

Built-in AI APIs for the Web

Thomas Steiner
tomac@google.com

Adding AI to your web app

Imagine you build an app to let users record and organize creative ideas:
ideas.example.com .

Features:

- Record ideas by voice (e.g., musical ideas like a tune or a melody, so OS-native dictation isn't an option).
- Enter ideas by keyboard (e.g., a catchy headline for a blog post).
- Sketch ideas with a drawing (e.g., the dimensions of a piece of furniture).
- Grab ideas by photo (e.g., to recall the design of a fabric).

Adding AI to your web app (cont'd)

Possible AI use:

- *"List all my ideas where I hum."*
- *"Based on this headline, write an opening paragraph."*
- *"How much wood do I need given this sketch?"*
- *"Where can I buy a t-shirt with this design?"*
- *"Rewrite this rough idea more formally."*
- *"Share this idea translated to Japanese with my friend 英治."*
- *"Summarize all my ideas from yesterday afternoon."*

Adding AI to your web app (cont'd)

Option 1: Use an AI provider in the cloud ☀️

- + Some of the most powerful models run (exclusively) in the cloud.
- + The app can remain small and doesn't consume much disk space.
- + No device hardware requirements, as all AI inference happens on the server.

- Can get expensive and make cost prediction unclear.
- Requires user data to be sent to the cloud.
- Doesn't work offline.

Adding AI to your web app (cont'd)

Option 2: Download AI models locally and run on the client

- + Privacy-preserving, as all processing happens locally.
- + Works offline once the initial model download has happened.
- + Is free to the developer as all inference happens on the client.

- Model downloads can be prohibitively large, and require lots of disk space.
- May not work on all hardware or be slow.
- Limits what models can be used, no way to protect a model.

Adding AI to your web app (cont'd)

Option 2: Download AI models locally and run on the client

- + Privacy-preserving, as all processing happens locally
- + Works offline once the initial model download has happened
- + Is free to the developer as all inference happens on the client

shameless plug



- Model downloads can be prohibitively large, and require lots of disk space.
- May not work on all hardware or be slow.
- Limits what models can be used, no way to protect a model.

Tuesday 3rd

Time CEST	HTML Room (rooms 9 + 10)	JS Room (rooms 4 + 5)	CSS Room (rooms 1 + 2)
10:00- 11:30	Web Platform: What are we working on?	TC39 proposals and integration with the web platform	Hackathon: Browser integration of Fontaines Rust font stack
11:30- 12:00	Break	Break	Break
12:00- 13:30	Browser Funding After Search Deals	Cross-Origin Storage (COS)	Multimedia in WebKit
13:30- 15:00	Lunch	Lunch	Lunch
15:00- 16:30	Browser fingerprinting defense: review of the strategies used by different browsers	JavaScript subsets for custom embedding use-cases	tvOS as a new platform for Chromium
16:30- 18:00	HTML-in-Canvas: Proposal and Progress	Gosub & Yavashark: A New Browser & JS/TS Engine	Chromium Embedding: Experiences and challenges

Adding AI to your web app (cont'd)

Proposed option 3: Make a shared AI model available to Web APIs (built-in AI)

- + Privacy-preserving, as all processing happens locally.
- + Works offline once the initial model download has happened.
- + Is free to the developer as all inference happens on the client.
- ± Model downloads can be prohibitively large, but it's a one-time operation.

- May not work on all hardware or be slow.
- Limits what models can be used, as you're bound to what the browser ships.

Built-in AI APIs



Web Machine Learning Community Group Charter

- This Charter: <https://webmachinelearning.github.io/charter/>
- Previous Charter: [a4da99c](#)
- Start Date: 2025-05-19
- Last Modified: [commits/master](#)

webmachinelearning.github.io/charter/#scope-of-work

Scope of Work

Task-specific APIs

A set of higher-level APIs that:

- Detect the language of the given input text;
- Translate input text to other languages;
- Produce textual summaries of input text;
- Produce new textual material given a writing task;
- Rephrase input text;
- Proofread input text.

Prompt API

A general-purpose API to prompt a language model directly using common prompting techniques to accomplish a variety of tasks, including:

- Classification, tagging, and keyword extraction of arbitrary text;
- Helping users compose text, such as blog posts, reviews, or biographies;
- Summarizing, e.g. of articles, user reviews, or chat logs;
- Generating titles or headlines from article contents;
- Answering questions based on the unstructured contents of a web page;
- Translation between languages.

Translator and Language Detector APIs



TABLE OF CONTENTS

1	Introduction
2	Dependencies
3	The translator API
3.1	Creation
3.2	Availability
3.3	The Translator class
3.4	Translation
3.4.1	The algorithm
3.4.2	Usage
3.4.3	Errors
3.5	Permissions policy integration
4	The language detector API
4.1	Creation
4.2	Availability
4.3	The LanguageDetector class
4.4	Language detection
4.4.1	The algorithm
4.4.2	Usage

Translator and Language Detector APIs

Draft Community Group Report, 9 May 2025



▼ More details about this document

This version:

<https://webmachinelearning.github.io/translation-api>

Issue Tracking:

[GitHub](#)

Editor:

[Domenic Denicola \(Google\)](#) d@domenic.me

Copyright © 2025 the Contributors to the Translator and Language Detector APIs Specification, published by the Web Machine Learning Community Group under the W3C Community Contributor License Agreement (CLA). A human-readable [summary](#) is available.

Abstract

The translator and language detector APIs give web pages the ability to translate text between languages, and detect the language of such text.

Status of this document

This specification was published by the [Web Machine Learning Community Group](#). It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the [W3C Community Contributor License Agreement \(CLA\)](#) there is a limited opt-out and other conditions apply. Learn more about [W3C Community and Business Groups](#).

Writing Assistance APIs



TABLE OF CONTENTS

1	Introduction
2	The summarizer API
2.1	Creation
2.2	Availability
2.3	The Summarizer class
2.4	Summarization
2.4.1	The algorithm
2.4.2	Usage
2.4.3	Options
2.4.4	Errors
2.5	Permissions policy integration
3	The writer API
3.1	Creation
3.2	Availability
3.3	The Writer class
3.4	Writing
3.4.1	The algorithm
3.4.2	Usage
3.4.3	Options

Writing Assistance APIs

Draft Community Group Report, 22 May 2025



▼ More details about this document

This version:

<https://webmachinelearning.github.io/writing-assistance-apis>

Issue Tracking:

[GitHub](#)

Editor:

[Domenic Denicola \(Google\)](mailto:domenic.denicola@google.com) d@domenic.me

Copyright © 2025 the Contributors to the Writing Assistance APIs Specification, published by the [Web Machine Learning Community Group](#) under the [W3C Community Contributor License Agreement \(CLA\)](#). A human-readable [summary](#) is available.

Abstract

The summarizer, writer, and rewriter APIs provide high-level interfaces to call on a browser or operating system's built-in language model to help with writing tasks.

Status of this document

This specification was published by the [Web Machine Learning Community Group](#). It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the [W3C Community Contributor License Agreement \(CLA\)](#) there is a limited opt-out and other conditions apply. Learn more about [W3C Community and Business Groups](#).

webmachinelearning.github.io/writing-assistance-apis/

Prompt API

The screenshot shows a GitHub repository page for 'prompt-api'. The repository is owned by 'webmachinelearning' and has 26 issues. The 'Code' tab is selected. The main content is the 'README.md' file, which contains the following text:

Explainer for the Prompt API

This proposal is an early design sketch by the Chrome built-in AI team to describe the problem below and solicit feedback on the proposed solution. It has not been approved to ship in Chrome.

Browsers and operating systems are increasingly expected to gain access to a language model. ([Example](#), [example](#), [example](#).) Language models are known for their versatility. With enough creative [prompting](#), they can help accomplish tasks as diverse as:

- Classification, tagging, and keyword extraction of arbitrary text;
- Helping users compose text, such as blog posts, reviews, or biographies;
- Summarizing, e.g. of articles, user reviews, or chat logs;
- Generating titles or headlines from article contents
- Answering questions based on the unstructured contents of a web page
- Translation between languages
- Proofreading

github.com/webmachinelearning/prompt-api/

Adding built-in AI to your web app

Possible AI use:

- *"List all my ideas where I hum."* → **Prompt API with audio input**
- *"Based on this headline, write an opening paragraph."* → **Writer API**
- *"How much wood do I need given this sketch?"* → **Prompt API w/ image input**
- *"Where can I buy a t-shirt with this design?"* → **Prompt API w/ image input**
- *"Rewrite this rough idea more formally."* → **Rewriter API**
- *"Share this idea translated to Japanese with my friend 英治."* → **Translator API**
- *"Summarize all my ideas from yesterday afternoon."* → **Summarizer API**

Live demo of the APIs

Built-in AI APIs

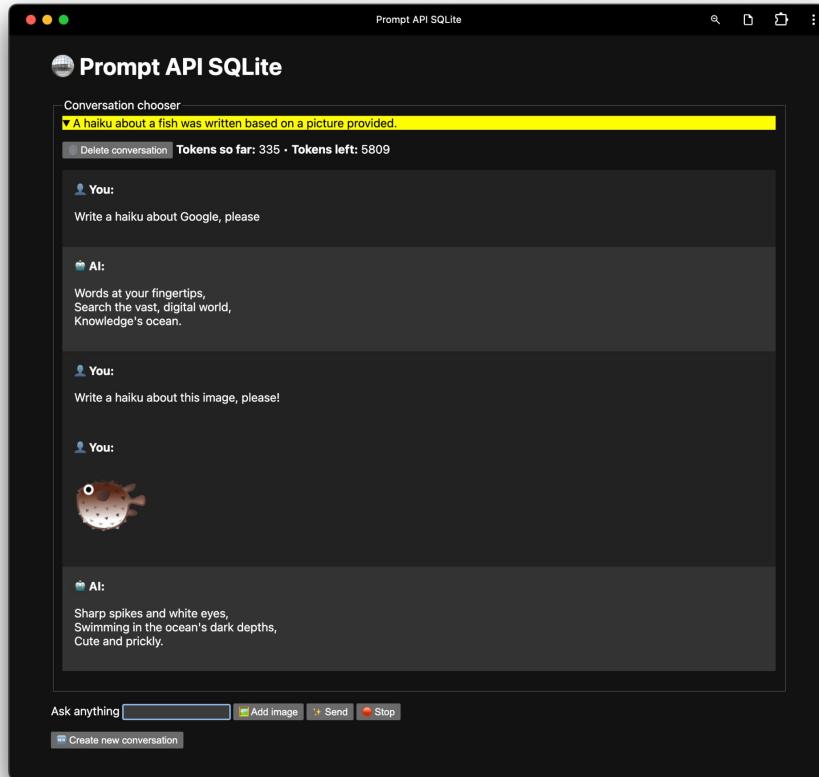
Web Engines Hackfest rocks!



Thomas Steiner
Developer Relations Engineer on the Google Chrome team (tomac@google.com)
Biography
Thomas Steiner is a Developer Relations Engineer at Google, focused on Web AI, WebAssembly, and Project Fugu. He's an alumnus of University of Lyon (Postdoc), Polytechnic University of Barcelona (Ph. D.), and University of Karlsruhe (MA).
He blogs at blog.tomayac.com and posts as @tomayac@toot.cafe. On all other places on the Internet, chances are that he's there under his online alias *tomayac*.
Prompt:

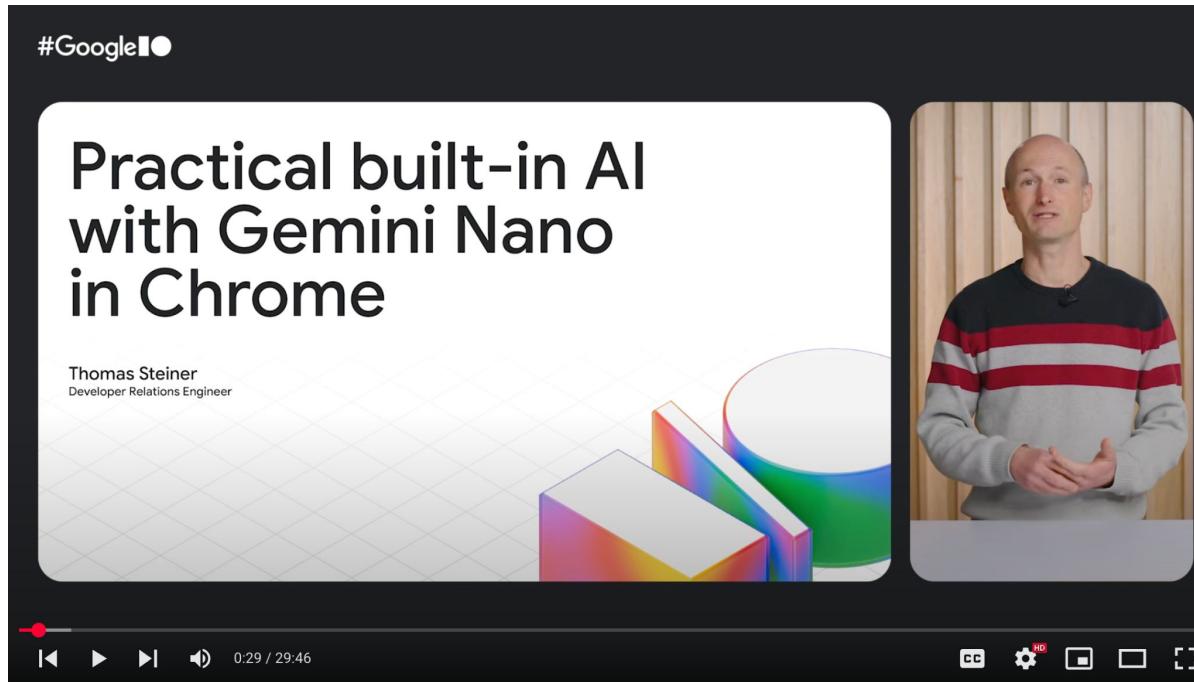
web-engines-hackfest-rocks.glitch.me/

Live demo of the APIs (cont'd)



tomayac.github.io/prompt-api-sqlite

Example partner implementations



youtu.be/CjpZCWYrSxM





Blogger helper function
extension for blog writers
powered by the **Prompt API**
for Chrome Extensions.

- Generating blog titles and headings.
- Improving and editing sentences.
- Drafting subsequent paragraphs.

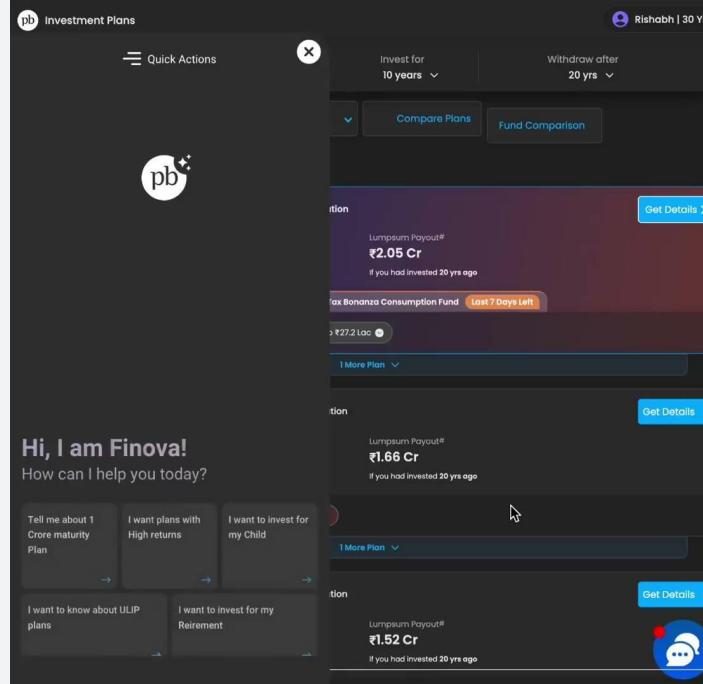
The screenshot shows the Ameba blog editor interface. At the top, there's a navigation bar with the Ameba logo, a 'ホームへ' (Home) link, and a '管理トップへ' (Manage Top) link. Below the navigation is a toolbar with tabs for 'ブログを書く' (Write a blog) and 'プレビュー' (Preview). The main area is a rich text editor with a title input field containing 'タイトルを入力してください' (Please enter a title), a 'タイトル生成' (Title generation) button, and a text area with the following content:

いやー、今日も元気いっぱいのうちのわんこと、いつものコースをのんびりお散歩してきましたよ！😊
朝の空気って、なんだか気持ちいいんですね。
わんこもそれを分かってるのか、しっぽをフリフリ、鼻をクンクンさせながら、ぐいぐい引っ張っていくんです（笑）。
まあ、それがまた可愛くて、ついついペースを合わせちゃうんですけどね！
いつもの公園に着くと、同じように犬の散歩に来ている人たちがチラホラ。
お互いのわんこを遊びながら、飼い主同士でちょっとしたおしゃべりをするのも、散歩の楽しみの一つだったりします。

To the right of the editor, there's a sidebar with various icons for media and design tools, including '写真・動画' (Photos/Videos), '縮文字' (Shrink Text), 'デザイン' (Design), 'PICK', and 'YouTube'. It also shows settings for image sizes (原寸, 小, 中, 大) and a date selector for '2025' and 'すべての月' (All Months). At the bottom right, there's a large '上传' (Upload) button with a plus sign and the text 'ここにドロップまたはクリックで' (Drop or click here) and '画像をアップロード' (Upload image).



Policybazaar used the **Language Detector API** to detect the user's input language and the **Translator API** to translate messages to and from English. This enables efficient and accessible insurance assistance to users in their native language.





Nykaa, India's leading beauty and lifestyle destination, tested the **Prompt API with image input** to let users run visual search sessions to find fitting outfits.

A screenshot of the Nykaa Fashion website homepage. At the top, there is a navigation bar with links for "All Brands", "Women", "Men", "Kids", "Home", "Tech", and "More". To the right of the navigation bar are search, account, wishlist, and cart icons. The main banner features two women in colorful saris standing in a green field under a cloudy sky, with the text "SPRING SUMMER 2025" and "Your little crash-course on what's moving this season". Below the banner, there are circular thumbnails for various categories: Indianwear, Westernwear, Men, Footwear, Lingerie, Jewellery, Bags, Kids, and Home. A section titled "TRENDIEST CATEGORIES" shows five more images of fashion items. The bottom of the page features a decorative footer bar with various icons.



Chefkoch streamlined the recipe publishing process for their users by integrating the **Prompt API with image input**. The API functions as client-side OCR, converting handwritten or photographed recipes into structured JSON output that can then be edited on the Chefkoch site.

The screenshot shows a mobile application interface for Chefkoch. At the top, there is a logo with a chef's hat icon and the word "CHEFKOCH". In the top right corner, there is a red "Menu" button. Below the header, the word "Digitize" is displayed in large letters. A photograph of a spiral-bound notebook is shown, with a handwritten recipe for "Bananenbrot" written on it. The list includes ingredients like "100g weiches Butter", "4 reife Bananen", and "200g brauner Zucker". Below the list, there are instructions: "Bananen schälen und pürieren", "Butter, Zucker, Vanillezucker, Ei verquirlen", "Mehl, Backpulver, Salz, Zimt Walnüsse unterrühren", "Teig in Kastenform 180° ca. 60 min.", and "Auskühlen lassen". A green "Digitize" button is located at the bottom left of the image area. To the right of the image, the word "Bananenbrot" is displayed again, followed by a bulleted list of the same ingredients and instructions.

Bananenbrot

- 100g weiches Butter
- 4 reife Bananen
- 200g brauner Zucker
- 1 Ei
- 1 Pck. Vanillezucker
- 1 1/2 TL Backpulver
- 1 Prise Salz
- 1/2 TL Zimt
- 220g Mehl
- 1 handvoll Walnüsse

- Bananen schälen und pürieren
- Butter, Zucker, Vanillezucker, Ei verquirlen
- Mehl, Backpulver, Salz, Zimt, Walnüsse unterrühren
- Teig in Kastenform 180° ca. 60 min.
- Auskühlen lassen



BrightSites' integration of the **Prompt API with image input** empowers journalists to automatically generate titles, captions, alt text, and descriptions directly from images within their CMS.

Flow

Add new image 1 of 1

Title*
Chelsea boots in brown

Caption
Brown Chelsea Boots

Alt Text
Brown Chelsea Boots: A classic style for any occasion.

Copyright*

Description*
Two brown Chelsea boots lie flat, showcasing their leather texture and rubber soles.

Notes

Workflow
Published

Upload



OLX used the **Prompt API** with **audio input** to transcribe audio messages, enhancing accessibility and enabling them to be read in more places and less time than the original audio.

4.7 (3) Livros R\$ 150,00 :

Ver perfil

São todos os livros da foto por esse preço
00:06 17:10 ✓

Por quanto você faz só o a garota do lago
00:15 17:11 ✓

Digite uma mensagem...

Firebase AI Logic

The screenshot shows the Firebase AI Logic documentation page. At the top, there's a navigation bar with links for Build, Run, Solutions, Pricing, Docs (which is highlighted), Community, Support, a search bar, and language selection (English). On the left, a sidebar lists categories like Overview, DEVELOP USING AI (with sub-options for Firebase Studio, Gemini in Firebase, and Firebase MCP Server), and BUILD AI-POWERED APPS (with sub-options for Firebase AI Logic, Introduction, and Get started). The main content area features a banner about announcements at I/O, followed by a breadcrumb trail (Firebase > Documentation > Firebase AI Logic > AI) and a "Was this helpful?" button with thumbs up and down icons. The main title is "Build hybrid experiences with on-device and cloud-hosted models". Below it, a note says "Experimental: Using the Firebase AI Logic SDKs to build hybrid experiences is an Experimental feature, which means that it isn't subject to any SLA or deprecation policy and could change in backwards-incompatible ways." A note also states that the initial release supports on-device inference for web apps running on Chrome on Desktop.

Firebase

Build ▾ Run ▾ Solutions Pricing Docs ▾ Community ▾ Support

Search / English ▾

Documentation > Firebase AI Logic

Overview Fundamentals ▾ AI ▾ Build ▾ Run ▾ Reference Samples

Filter

Overview

DEVELOP USING AI

Firebase Studio

Gemini in Firebase

Firebase MCP Server

BUILD AI-POWERED APPS

Firebase AI Logic

Introduction

Get started

Here's everything we announced at I/O, from new Firebase Studio features to more ways to integrate AI. [Read blog.](#)

Firebase > Documentation > Firebase AI Logic > AI

Was this helpful? Like Dislike

Build hybrid experiences with on-device and cloud-hosted models

Send feedback

Experimental: Using the Firebase AI Logic SDKs to build hybrid experiences is an Experimental feature, which means that it isn't subject to any SLA or deprecation policy and could change in backwards-incompatible ways.

This initial release only supports on-device inference for web apps running on Chrome on Desktop.

Info

Chat

Seeking feedback



Mozilla Standards Positions

This page tracks Mozilla's positions on open Web and Web-related specifications submitted to standards bodies like the IETF, W3C, WHATWG, and Ecma TC39. Please remember, this isn't a commitment to implement or participate; it's just what we think right now. See [dev-platform](#) to find out what we're implementing. Want Mozilla's position on a specification? [Find out more](#).

Search Writing Assistance APIs

Link Specification Position Concerns Topics Venues More info ↗

Writing Assistance APIs ↗

#1067

torgo on Aug 7, 2024

Hi Domenic - Our feedback that we sent above stands, particularly regarding the name space of this API. We don't think it belongs in a .ai namespace. We think that the name spaces should be function-led rather than technology-used-to-deliver-the-function-led. Your proposal under [point 6](#) (AITranslatorCapabilities) is a step forward but we would prefer [Translator.capabilities\(\)](#) . We also think your response to point 3 of our feedback is fine. We also would strongly encourage that the spec include a non-normative note encouraging the use of professional translation services where possible (point 7 of our original feedback). Please consider this. We're going to close as [satisfied with concerns](#) due to these concerns. Given the current state of the spec, and lack of venue, we understand this to be an early review. We expect that addressing the points above will be important for further standardization. ✨

3

torgo closed this as [completed](#) on Aug 7, 2024

ML Prompt API #495

Open

domenic opened 3 weeks ago · edited by domenic

WebKittens @marcoscosses

Please remember, this isn't a commitment to implement or participate; it's just what we think right now. See [dev-platform](#) to find out what we're implementing. Want Mozilla's position on a specification? [Find out more](#).

Search Writing Assistance APIs

Link Specification Position Concerns Topics Venues More info ↗

Writing Assistance APIs ↗

#1067

ML Prompt API #495

Open

domenic opened 3 weeks ago · edited by domenic

Labels No labels

Type In-Task

Projects No projects

Milestone No milestone

Relationships None yet

Development Code with Capital Agent Mode

No branches or pull requests

Notifications Customise

You're receiving notifications because you're subscribed to this thread.

Participants

ML Prompt API #1213

Open

domenic opened on Apr 28

Specification title Prompt API

Specification or proposal URL (if available)

No response

Explainer URL (if available)

<https://github.com/mozilla/web-platform-tests/tree/main/test/testMLPromptAPI.js>

Proposed author(s)

domenic

MDN URL

No response

Caniuse.com URL

No response

Bugzilla URL

No response

Mozilla who can provide input

No response

WebKit standardize-position

No response

Other information

File a new pull request at [https://github.com/whatwg/html/pull/new](#)

Code with Capital Agent Mode

No branches or pull requests

Notifications Customise

You're receiving notifications because you're subscribed to this thread.

Participants

Mozilla

Link Specification Position Concerns Topics Venues More info ↗

Writing Assistance APIs ↗

#1067

Mozilla



Learn more about Chrome's implementation

chrome for developers Get inspired Blog Docs ▾ New in Chrome Search

AI on Chrome

- AI
- Built-in
- WebGPU
- Extensions and AI
- DevTools and AI

Gemini Nano for Chrome

- Built-in AI
- Get started
- Benefits of client-side AI
- Join the EPP

APIs

- API status and overview
- Writer API
- Rewriter API
- Translator API
- Language Detector API

Home > Docs > AI on Chrome > Built-in Was this helpful?

Get started with built-in AI

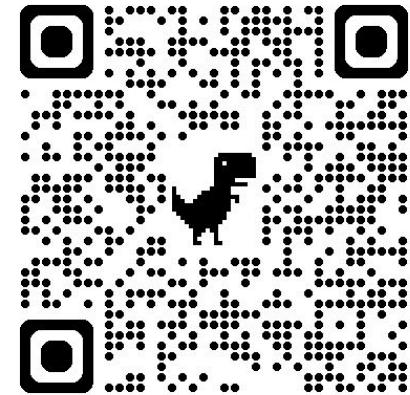
Alexandra Klepper

Published: December 12, 2024, Last updated: May 20, 2025

With [built-in AI APIs](#), your web application can perform AI-powered tasks without needing to deploy or manage its own AI models.

Requirements

We are working to [standardize these APIs across browsers](#).



Thank you

Thomas Steiner
tomac@google.com