jsvelox — Vollstaendiger Leitfaden

Alles, was Sie ueber jsvelox wissen muessen

v2.2.0 — Codename: Quagga

github.com/yunus6141/jsvelox

// 01

Was ist jsvelox?

jsvelox ist eine leichtgewichtige JavaScript-Bibliothek, die es Ihnen ermoeglicht, KI-Anbieter - Claude, OpenAI und Gemini - mit nur wenigen Codezeilen in Ihre Projekte zu integrieren. Im Gegensatz zu schweren KI-Agent-Plattformen, die Ressourcen auch fuer ungenutzte Funktionen verbrauchen, folgt jsvelox einem einfachen Prinzip: Verwenden Sie nur das, was Sie benoetigen.

Entwickelt fuer Entwickler, die KI-Integration ohne den Overhead visueller Pipeline-Builder oder komplexer Server-Setups wuenschen.

// 02

Welche Probleme loest jsvelox?

Zu komplex	KI-Agent-Plattformen erfordern visuelle Schnittstellen und steile Lernkurven.	✔ jsvelox verwendet einfache JavaScript-Objekte - kein Drag-and-Drop.
Zu schwer	Plattformen wie n8n verbrauchen RAM fuer jeden Knoten, auch ungenutzte.	✔ jsvelox verwendet nur Ressourcen fuer das, was Sie tatsaechlich aufrufen.
Zu viele Dateien	Die Integration von KI bedeutet oft die Verwaltung mehrerer API-Clients.	✔ Eine Bibliothek, drei KI-Anbieter, einheitliche API.
Schwer anzupassen	Das Training einer KI mit eigenen Daten erfordert komplexe Setups.	✔ systemPrompt und context ermoeglicht KI-Training im selben Konfigurationsobjekt.

// 03

Code-Struktur

jsvelox basiert auf einer einzigen Klasse: VeloxAI. So funktioniert es:

Erforderliche Konfiguration

```
import VeloxAI from 'jsvelox'

const ai = new VeloxAI({
  provider: 'gemini',  // Erforderlich: claude | openai | gemini
  apiKey: 'ihr-schluessel',  // Erforderlich: Ihr API-Schluessel
  memoryLimit: 10,      // Optional: max. Nachrichten
  timeout: 15000        // Optional: ms bis Timeout
})

const antwort = await ai.send('Hallo!')
console.log(antwort)
```

KI-Trainingskonfiguration

```
const ai = new VeloxAI({
  provider: 'gemini',
  apiKey: 'ihr-schluessel',
  systemPrompt: 'Sie sind ein Shop-Assistent. Antworten Sie auf Deutsch.',
  context: [
    { name: 'Produkt A', preis: '29 EUR', vorteil: 'Steigert Energie' },
    { name: 'Produkt B', preis: '49 EUR', vorteil: 'Verbessert Schlaf' }
  ]
})
```

// 04

Verfuegbare Methoden

Method	Description
`send(nachricht)`	Nachricht senden und Antwort empfangen
`sendWithRetry(msg, n)`	Senden mit automatischem Wiederholungsversuch (Standard: 3 Mal)
`onSend(btn, input, res)`	Schaltflaeche/Eingabe/Antwort in einer Zeile verbinden
`getMemory()`	Gespraechsverlauf abrufen
`clearMemory()`	Gespraechsverlauf loeschen

// 05

Haeufige Fehler und Loesungen

FEHLER	**API-Schluessel erforderlich!**
Cause:	VeloxAI wurde ohne apiKey erstellt.
Fix:	`Immer apiKey angeben:` `new VeloxAI({ provider: 'gemini', apiKey: 'schluessel' })`

FEHLER	**Anbieter erforderlich!**
Cause:	Kein KI-Anbieter angegeben.
Fix:	`provider hinzufuegen:` `new VeloxAI({ provider: 'gemini', apiKey: 'schluessel' })`

FEHLER	**Ungueltige Anbieter!**
Cause:	Nicht unterstuetzter Anbietername. Beispiel: provider: 'gpt'
Fix:	`Nur verwenden: claude | openai | gemini`

FEHLER	**Nachricht darf nicht leer sein!**
Cause:	ai.send() mit leerem String aufgerufen.
Fix:	`Nachricht vor dem Senden pruefen:` `if (msg.trim()) { await ai.send(msg) }`

WARNUNG	**Zeitlimit ueberschritten!**
Cause:	KI hat nicht innerhalb des Zeitlimits geantwortet.
Fix:	`Timeout erhoehen oder sendWithRetry verwenden.`

FEHLER	**Ungueltiger API-Schluessel**
Cause:	Ihr API-Schluessel ist falsch oder abgelaufen.
Fix:	`Gueltigen Schluessel von console.anthropic.com, platform.openai.com` `oder aistudio.google.com holen.`

WARNUNG	**Kontingent ueberschritten**
Cause:	Tageskontingent der kostenlosen API verbraucht.
Fix:	`Warten Sie auf Kontingent-Ruecksetzung oder upgraden Sie Ihren Plan.`

FEHLER	**Elemente nicht gefunden**
Cause:	CSS-Selektoren fuer onSend() stimmen nicht ueberein.
Fix:	`Selektoren pruefen:` `ai.onSend('#btn', '#input', '#ausgabe')`

WARNUNG	**CORS-Fehler (Browser)**
Cause:	API direkt vom Browser aufgerufen.
Fix:	`jsvelox im Node.js-Backend verwenden, nicht direkt im Browser.`

// 06

Fehlerbehandlung mit try-catch

```
try {
  const antwort = await ai.send('Hallo!')
  console.log(antwort)
} catch (fehler) {
  if (fehler.message.includes('Timeout')) {
    console.log('KI zu langsam, bitte erneut versuchen')
  } else {
    console.log('Fehler:', fehler.message)
  }
}
```

// 07

Wann funktioniert jsvelox am besten?

- ✓ Node.js Backend-Anwendungen
- ✓ Express.js oder Fastify-Server
- ✓ Next.js API-Routen
- ✓ CLI-Tools und Skripte
- ✓ Beliebige JavaScript/TypeScript-Projekte auf Node.js 18+

Wann funktioniert jsvelox NICHT?

- ✗ Direkte Browser-Nutzung (CORS-Einschraenkung) - kommt in v3.0
- ✗ Deno oder Bun Umgebungen (nicht getestet)
- ✗ Node.js-Versionen unter 18

// 08

TypeScript-Verwendung

```
import VeloxAI, { VeloxConfig } from 'jsvelox'

const config: VeloxConfig = {
  provider: 'gemini',
  apiKey: process.env.GEMINI_KEY!,
  memoryLimit: 20
}

const ai = new VeloxAI(config)
const antwort: string = await ai.send('Hallo!')
```