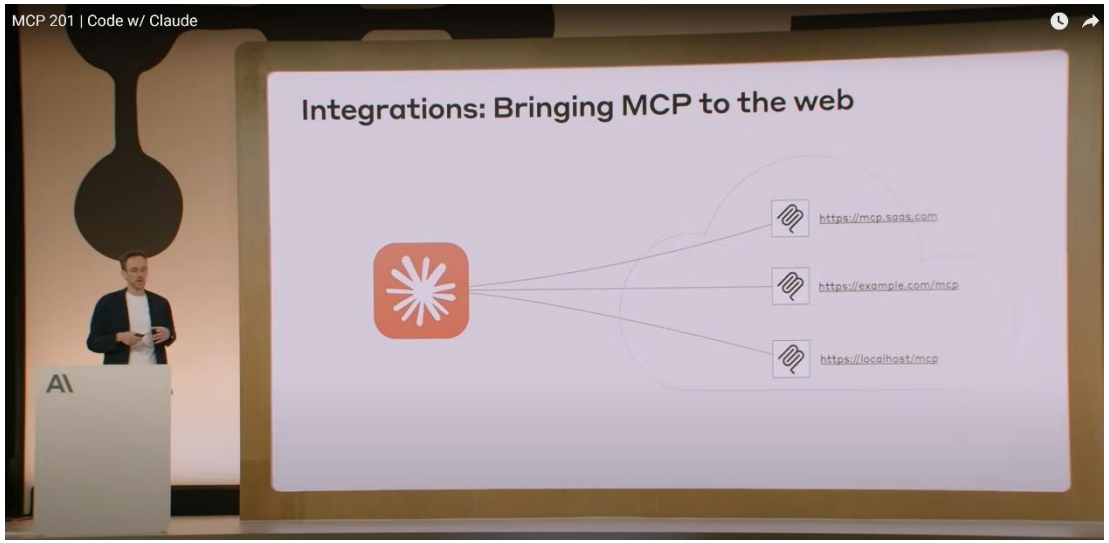
Supports AI assistants in remotely controlling web apps

Supports various Copilot and Agent development platforms in adding MCP tools to control web apps.



Standards Suggestions for Web Application MCP Servers

On May 22, 2025, David Soria Parra, co-creator of MCP, shared his insights at "Code w/ Claude."



David's point Currently, over 10,000 MCP servers run in local environments, but we believe the future of MCP is on the web. Web-enabled MCP servers mean that an MCP service will no longer be a local Docker container but rather a website. Clients will directly connect and interact with the MCP interfaces exposed by the website.

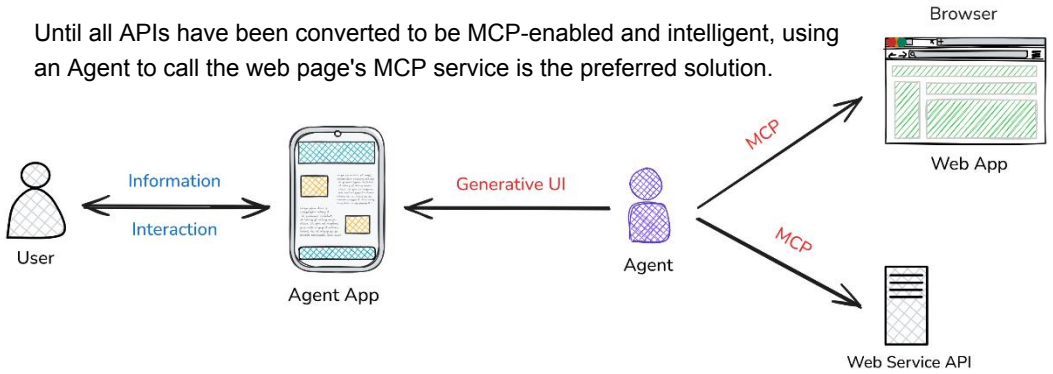
> Web apps with built-in MCP servers are a way for browsers to merge human-computer interaction with human-agent interaction.

> An Agent operates on a web page on behalf of a human. The actions are predefined, making the process more secure and transparent.

> The cost of converting a web page to an MCP-enabled one is low, which can encourage a vast number of websites to become tools for AI agents

MCP services for the web can transition from the frontend to the backend

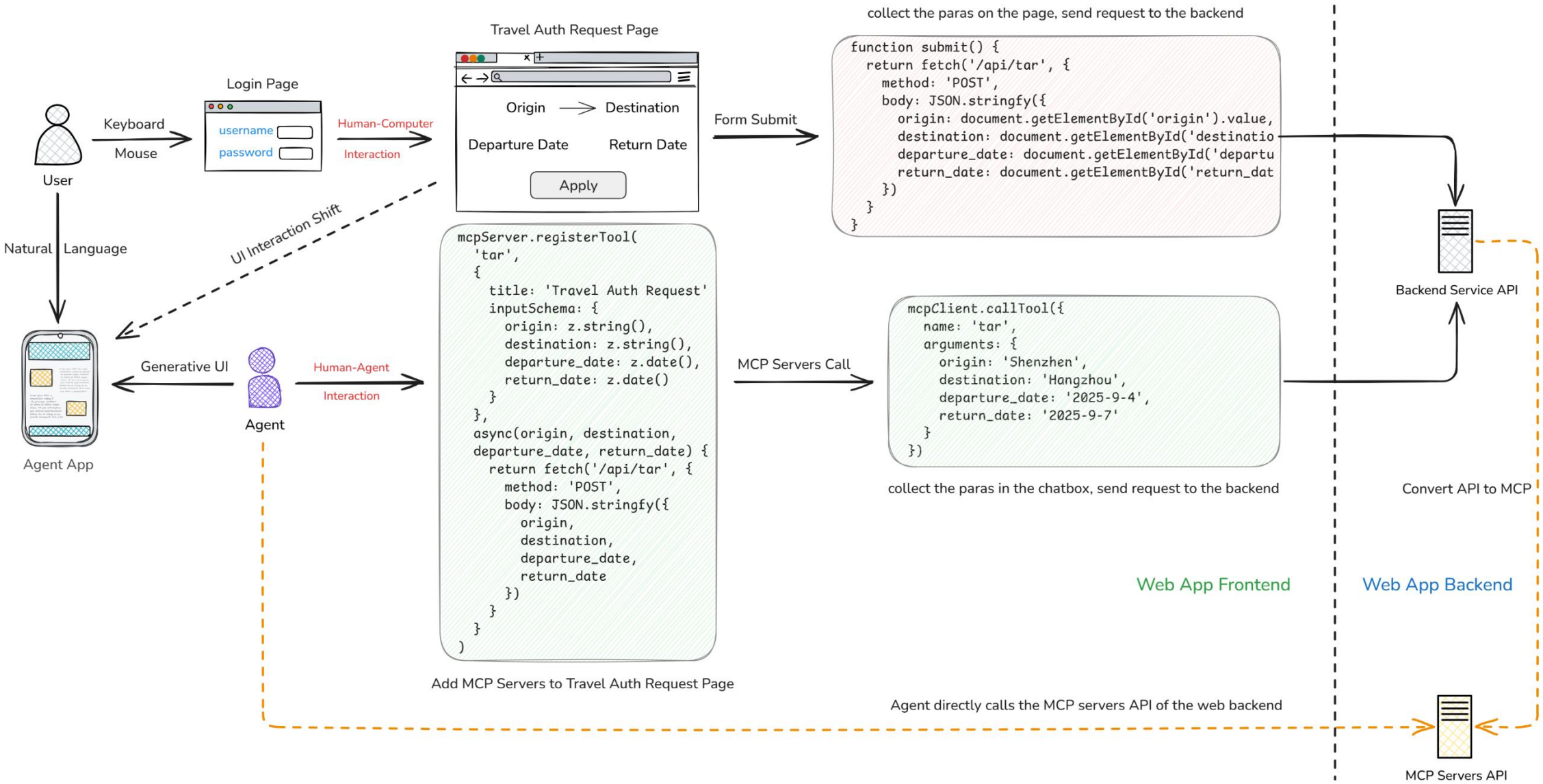
Until all APIs have been converted to be MCP-enabled and intelligent, using an Agent to call the web page's MCP service is the preferred solution.



Web Standards Suggestion: Embed the MCP SDK into the window object

```
const server = new window.MCP.Server({ name: "demo-server", version: "1.0.0" });

server.registerTool("add",
  {
    title: "Addition Tool",
    description: "Add two numbers",
    inputSchema: { a: z.number(), b: z.number() }
  },
  async ({ a, b }) => ({
    content: [{ type: "text", text: String(a + b) }]
  })
);

await server.connect(new window.MCP.ServerTransport());

const client = new window.MCP.Client({ name: "example-client", version: "1.0.0" });

await client.connect(new window.MCP.ClientTransport());

const result = await client.callTool({ name: "add", arguments: { a: "1", b: "2" }});
```

Integrate Human-Computer & Human-Agent Interaction Paradigms



WebMCP Proposal

WebML CG Teleconference – 18 August 2025

WebML CG Teleconference [EU-Asia-Pacific] – 18 August 2025

Agenda

2025-08-18-cg-agenda.md

Minutes

Participants

Anssi_Kostiainen, Tarek_Ziade, Brandon_Walderman, Adam_Sobienski, Zoltan_Kis

[Anssi acknowledges some participants are on vacation so feedback via GH is preferred especially for Built-in AI APIs topics]

TPAC 2025 F2F announced (Kobe, Japan, 10–14 Nov 2025)

Anssi: TPAC is the W3C's annual all-groups meeting, for both Working Groups and Community Groups

Agentic web & WebMCP

Anssi: a new agentic web workstream launched and the its first exploration is the WebMCP proposal

... WebMCP in abstract: "Enabling web apps to provide JavaScript-based tools that can be accessed by AI agents and assistive technologies to create collaborative, human-in-the-loop workflows."

-> WebMCP repo https://github.com/webmachinelearning/webmcp

-> WebMCP explainer https://github.com/webmachinelearning/webmcp/blob/main/docs/explainer.md

-> WebMCP proposal https://github.com/webmachinelearning/webmcp/blob/main/docs/proposal.md

Anssi: the scope of work in this initial exploration phase is an explainer document and a high-level proposal that explores possible API shape without going into spec details

... the initial explainer and proposal is based on contributions by Microsoft ("Web Model Context") and Google ("Script Tools") and we've now completed the first part of merging the two proposals

... also open-source projects have informed the initial design and I want to acknowledge the following two projects specifically:

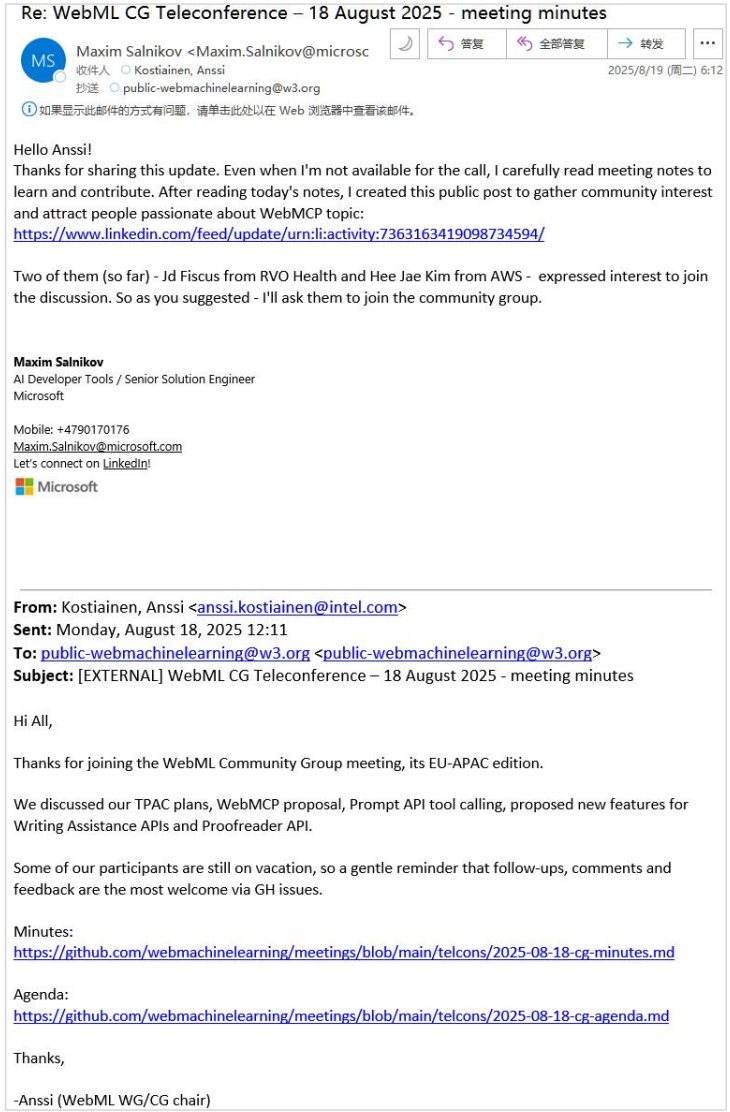
-> https://github.com/MiguelsPizza/WebMCP

-> https://github.com/jasonjmcghee/WebMCP

... I will initiate the Community Group charter change process soon to allow the group advance to the more formal specification drafting stage

... if you're aware of folks interested in contributing to this agentic web development, please extend my invite to them to join the Community Group:

-> https://webmachinelearning.github.io/community/#join



Re: WebML CG Teleconference – 18 August 2025 - meeting minutes

Maxim Salnikov <Maxim.Salnikov@microsc
收件人 Kostiainen, Anssi
抄送 public-webmachinelearning@w3.org
2025/8/19 (周二) 6:12

如果显示此邮件的方式有问题，请单击此处以在 Web 浏览器中查看该邮件。

Hello Anssi!
Thanks for sharing this update. Even when I'm not available for the call, I carefully read meeting notes to learn and contribute. After reading today's notes, I created this public post to gather community interest and attract people passionate about WebMCP topic:
https://www.linkedin.com/feed/update/urn:li:activity:7363163419098734594/

Two of them (so far) - Jd Fiscus from RVO Health and Hee Jae Kim from AWS - expressed interest to join the discussion. So as you suggested - I'll ask them to join the community group.

Maxim Salnikov
AI Developer Tools / Senior Solution Engineer
Microsoft

Mobile: +4790170176
Maxim.Salnikov@microsoft.com
Let's connect on LinkedIn!
Microsoft

From: Kostiainen, Anssi <anssi.kostiainen@intel.com>
Sent: Monday, August 18, 2025 12:11
To: public-webmachinelearning@w3.org <public-webmachinelearning@w3.org>
Subject: [EXTERNAL] WebML CG Teleconference – 18 August 2025 - meeting minutes

Hi All,

Thanks for joining the WebML Community Group meeting, its EU-APAC edition.

We discussed our TPAC plans, WebMCP proposal, Prompt API tool calling, proposed new features for Writing Assistance APIs and Proofreader API.

Some of our participants are still on vacation, so a gentle reminder that follow-ups, comments and feedback are the most welcome via GH issues.

Minutes:
https://github.com/webmachinelearning/meetings/blob/main/telcons/2025-08-18-cg-minutes.md

Agenda:
https://github.com/webmachinelearning/meetings/blob/main/telcons/2025-08-18-cg-agenda.md

Thanks,

-Anssi (WebML WG/CG chair)

Getting Started (for website developers)

To use WebMCP, simply include the script on your page:

```
<script src="webmcp.js"></script>
```

The WebMCP widget will automatically initialize and appear in the bottom right corner of your page.

MCP Features

Tools | Prompts | Resources

Registering Tools

Tools allow the LLM to perform actions on your website. They are registered via calling `registerTool`.

```
// Initialize with custom options
const mcp = new WebMCP({
    color: '#4CAF50',
    position: 'top-right',
    size: '40px',
    padding: '15px'
});

// Register custom tools
mcp.registerTool(
    'weather',
    'Get weather information',
    {
        location: { type: "string" }
    },
    function(args) {
        return {
            content: [{
                type: "text",
                text: `Weather for ${args.location}: Sunny, 22°C`
            }]
        };
    }
);
```

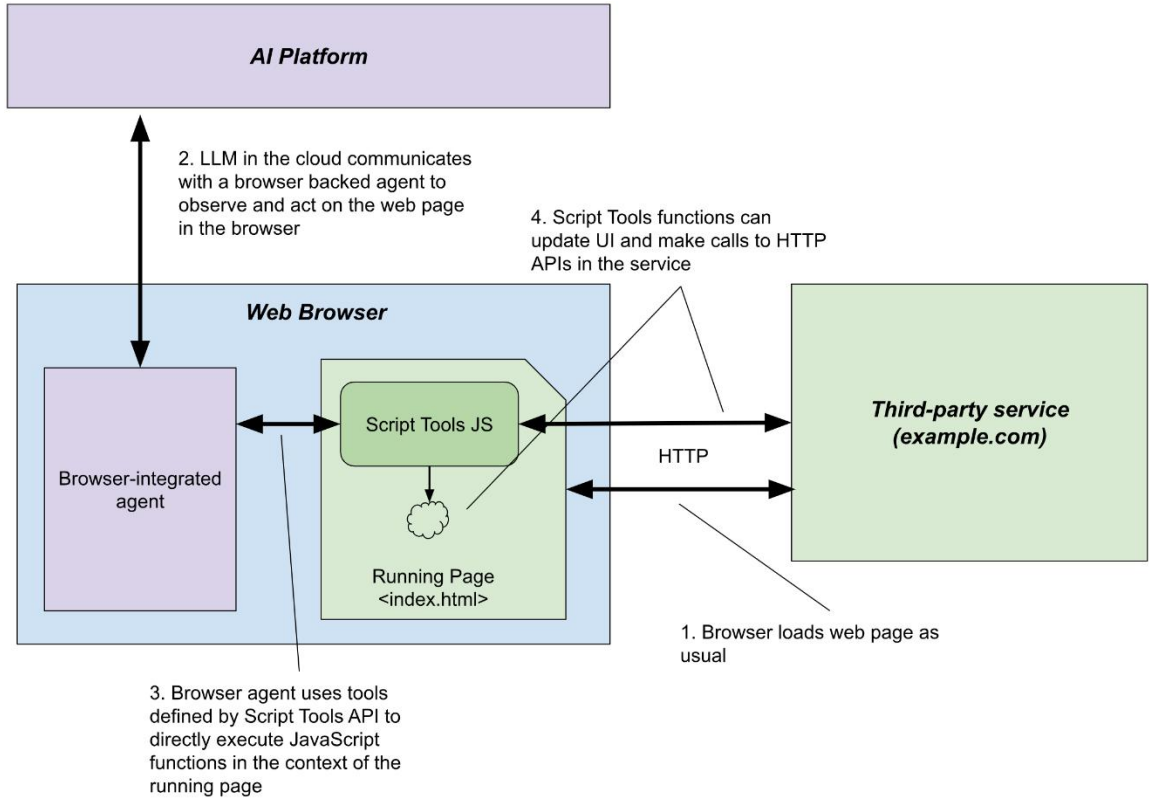
To provide the best experience for users, it is recommended to register all tools directly after loading `<script src="webmcp.js"></script>`, as MCP clients may need to be restarted to get the available tools.

For the sake of demonstration, the below can be clicked to dynamically register a new tool:

Register Weather Tool | Register Time Tool

WebMCP – Web Machine Learning CG proposal

WebMCP is a proposal for a web API that enables web pages to provide agent-specific paths in their UI. With WebMCP, agent-service interaction takes place via app-controlled UI, providing a shared context available to app, agent, and user. In contrast to backend integrations, WebMCP tools are available to an agent only once it has loaded a page and they execute on the client. Page content and actuation remain available to the agent (and the user) but the agent also has access to tools which it can use to achieve its goal more directly.



API The window.agent interface is introduced to represent an abstract AI agent that is connected to the page and uses the page's context. The agent object has a single method provideContext that's used to update the context (currently just tools) available to the agent. The method takes an object with a tools property which is a list of tool descriptors. The tool descriptors look as shown in this example below, which aligns with the Prompt API's tool use specification, and other libraries like the MCP SDK.

```
// Declare tool schema and implementation functions.
window.agent.provideContext({
    tools: [
        {
            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"]
            },
            async execute({ text }) => {
                // Add todo item and update UI.
                return /* structured content response */
            }
        }
    ]
});

window.agent.addEventListener('toolcall', async e => {
    if (e.name === "add-todo") {
        // Add todo item and update UI.
        e.respondWith(/* structured content response */);
        return;
    } // etc...
});
```

https://github.com/webmachinelearning/webmcp/

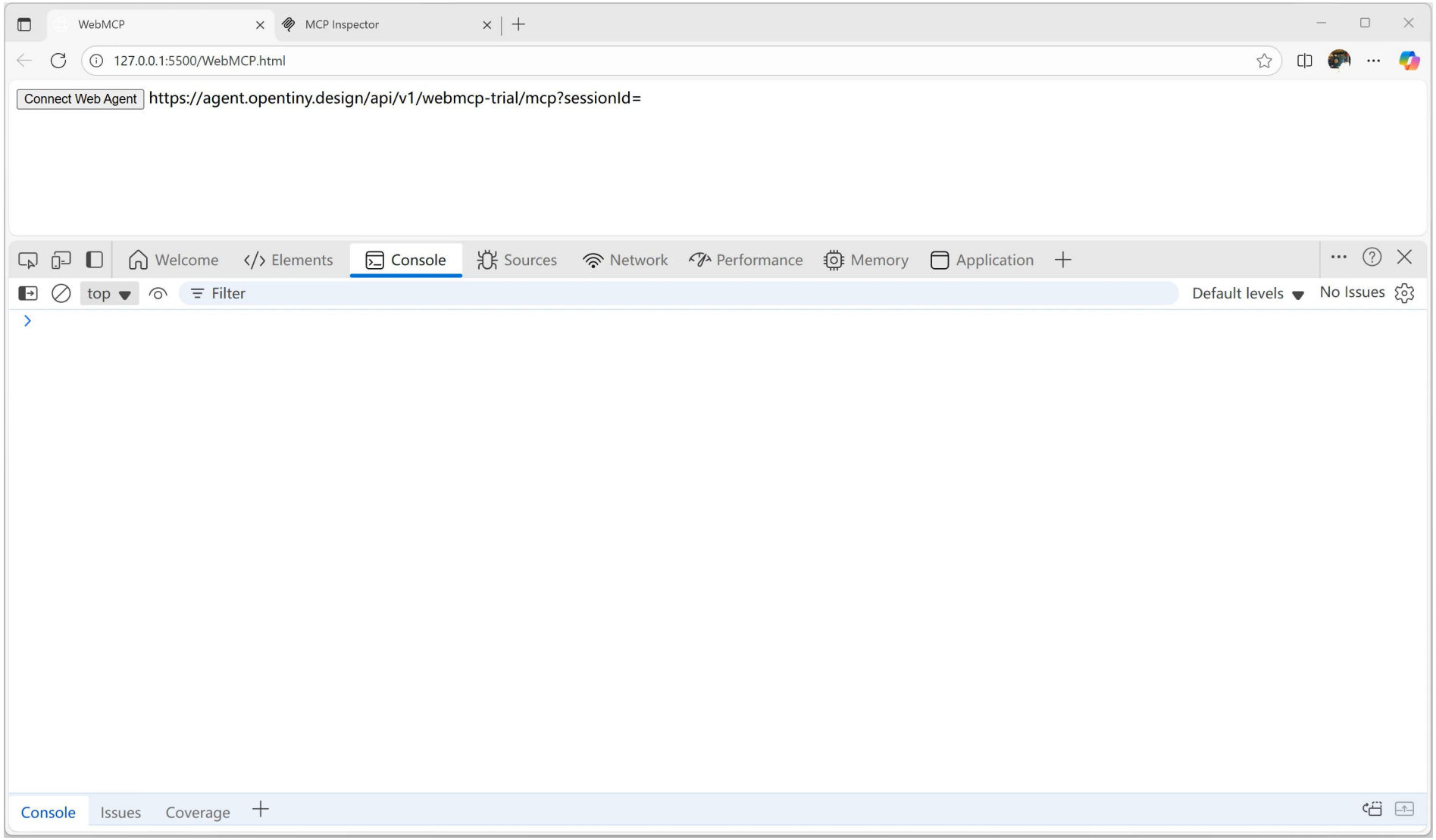
Our WebMCP Solution: SDK Example

Fully compatible with MCP SDK Create an MCP Server and MCP Client using standard MCP SDK methods. It not only supports registering MCP Tools, but also supports registering MCP Prompts, MCP Resources, and more. It can connect to a remote Web Agent service, allowing both the page Client and the remote MCP Host to call the page MCP Server's Tools, Prompts, and Resources.

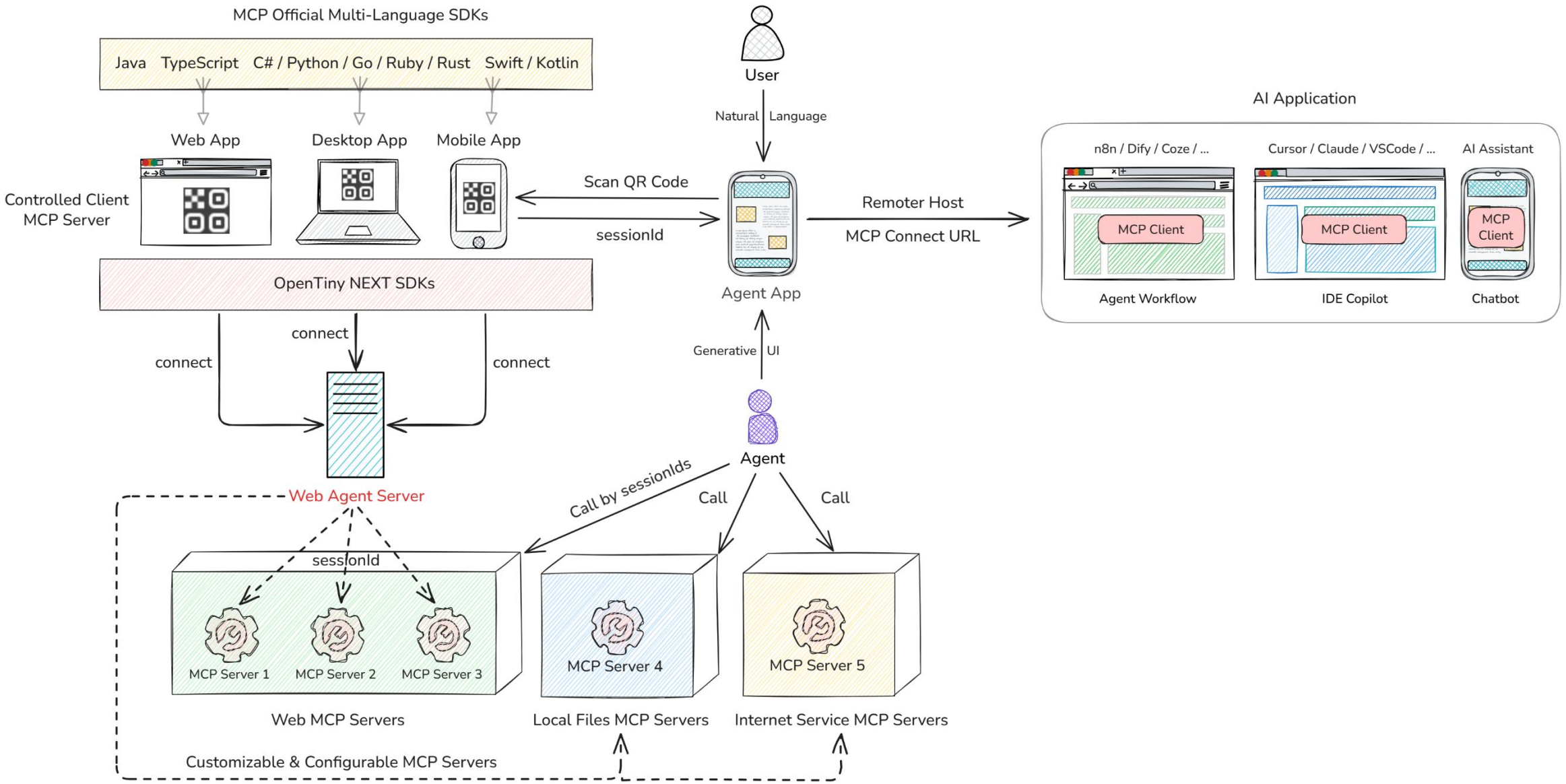
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>WebMCP</title>
    <script src="webmcp.js"></script>
    <script>
      const { createMessageChannelPairTransport, WebMcpServer, WebMcpClient, ResourceTemplate, z } = WebMCP;

      async function connect() {
        // Create pair MCP transports
        const [serverTransport, clientTransport] = createMessageChannelPairTransport();
        // Create an MCP server
        const server = new WebMcpServer({ name: 'demo-server', version: '1.0.0' });

        // Add an addition tool
        server.registerTool(
          'add',
          {
            title: 'Addition Tool',
            description: 'Add two numbers',
            inputSchema: { a: z.number(), b: z.number() }
          },
          async ({ a, b }) => ({
            content: [{ type: 'text', text: String(a + b) }]
          })
        );

        // Add a dynamic greeting resource
        server.registerResource(
          'greeting',
          new ResourceTemplate('greeting://{name}', { list: undefined }),
          {
            title: 'Greeting Resource',
            description: 'Dynamic greeting generator'
          },
          async (uri, { name }) => ({
            contents: [{ uri: uri.href, text: `Hello, ${name}!` }]
          })
        );

        // Add a code review prompt
        server.registerPrompt(
          'review',
          {
            title: 'Code Review',
            description: 'Review code for best practices and potential issues',
            argsSchema: { code: z.string() }
          },
          ({ code }) => ({
            messages: [{ role: 'user', content: { type: 'text', text: `code: ${code}` } }]
          })
        );

        // Create an MCP Client
        const client = new WebMcpClient({ name: 'demo-client', version: '1.0.0' });

        // Connect the client and server
        await server.connect(serverTransport);
        await client.connect(clientTransport);

        // Client callTool, readResource, and getPrompt
        console.log(await client.callTool({ name: 'add', arguments: { a: 5, b: 6 } }));
        console.log(await client.readResource({ uri: 'greeting://John' }));
        console.log(await client.getPrompt({ name: 'review', arguments: { code: 'x' } }));

        // Connect to the Web Agent server
        const { transport, sessionId } = await client.connect({
          url: 'https://agent.opentiny.design/api/v1/webmcp-trial/mcp',
          agent: true
        });

        // Display the session ID
        document.getElementById('sessionId').innerText = sessionId;

        window.addEventListener('pagehide', async () => {
          await transport.terminateSession();
        });
      }
    </script>
  </head>
  <body>
    <button onclick="connect()">Connect Web Agent</button>
    https://agent.opentiny.design/api/v1/webmcp-trial/mcp?sessionId=<span id="sessionId"></span>
  </body>
</html>
```

Remote MCP Host add the above MCP Server url

Our WebMCP Solution: SDK Demo

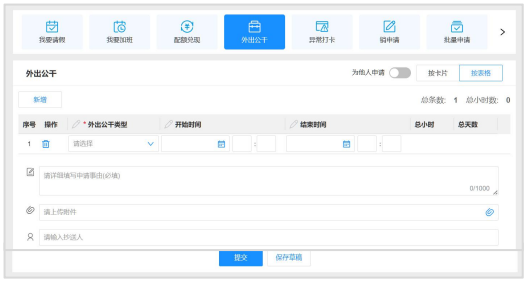


Our WebMCP Solution: Architecture



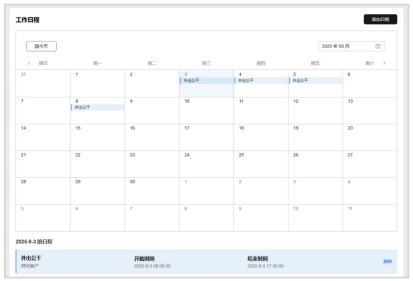
Our WebMCP Solution: Sample

Web App



On Business Trip

Desktop App



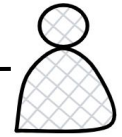
Daily Schedule

Mobile App

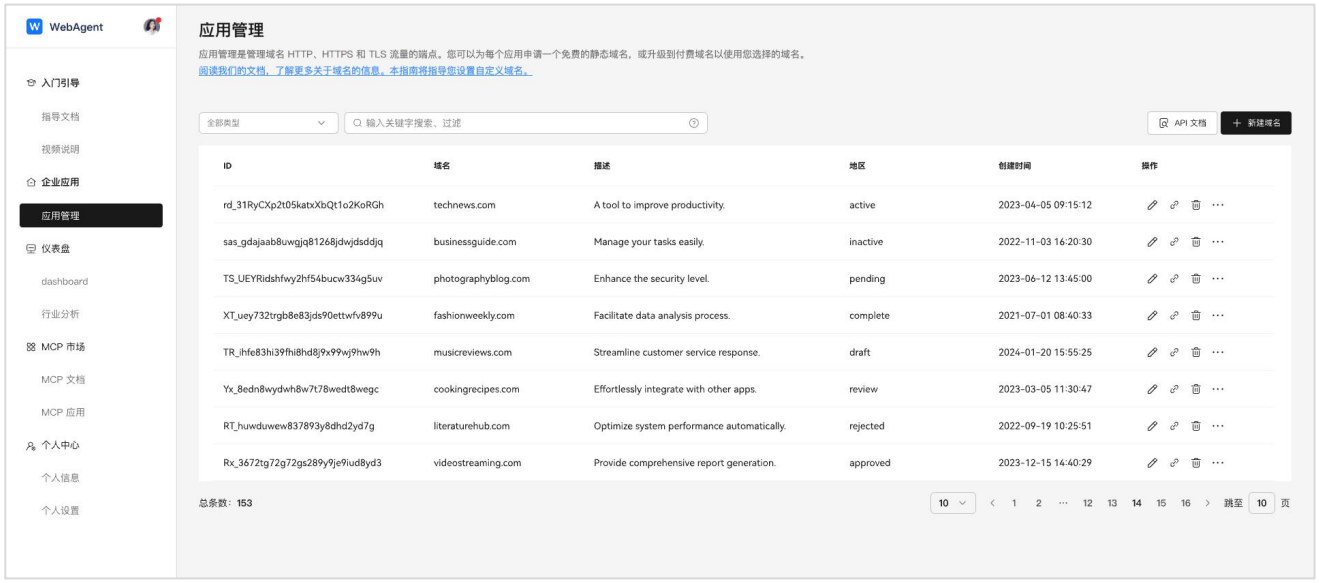
Travel Request

MCP Servers Call

TinyRobot



User



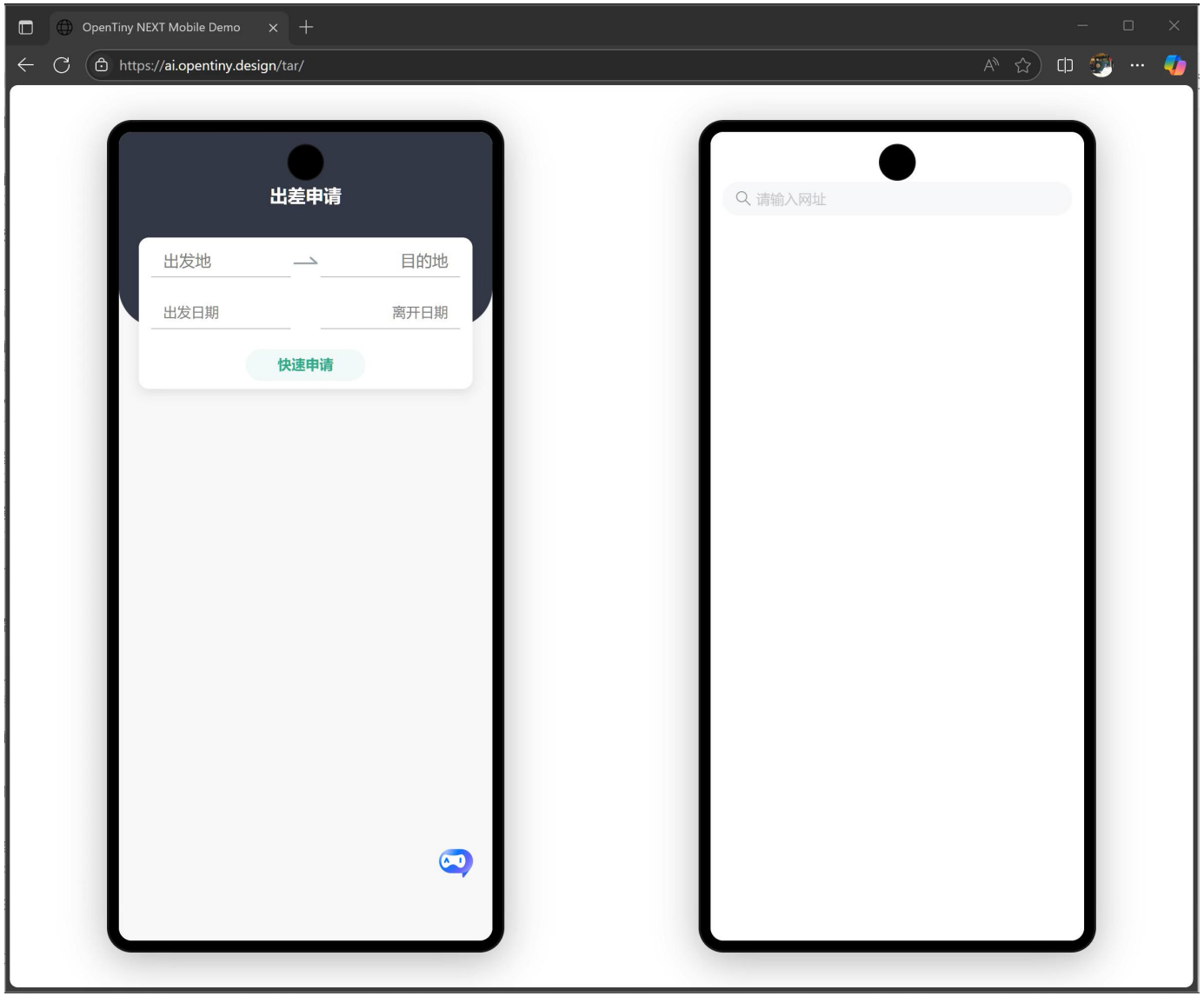
WebAgent Server Platform

TinyRobot Website

Our WebMCP Interactive Experience

Scan QR Code or Visit Website

https://ai.opentiny.design/tar/



Generative UI and MCP servers are the future of Web AI

THANKS