

# Built-in AI Early Preview Program

## *The Writer and Rewriter APIs*

### Authors

Thomas Steiner  
Kenji Baheux

### Contact

See [this section](#)

### Last-updated

Sep 4, 2024  
See [changelog](#).

## Intro

Thanks for participating in our Early Preview Program for built-in AI capabilities ([article](#), [talk at Google I/O 2024](#)). As always we are [eager to hear your feedback](#) about this program and our APIs.



Know of other folks who would love to join this program? Or perhaps you got access to this document from a friend?

**Sign up** to get the latest updates directly in your inbox.

This update provides an early look at the writer and rewriter APIs, which are part of our on-going work on task APIs. You can explore how these APIs work through local prototypes.

## Writer and Rewriter APIs

### Purpose

The writer API empowers you to create new content that conforms to the specified writing task, while the rewriter API provides tools for revising and restructuring text.



### The Writer and Rewriter APIs vs. the Prompt API

Internally, the Writer and Rewriter APIs rely on a fine-tuned model for the tasks of writing and rewriting, whereas the Prompt API uses the baseline model. Unlike with the free-form Prompt API, with the Writer and Rewriter API we know what the use case is, and can therefore prime the model accordingly.

### Early Preview Goals

The goals for this early preview are to hear your feedback on the following aspects:


1. The **quality of the written or rewritten texts**, via [this feedback channel](#).

2. Issues with Chrome's current implementation, via [this feedback channel](#).
3. The [eventual shape of the API](#), via [this feedback channel](#).

## Requirements

Our Built-in AI program is **currently focused on desktop platforms**. In addition, the following conditions are required for Chrome to download and run Gemini Nano.

Aspect	Windows	MacOS	Linux
OS version	10, 11	≥ 13 (Ventura)	Not specified
Storage	At least 22 GB on the volume that contains your Chrome profile. <i>Note that the model requires a lot less storage, it's just for the sake of having an ample storage margin. If after the download the available storage space falls below 10 GB, the model will be <b>deleted again</b>.</i>		
GPU	Integrated GPU, or discrete GPU (e.g. video card).		
Video RAM	4 GB (minimum)		
Network connection	A non-metered connection		

	These are not necessarily the final requirements for Gemini Nano in Chrome.
	<b>Not yet supported:</b> <ul style="list-style-type: none"><li>• Chrome for Android</li><li>• Chrome for iOS</li><li>• Chrome for ChromeOS</li></ul>

## Setup

### Prerequisites

1. Acknowledge [Google's Generative AI Prohibited Uses Policy](#).
2. Download Chrome [Canary channel](#), and [confirm that your version](#) is equal or newer than 129.0.6639.0.
3. Check that your device meets the [requirements](#).
  - Don't skip this step, in particular make sure that you have **at least 22 GB of free storage space**.
  - If after the download the available storage space falls below 10 GB, the model will be **deleted again**.
  - Note that this is the same Gemini Nano model used by [the Prompt API](#), so if you already have the Prompt API set up, you don't need to worry about storage space.

## Enable Gemini Nano

 **Ignore this section if you already have set up the Prompt API!**

Follow these steps to enable Gemini Nano:

1. Open a new tab in Chrome, go to `chrome://flags/#optimization-guide-on-device-model`
2. Select **Enabled BypassPerfRequirement**
  - This bypass performance checks which might get in the way of having Gemini Nano downloaded on your device.
3. Relaunch Chrome.

## Confirm availability of Gemini Nano

1. Open DevTools and send `(await ai.assistant.capabilities()).available;` in the console.
2. If this returns **“readily”**, then you are all set.

## Enable the writer and the rewriter APIs

Follow these steps to enable the summarization API [flag](#) for local experimentation:

1. Open a new tab in Chrome, go to `chrome://flags/#writer-api-for-gemini-nano`
2. Select **Enabled**
3. Go to `chrome://flags/#rewriter-api-for-gemini-nano`
4. Select **Enabled**
5. Relaunch Chrome.

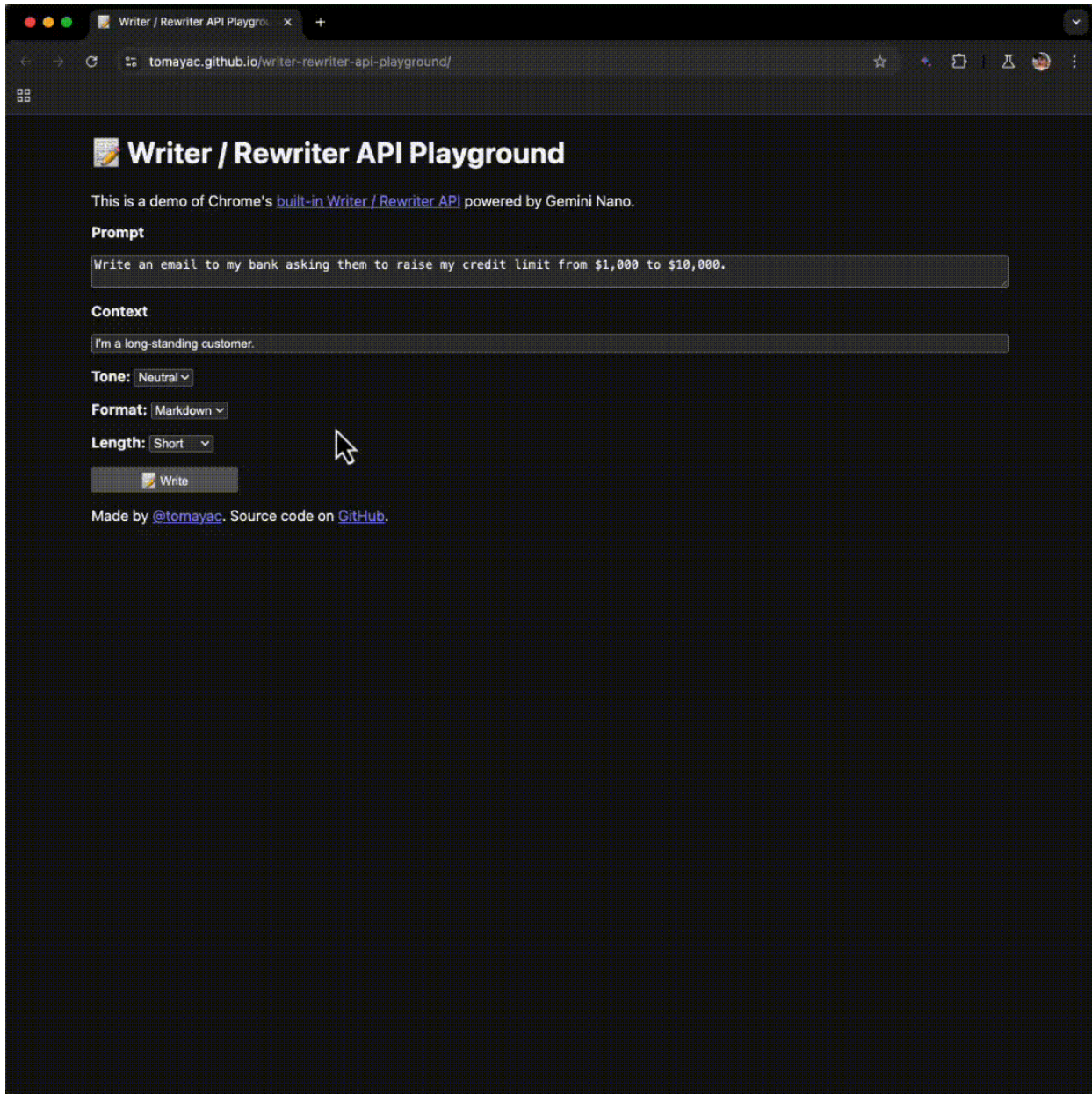
## Finalize the overall set up

1. Open DevTools and send `await ai.writer.create();` in the console.
  - Don't worry if the call fails, as the point is to force Chrome to schedule a model download.
2. Now try `(await ai.assistant.capabilities()).available;` in the console until the response changes to **“readily”**.
  - This may take about 3 to 5 minutes depending on your network connection, so let your Chrome instance run for a while.
  - If this still fails after waiting for a while, please see the [troubleshooting section](#).

## Demo

With the writer and rewrite APIs enabled, head over to this Chrome Dev Playground to try it out:


- <https://chrome.dev/web-ai-demos/writer-rewriter-api-playground/> ([code](#))



## API overview

### Sample code

Note: the current implementation isn't final and will evolve to support additional options and get closer to the design pattern described in the most recent [API explainer](#).

	<b>Explainer, explained.</b>  An <a href="#">explainer</a> is a document that describes a proposed web platform feature or collection of features. As work progresses, explainers facilitate discussion and, hopefully, consensus around the approach and feature design. Explainers are updated as design progresses.
---	--

## Writer API

JavaScript

```
// Non-streaming
const writer = await ai.writer.create();

const result = await writer.write(
  "A draft for an inquiry to my bank about how to enable wire transfers on my account."
);
```

JavaScript

```
// Streaming
const writer = await ai.writer.create();

const stream = await writer.writeStreaming(
  "A draft for an inquiry to my bank about how to enable wire transfers on my account."
);

for await (const chunk of stream) {
  composeTextbox.append(chunk);
}
```

JavaScript

```
// Shared context and per writing task context
const writer = await ai.writer.create({
  sharedContext: "This is for publishing on [popular website name], a business and employment-focused social media platform."
});
```

```

const stream = await writer.writeStreaming(
  "loving all this work on GenAI at Google! So much to learn and so many new
  things I can do!",
  { context: " The request comes from someone working at a startup providing an
  e-commerce CMS solution."}
);

for await (const chunk of stream) {
  composeTextbox.append(chunk);
}

```

JavaScript

```

// Reusing a writer
const writer = await ai.writer.create({ tone: "formal" });

const reviews = await Promise.all(
  Array.from(
    document.querySelectorAll("#reviews > .review"),
    (reviewEl) => writer.write(reviewEl.textContent)
  ),
);

```

JavaScript

```

// Aborting a writer
const controller = new AbortController();
stopButton.onclick = () => controller.abort();

const writer = await ai.writer.create({ signal: controller.signal });
await writer.write(document.body.textContent, { signal: controller.signal });

// Destroying a writer
writer.destroy();

```

## Rewriter API

JavaScript

```
// Non-streaming
const rewriter = await ai.rewriter.create(
  // `sharedContext` pending implementation:
  // sharedContext: "A review for the Flux Capacitor 3000 from TimeMachines Inc."
);

const result = await rewriter.rewrite(reviewEl.textContent, {
  context: "Avoid any toxic language and be as constructive as possible."
});
```

JavaScript

```
// Streaming
const rewriter = await ai.rewriter.create({
  // `sharedContext` pending implementation:
  // sharedContext: "A review for the Flux Capacitor 3000 from TimeMachines Inc."
});

const stream = await rewriter.rewriteStreaming(reviewEl.textContent, {
  // `context` pending implementation:
  // context: "Avoid any toxic language and be as constructive as possible."
});

for await (const chunk of stream) {
  composeTextbox.append(chunk);
}
```

JavaScript

```
// Reusing a rewriter
const rewriter = await ai.rewriter.create({
  // `sharedContext` pending implementation:
  // sharedContext: "A review for the Flux Capacitor 3000 from TimeMachines Inc."
});

const rewrittenReviews = await Promise.all(
  Array.from(
```

```

document.querySelectorAll("#reviews > .review"),
(reviewEl) => rewriter.rewrite(reviewEl.textContent, {
  // `context` pending implementation:
  // context: "Avoid any toxic language and be as constructive as
possible."
}))
),
);

```

JavaScript

```

// Aborting a rewriter
const controller = new AbortController();
stopButton.onclick = () => controller.abort();

const rewriter = await ai.rewriter.create({ signal: controller.signal });
await rewriter.rewrite(document.body.textContent, { signal: controller.signal
});

// Destroying a rewriter
rewriter.destroy();

```

## Appendix

### Full API surface

The full API surface is described below. See [Web IDL](#) for details on the language. The current implementation is a work in progress and does not support everything described below.

Unset

```

// Shared self.ai APIs

partial interface WindowOrWorkerGlobalScope {
  [Replaceable] readonly attribute AI ai;
};

[Exposed=(Window,Worker)]
interface AI {
  readonly attribute AIWriterFactory writer;

```



```

    readonly attribute AIREewriterFactory rewriter;
};

[Exposed=(Window,Worker)]
interface AICreateMonitor : EventTarget {
    attribute EventHandler ondownloadprogress;
};

callback AICreateMonitorCallback = undefined (AICreateMonitor monitor);

enum AICapabilityAvailability { "readily", "after-download", "no" };

// Writer

[Exposed=(Window,Worker)]
interface AIWriterFactory {
    Promise<AIWriter> create(optional AIWriterCreateOptions options = {});
    Promise<AIWriterCapabilities> capabilities();
};

[Exposed=(Window,Worker)]
interface AIWriter {
    Promise<DOMString> write(DOMString writingTask, optional AIWriterWriteOptions
options = {});
    ReadableStream writeStreaming(DOMString writingTask, optional
AIWriterWriteOptions options = {});

    readonly attribute DOMString sharedContext;
    readonly attribute AIWriterTone tone;
    readonly attribute AIWriterFormat format;
    readonly attribute AIWriterLength length;

    undefined destroy();
};

[Exposed=(Window,Worker)]
interface AIWriterCapabilities {
    readonly attribute AICapabilityAvailability available;

    AICapabilityAvailability supportsTone(AIWriterTone tone);
    AICapabilityAvailability supportsFormat(AIWriterFormat format);
    AICapabilityAvailability supportsLength(AIWriterLength length);

    AICapabilityAvailability supportsInputLanguage(DOMString languageTag);

```

```

};

dictionary AIWriterCreateOptions {
    AbortSignal signal;
    AICreateMonitorCallback monitor;

    DOMString sharedContext;
    AIWriterTone tone = "key-points",
    AIWriterFormat format = "markdown",
    AIWriterLength length = "short"
};

dictionary AIWriterWriteOptions {
    DOMString context;
    AbortSignal signal;
};

enum AIWriterTone { "formal", "neutral", "casual" };
enum AIWriterFormat { "plain-text", "markdown" };
enum AIWriterLength { "short", "medium", "long" };

// Rewriter

[Exposed=(Window,Worker)]
interface AIRewriterFactory {
    Promise<AIRewriter> create(optional AIRewriterCreateOptions options = {});
    Promise<AIRewriterCapabilities> capabilities();
};

[Exposed=(Window,Worker)]
interface AIRewriter {
    Promise<DOMString> rewrite(DOMString input, optional AIRewriterRewriteOptions
options = {});
    ReadableStream rewriteStreaming(DOMString input, optional
AIRewriterRewriteOptions options = {});

    readonly attribute DOMString sharedContext;
    readonly attribute AIRewriterTone tone;
    readonly attribute AIRewriterFormat format;
    readonly attribute AIRewriterLength length;

    undefined destroy();
};

```

```

[Exposed=(Window,Worker)]
interface AIREewriterCapabilities {
    readonly attribute AICapabilityAvailability available;

    AICapabilityAvailability supportsTone(AIREewriterTone tone);
    AICapabilityAvailability supportsFormat(AIREewriterFormat format);
    AICapabilityAvailability supportsLength(AIREewriterLength length);

    AICapabilityAvailability supportsInputLanguage(DOMString languageTag);
};

dictionary AIREewriterCreateOptions {
    AbortSignal signal;
    AICreateMonitorCallback monitor;

    DOMString sharedContext;
    AIREewriterTone tone = "as-is";
    AIREewriterFormat format = "as-is";
    AIREewriterLength length = "as-is";
};

dictionary AIREewriterRewriteOptions {
    DOMString context;
    AbortSignal signal;
};

enum AIREewriterTone { "as-is", "more-formal", "more-casual" };
enum AIREewriterFormat { "as-is", "plain-text", "markdown" };
enum AIREewriterLength { "as-is", "shorter", "longer" };

```

## General feedback

### Feedback form for quality or technical issues

If you experience quality or technical issues, consider [sharing details](#). Your reports will help us refine and improve our models, APIs, and components in the AI runtime layer, to ensure safety and responsible use.

- Handy shortlink: [goo.gle/chrome-ai-dev-preview-feedback-quality](https://goo.gle/chrome-ai-dev-preview-feedback-quality)

### Feedback about Chrome's behavior / implementation of the API

If you want to report bugs or other issues related to Chrome's behavior / implementation of the API, provide as many details as possible (e.g. repro steps) in a [public chromium bug report](#).

## Feedback about the APIs

If you want to report ergonomic issues or other problems related to the API itself, see if there is any related issue first and if not then file a public spec issue:

- [Writer and Rewriter APIs spec issues](#)

## Other feedback

For other questions or issues, reach out directly by sending an email to [the mailing list owners](mailto:chrome-ai-dev-preview+owners@chromium.org) ([chrome-ai-dev-preview+owners@chromium.org](mailto:chrome-ai-dev-preview+owners@chromium.org)). We'll do our best to be as responsive as possible or update existing documents when more appropriate.

## FAQ

### Participation in the Early Preview Program

#### Opt-out and unsubscribe

To opt-out from the Early Preview Program, simply send an email to:

- [chrome-ai-dev-preview+unsubscribe@chromium.org](mailto:chrome-ai-dev-preview+unsubscribe@chromium.org).

#### Opt-in

If you know someone who would like to join the program, ask them to fill out [this form](#) and that they communicate their eagerness to provide feedback when answering the last question of the survey!

## Other updates

Links to all previous updates and surveys we've sent can be found in [The Context Index](#) also available via [goo.gle/chrome-ai-dev-preview-index](https://goo.gle/chrome-ai-dev-preview-index)

## Changelog

Date	Changes
Aug 27, 2024	<ul style="list-style-type: none"><li>• First version.</li></ul>
Aug 27, 2024	<ul style="list-style-type: none"><li>• Added clarification to differentiate the Writer / Rewriter API from the Prompt API.</li></ul>
Aug 29, 2024	<ul style="list-style-type: none"><li>• Added code snippet to show that sharedContext and context are also supported by the Writer API.</li><li>• Fixed an issue with the sample code:<ul style="list-style-type: none"><li>◦ before : for (const chunk of stream) { ...</li><li>◦ after: for await (const chunk of stream) { ...</li></ul></li></ul>
Sep 4, 2024	<ul style="list-style-type: none"><li>• Commented out `sharedContext` and `context` in Rewriter API, as they're both pending implementations.</li></ul>

