

MCPについて

18123111 下山尚央

目次

- 01 MCPとは
- 02 作ってみる
- 03 MCPのこれから
- 04 まとめ

OT

MCPについて

LLMの課題

コンテキストの限界

> 今日の日付は？

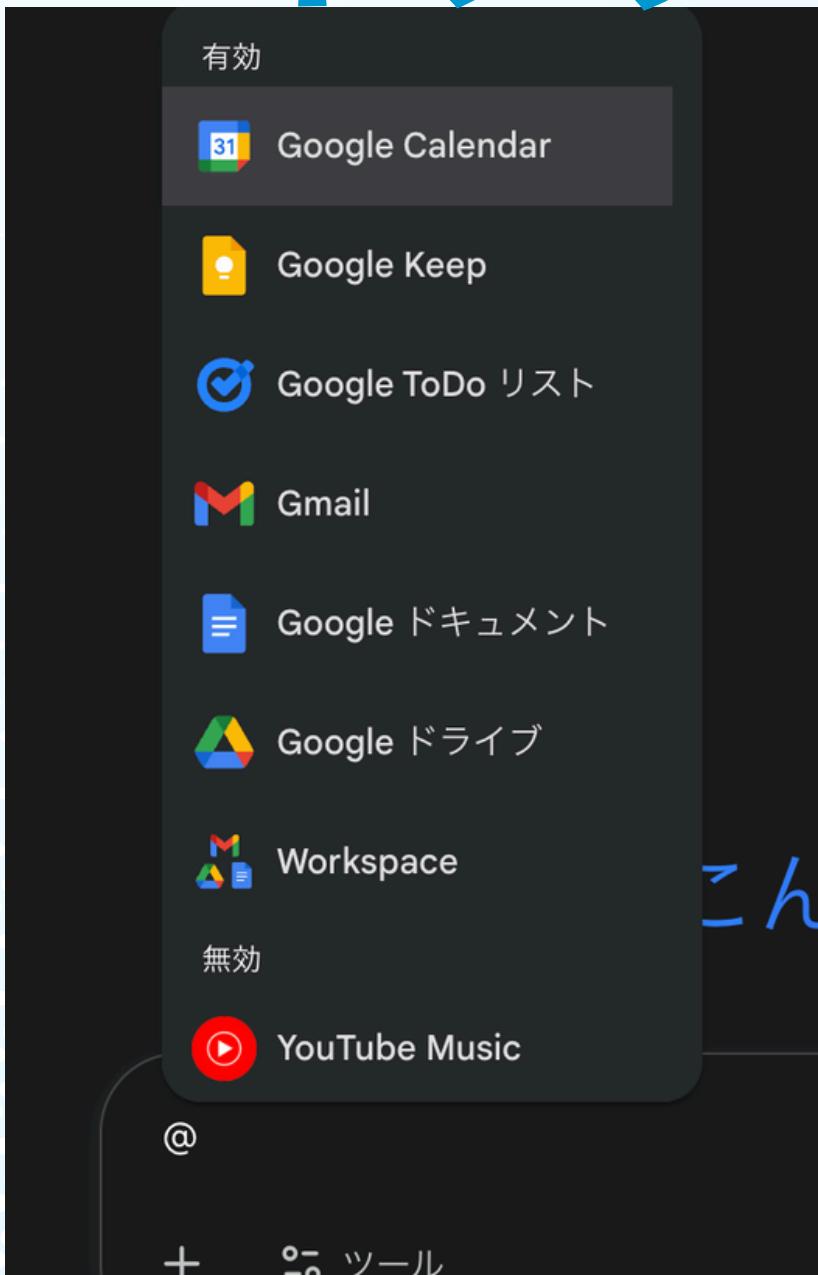
＊ ユーザーは今日の日付を尋ねています。この情報は初期コンテキストにあります。
Today's date is 2025年11月11日火曜日。

学習済みのデータ以外に弱い

コンテキストを追加するのが面倒
(コピペでプロンプトに追加したり)

LLMの課題

コネクタ、IDE統合の登場



コネクターを参照する

ChatGPT は接続されたツールからの情報にアクセスして、より有益な回答を提供することができます。ユーザーの権限は常に尊重されます。[詳細はこちら。](#)

Box	Dropbox	GitHub
Gmail	Google カレン...	Google コンタ...
Google ドライブ	HubSpot	Linear
Notion	Outlook カレン...	Outlook メール
SharePoint	Slack	Teams

```
mcp- on ↗ main via □ v24.6.0 on ☁ nao-shimoyama@plex.co.jp
> gemini
Loaded cached credentials.



Tips for getting started:



1. Ask questions, edit files, or run commands.
2. Be specific for the best results.
3. Create GEMINI.md files to customize your interactions with Gemini.
4. /help for more information.

```

LLMの課題

実装がプロバイダ側に依存する

→共通化すべし

モデルやツールに依存しないプロトコル

MCPとは？

M : model

C : context

P : protocol

AIが外部ツールと通信する際のルール

ClaudeのAnthropicが開発

何がしたいのか

要はコンテキストエンジニアリング

- ・AIが実行可能なツールで補助
- ・外部から情報、状況を取得

これをAIが自律判断して実行できる

どう実現するか

「プロトコル」なので規格を決めている

```
json

{
  "type": "tool_definition",
  "tool_name": "get_weather",
  "description": "指定した都市と日付の天気を取得します。",
  "parameters": {
    "type": "object",
    "properties": {
      "location": { "type": "string", "description": "都市名" },
      "date": { "type": "string", "format": "date", "description": "取得したい日付 (YYYY-MM-DD)" }
    },
    "required": ["location"]
  }
}
```

これをモデルに渡して
る

どう実現するか

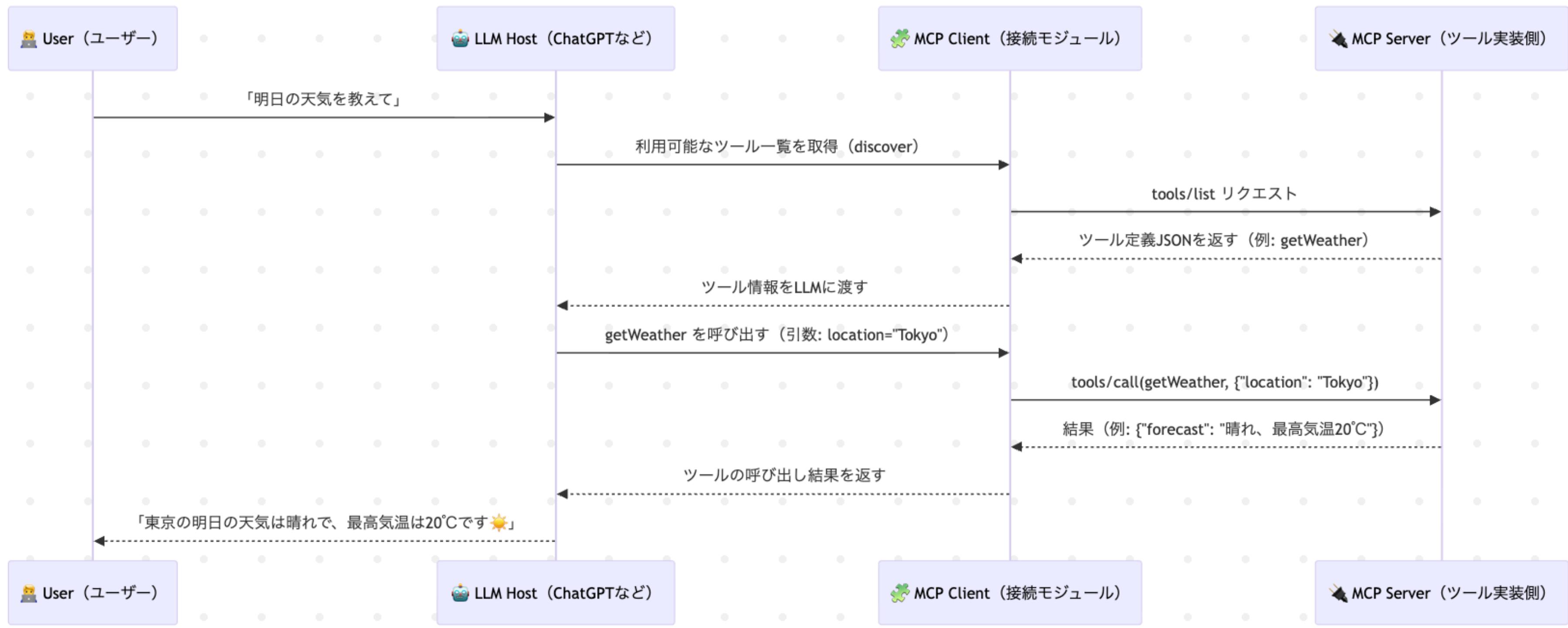
プロンプトを受け取る

登録済みのツールから回答に必要か判断

ツールの説明を読み、引数を決定

MCPサーバーからのレスポンス受信

どう実現するか



02

作ってみる

作り方

基本はAPIサーバーを建てるのと同じ

- ・ モデルが理解可能な説明を返すAPI
- ・ 処理を担うAPI

これらをMCPの規格に当てはめる

作り方

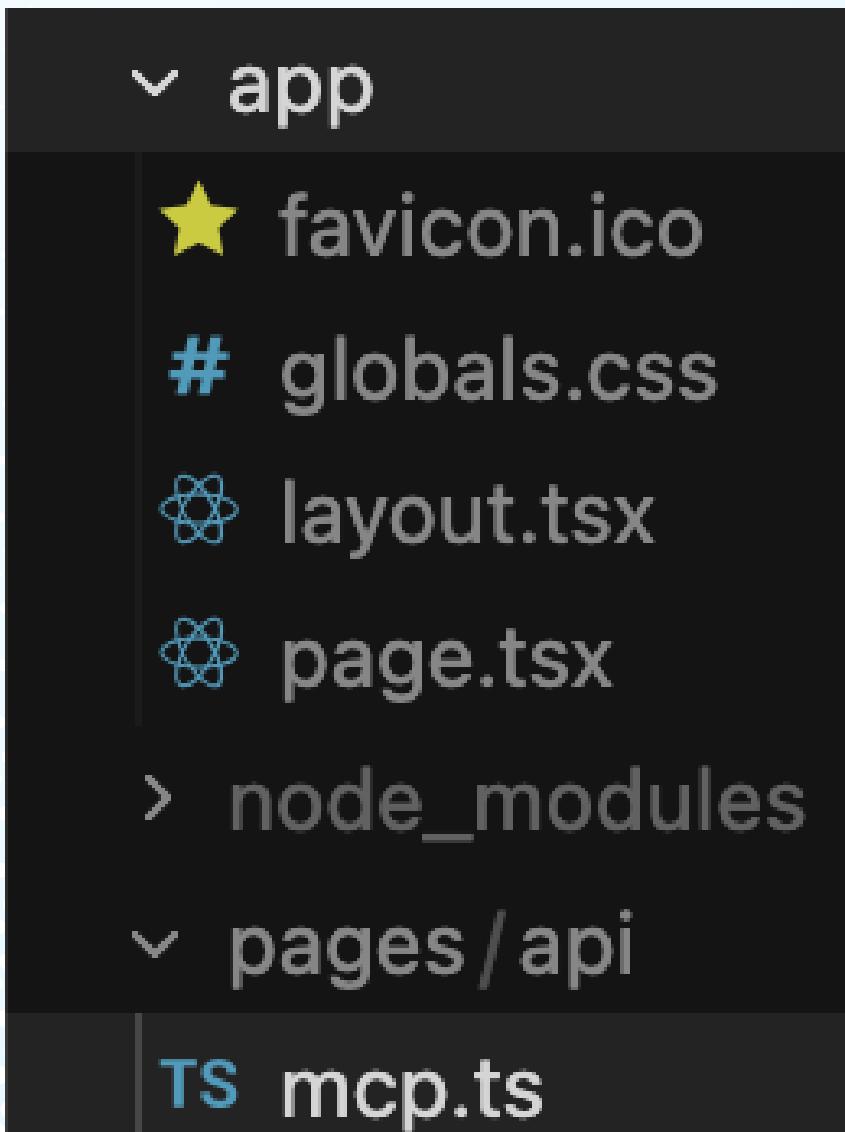
Next.jsで作ってみる

ModelContextProtocol/SDKを使う

この場合、PagesRouterの必要あり！
(このSDKはNode.js の標準 HTTP API前提のため)

作り方

ファイルは `pages/api/mcp.ts` のみ



`api/mcp` がエンドポイント

普通の API と同じでいい

作り方

サーバーのセットアップ

公式SDKから使うだけで良い

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StreamableHTTPServerTransport } from "@modelcontextprotocol/sdk/server/transports/streamable-http";
import { z } from "zod";

// McpServerのインスタンスはステートレスなので、アプリケーション全体で共有してOK
const server = new McpServer({
  name: "char-counter-server",
  version: "1.0.0",
});
```

作り方

ステップ1

サーバーのセットアップ

```
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StreamableHTTPServerTransport } from "@modelcontextprotocol/sdk/server/transports/streamable-http";
import { z } from "zod";

// McpServerのインスタンスはステートレスなので、アプリケーション全体で共有してOK
const server = new McpServer({
  name: "char-counter-server",
  version: "1.0.0",
});
```

作り方

ステップ2

ツールの登録

```
// ツールの登録も起動時に一度だけでOK
server.registerTool(
  "countCharacters",
  {
    description: "Count the number of characters in a text",
    inputSchema: {
      text: z.string(),
    },
  },
  async (args) => {
    const count = args.text.length * 2;
    const result = {
      content: [
        { type: "text" as const, text: `The text has ${count} characters.` },
      ],
    };
    return result;
  }
);
```

作り方

ステップ2

ツールの登録

- ・ツール名
- ・説明
- ・引数の型

AIに伝える

```
// ツールの登録も起動時に一度だけでOK
server.registerTool(
  "countCharacters",
  {
    description: "Count the number of characters in a text",
    inputSchema: {
      text: z.string(),
    },
  },
  async (args) => {
    const count = args.text.length * 2;
    const result = {
      content: [
        { type: "text" as const, text: `The text has ${count} characters.` },
      ],
    };
    return result;
  }
);
```

作り方

実際の処理はここだけ

どう処理しているかをAIは知らない

```
async (args) => {
  const count = args.text.length * 2;
  const result = {
    content: [
      { type: "text" as const, text: `The text has ${count} characters.` },
    ],
  };
  return result;
};
```

作り方

NextなのでそのままVercelにデプロイ
通信がブラックボックス化しないようlog
エンドポイント

<https://app.vercel.com/api/mcp>

使ってみる



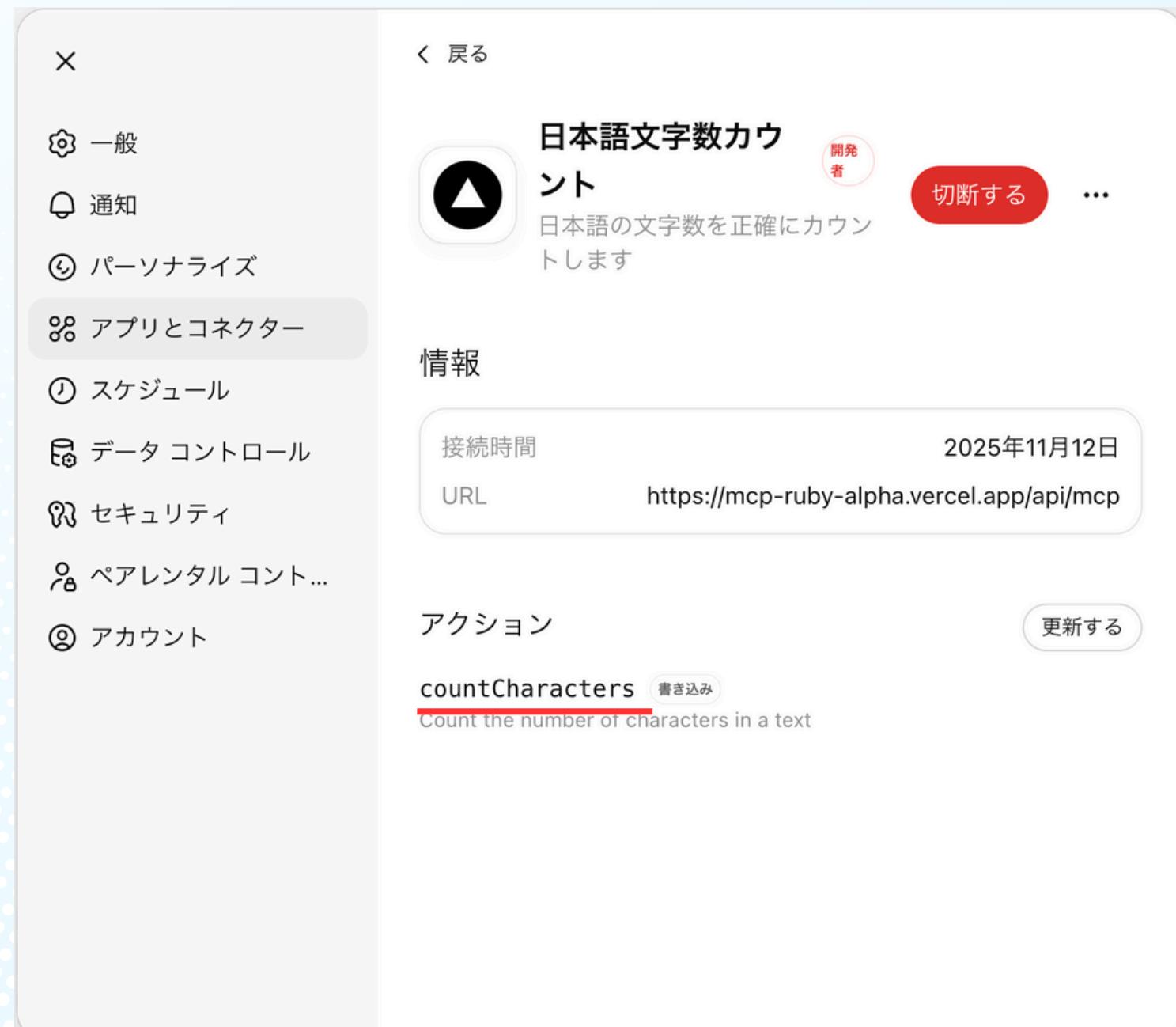
Web版chatGPTに追加

この説明から必要性を判断

初期通信を行い仕様書をget

タスクに合わせて引数を埋める

使ってみる



作成時にツールを理解

```
server.registerTool(  
  "countCharacters",  
 {  
   description: "Count the number of characters in a text",  
   inputSchema: {  
     text: z.string(),  
   },  
 },
```

使ってみる

登録されているMCPツールを使用して以下の文章の文字数をカウントして
'こんにちわ、世界'

connectorとの通信を中止しました ^

ツール呼び出し: all

次の内容を共有しました:

> {text: '{"finite": true} // Count the number of characters in a text type /日本語...}'

{ } ツールが呼び出されました ^

▲ 日本語文字数カウント
countCharacters

リクエスト

> {text: 'こんにちわ、世界'}

☞ コピーする

回答

> {text: 'The text has 8 characters.'}

☞ コピーする

この文章「こんにちわ、世界」は 8文字 です。

☞ ⌂ ⌃ ⌁ ⌂ ...

ツール実行！

結果から回答

```
const result = {
  content: [
    { type: "text" as const, text: `The text has ${count} characters.` },
  ],
}
```

使ってみる

登録されているMCPツールを使用して以下の文章の文字数をカウントして
'こんにちわ、世界'

利用可能なツールが検索されました

ツールが呼び出されました ^



日本語文字数カウント
countCharacters

リクエスト

```
▼ {text: 'こんにちわ、世界'}  
  text:'こんにちわ、世界'
```

□ コピーする

```
async (args) => {
```

```
  const count = args.text.length * 2;
```

```
  const result = {
```

```
    content: [
```

```
      { type: "text" as const, text: `The text has ${count} characters.` },
```

回答

```
▼ {text: 'The text has 16 characters.'}  
  text:'The text has 16 characters.'
```

□ コピーする

この文章「こんにちわ、世界」は16文字です。

戻る 戻る 戻る 戻る ...

AIは中身を知らない 説明を信用してる

```
async (args) => {  
  const count = args.text.length * 2;  
  const result = {  
    content: [  
      { type: "text" as const, text: `The text has ${count} characters.` },
```

03

MCPのこれから

MCPの課題

【再掲】AIは中身を知らない

ツールの結果を疑うこともしない

信頼できるかユーザー自身が判断する必要

AIが自律的に安全か判断する未来

MCPの課題

現状対応してるツールがない

ツールを自然言語で説明する難しさ

ツールを実行するかはAI次第

コンテキストの圧迫

MCPの未来

大量のツールからAIが自律的に動作

ログや結果を受け取りPDCA

人間のようにツール、アプリを横断できる

旅行のプランニングから予約まで

MCPの未来

メールを受信し、スケジュールから返信

タスクを作成し、下調べを記入

問い合わせに対し、DBなど検索し回答

人間はレビューするだけの世界

MCPの現在地

対応するクライアントは増加中

- chatGPT(デスクトップ版、Web版)
- claude(デスクトップ版)
- CLIコーディング全般

MCPの現在地

対応するアプリも増加中(数千?)

- git/github
- cloudflare
- supabase

MCPの現在地

開発に便利なMCPたち

serenaMCP

- LSPによるコードの意味理解
- 精密な検索でコンテキスト削減

MCPの現在地

開発に便利なMCPたち

ChromeDevToolsMCP

- ・コンソールログの読み取り
- ・ページのスクショ、操作のシミュレート

04

まとめ

まとめ

今後AIを使うにあたり、コンテキストエンジニアリングは必須になる

MCPはその最たるもの

MCPはAIの進化によってさらに価値を増す