

Built-in AI Early Preview Program

The Prompt API with multimodal capabilities

Authors

Kenji Baheux
Alexandra Klepper
François Beaufort

Contact

See [Feedback](#)

Last-updated

May 15, 2025
See [changelog](#).

Intro

Thanks for participating in our Early Preview Program for built-in AI capabilities (Talks at Google I/O 2025: [web](#), [Chrome Extensions](#)). 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.

The [Prompt API](#) was initially introduced in 2024, providing text-based prompting capabilities. Starting in Chrome 138, we are rolling out an initial release of this text-only Prompt API scoped to Chrome Extensions (please note that the timing / milestone may shift slightly).

This document, however, focuses on the exciting **new multimodal capabilities for the Prompt API**. While the full public availability of multimodal capabilities on the web will begin in an upcoming Origin Trial, you can enable and test them for local experimentation today, as explained in this document.

Our multimodal journey begins with **using audio or images as inputs with the Prompt API**, to receive text outputs. Your feedback and explorations will inform the future of this API and improvements to Gemini Nano. It may even result in dedicated task APIs (such as APIs for audio transcription or image description), ensuring we meet your needs and the needs of your users.

Prompt API with multimodal capabilities

Purpose

The Prompt API with multimodal capabilities is provided for local experimentation. Prompt a multimodal version of [Gemini Nano](#) to describe the contents of an image, transcribe an audio file, or so much more.

Our goals

We hope to hear your feedback on the following aspects:

1. The **quality of the output**, 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](#).


Availability

The updated Prompt API is available, behind an experimental flag, from Chrome 138+ on desktop platforms.

- You'll need Version **138.0.7190.0** or above
- We recommend using [Chrome Canary](#).

Requirements

Built-in AI is **currently focused on desktop platforms**. In addition, you must meet the [hardware requirements for Gemini Nano](#).

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

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 138.0.7190.0.
3. Check that your device meets the [requirements](#).
 - Don't skip this step. You must have **at least 22 GB of free storage space** (the model only uses a small fraction of this storage space, the margin is to avoid putting a disk into storage pressure).
 - If the available storage space falls below 10 GB after the download, **the model will be deleted**.

Enable the Prompt API

1. `chrome://flags/#optimization-guide-on-device-model` and select **Enabled BypassPerfRequirement**.

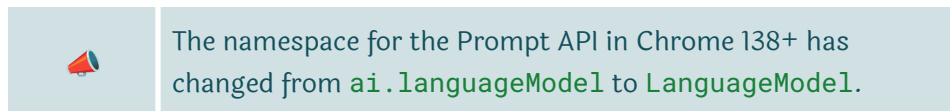
2. Enable the new multimodal input flags
 - o <chrome://flags/#prompt-api-for-gemini-nano-multimodal-input> and select **Enabled**.
3. Click **Relaunch** or restart Chrome.

Confirm Gemini Nano is downloaded

To confirm Gemini Nano has downloaded and works as intended:

1. Open DevTools Console.
2. Type `await LanguageModel.availability({expectedInputs:[{type:'image'}},{type:'audio'}}]);` in the console. This should return **available**.

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, please see the [troubleshooting section](#).



Demo

With the Prompt API enabled, head over to the following demos.

Web application for images

Visit the [Prompt API with image input demo](#) and attempt to draw the world-famous Mona Lisa. Then ask Gemini Nano to evaluate your creation. You will receive a text response.

Web application for audio

Try recording a 5 second snippet or uploading a file to transcribe in the [MediaRecorder + Audio Prompt API](#) demo.

Extension

Code: [GitHub Extension](#)

This Chrome extension serves as a practical demonstration of the multimodal capabilities. It allows a user to select an image, triggering the extension to use the Prompt API to generate a description of that image, and then have it speak the generated description aloud using the [WebSpeech API](#). A

crucial aspect of this demo is that analyzing the image and generating its description, is performed entirely on the user's device.

To install, check out GitHub repo main branch locally, load the **src** folder as an unpacked extension in **chrome://extensions**, and right-click any image when browsing to hear what this image is about.

For more on how to work with unpacked Extensions, [read the documentation](#).

API overview



TypeScript type definitions for the Built-in AI APIs are available as part of the DefinitelyTyped APIs, in the [@types/dom-chromium-ai package](#). Track [updates to this package](#).

We are using the Prompt API's latest design as the baseline for the multimodal enhanced Prompt APIs. Refer to the [Explainer](#) for API usage and syntax.

Create a session

JavaScript

```
const session = await LanguageModel.create({
  expectedInputs: [{type: 'image'}, {type: 'audio'}]
});
```

expectedInputs tells the API which types of input will be used when prompting, so the browser can select the right model to create. Currently accepted input types are image and audio.

Read our [prompt best practices](#) to understand how to optimize your output from Gemini Nano in Chrome.

Add an image to your prompt

You can prompt the model with an image by adding an object to your prompt. For example:

JavaScript

```
const session = await LanguageModel.create({expectedInputs: [{type: 'image'}]});
```

```
const response = await fetch('/my/directory/image.png');
const blob = await response.blob();
const imageBitmap = await createImageBitmap(blob);

const result = await session.prompt([
  'Analyze the provided image. Generate a detailed and engaging product description suitable for an e-commerce website',
  { type: "image", content: imageBitmap }
]);
```

Review the [image limitations](#) to learn what file types can be used and other restrictions. The model outputs a text response.

Add audio to your prompt

Similar to images, update your prompt to include an audio object. For example, using an audio buffer:

```
JavaScript
const session = await LanguageModel.create({expectedInputs:[{type:'audio'}]});
const audio = await fetch('/my/directory/audio.mp3');
const audioCtx = new AudioContext();
const arrayBuffer = await audio.arrayBuffer();
const audioBuffer = await audioCtx.decodeAudioData(arrayBuffer);
const response = await session.prompt([
  { type: "audio", content: audioBuffer },
  "Transcribe this audio:"
]);
```

Example code using audio blob:

```
JavaScript
const session = await LanguageModel.create({expectedInputs:[{type:'audio'}]});
const audio = await fetch('/my/directory/audio.mp3');
const content = await audio.blob();
const response = await session.prompt([
  { type: "audio", content: content },
  "Transcribe this audio:"
]);
```

Review the [audio limitations](#) to learn what file types can be used and other limitations. You can [live capture audio](#) to add to the prompt.

Stream output

The example prompts create non-streaming output. You can stream the output by replacing `session.prompt()` with `session.promptStreaming()`.

Learn more about [streaming from LLMs](#).

End a session

Call `destroy()` to free resources if you no longer need a session. Learn more about [session termination](#).

Example use cases

Describe an image on the page.

```
JavaScript
images = document.getElementsByTagName('img');
languageModel = await LanguageModel.create({expectedInputs:[{type:'image'}]});
output = await languageModel.prompt(['describe this image', {type: 'image',
content: images[0]}]);
```

Describe an image from a file handle.

```
JavaScript
languageModel = await LanguageModel.create({expectedInputs:[{type:'image'}]});
fileSystemFileHandle = fileSystemHandle;
file = await fileSystemFileHandle.getFile();
output = await languageModel.prompt(['describe this image', {type: 'image',
content: createImageBitmap(file)}]);
```

Transcribe an audio file from a URL resource.

```
JavaScript
// Be careful of access controls over remote resources (see the Errors section)
audioCtx = new AudioContext();
audio = await
fetch('https://cdn.glitch.global/e5522a51-ab87-443b-80b5-2ce583856d56/en_jeremy
t_2.wav?v=1740437231265')
buffer = await audioCtx.decodeAudioData(await audio.arrayBuffer());
```

```
languageModel = await LanguageModel.create({expectedInputs:[{type:'audio'}]});
response = await languageModel.prompt(['Transcribe this audio.', {type:
'audio', content: buffer}]);
```

Stream the response for describing an image on the page.

```
JavaScript
images = document.getElementsByTagName('img');
languageModel = await LanguageModel.create({expectedInputs:[{type:'image'}]});
stream = await languageModel.promptStreaming(
  ['describe this image', {type: 'image', content: images[0]}]);
result = '';
for await (chunk of stream) { result += chunk; console.log("chunk: " + chunk);
}
console.log("result: " + result);
```

(Optional) Turn on additional API flags

Enable the following flags for other text-to-text APIs.

- Summarizer API: <chrome://flags/#summarization-api-for-gemini-nano>
- Translator API: <chrome://flags/#translation-api>
- Language Detector API: <chrome://flags/#language-detection-api>
- Writer API: <chrome://flags/#writer-api-for-gemini-nano>
- Rewriter API: <chrome://flags/#rewriter-api-for-gemini-nano>

Additionally, disabling <chrome://flags/#text-safety-classifier> may help these APIs run faster, or with fewer hurdles.

Prompt 101

To write the best prompts, we recommend reading the following:

- [People + AI guidebook](#) which talks about the *if*, *when*, and *how* to use AI (including UX examples).
- [Prompt design strategies](#) which gives concrete guidance on how to best prompt an LLM.

In addition, here are some recommendations for Gemini Nano in Chrome. These suggestions were shared with the first iteration of the Prompt API.

DOs	DONTs
<p>Include examples to guide your prompt. One example, or one shot prompting, is better than no examples. Few shot prompting is best in guiding the model to return your intended results.</p>	<p>Avoid use cases with right or wrong answers. The model might be too small to answer correctly knowledge questions, and may also struggle with tasks that depend on getting the perfect answer. Design your feature or UX with these imperfections in mind.</p>
<p>Add rules. You can improve the quality of the output by adding rules in your prompt.</p> <p>Examples: “Respond in a friendly tone”, “Use the category ‘ambiguous’ if it’s unclear”, “Do not provide any explanation.”</p>	<p>Avoid use cases that bypass the user. LLMs may not always have a satisfying response. So, it’s better to position your AI feature as a tool to support the user in their tasks (e.g. generating a list of potential keywords that the user can quickly review and adjust).</p>
<p>Add a persona. This can help the model return something that’s more aligned to your needs.</p> <p>Example: “You just bought a product, and are writing a review for said product. Please rephrase [...]”</p>	<p>Avoid customizing the parameters. While it’s possible to change the parameters (such as temperature and topK), we strongly recommend that you keep the default values unless you’ve run out of ideas with the prompt or you need the model to behave differently.</p>
<p>(Non-English)</p> <p>Specify the output language. If you prompt the model in English but need a non-English response, specify the output language in the prompt.</p> <p>Example: “Please rephrase the following sentence in Spanish. Hola, cómo estás”.</p>	
<p>Use AI capabilities responsibly. Generative AI models can help you or your users be more productive, enhance creativity and learning. We expect you to use and engage with this technology in accordance with Google’s Generative AI Prohibited Uses Policy.</p>	

Errors

- If you use CORS data, you must set the [crossorigin attribute](#). Without this attribute, prompts using cross-origin data are rejected with a `SecurityError DOMException`.
- Invalid content types, such as `{ type: "image", content: new ArrayBuffer(1024) }`, are rejected with a ``TypeError``.

Caveats and limitations

This version of the model has the following limitations.

Audio limitations

- The audio encoder is trained for transcription, but trying other use cases is encouraged. [Report](#) back quality issues and desired use cases so we can prioritize accordingly.
- Audio content can be any of the following:
 - AudioBuffer or Blob in .wav, .mp3, .ogg, or .m4a format.
 - Mono sound (Stereo is also supported but will be automatically down-mixed to Mono).
 - 30 seconds or less. If more than 30 seconds are provided, only the first 30 seconds are processed.
- You can add multiple audio files, but the total time across all files will be capped at 30s.

Image limitations

- **Automatic 768x768px pre-processing:** The model automatically resizes and squares all submitted images to 768x768 pixels. Manual pre-resizing to this specific dimension is **unnecessary and potentially detrimental**.
 - Future model iterations might, for example, use different input resolutions or pre-processing logic, which could make your client-side pre-sizing to 768x768px suboptimal.
- **Source image recommendations for optimal quality:**
 - **Resolution:** Aim for original images with dimensions of at least 768 pixels (e.g., 768x768 pixels, 1024x768 pixels). For images larger than 768x768 pixels, ensure critical details will remain distinct after potential downscaling by the model. *You can preview the model's likely input by temporarily resizing a local copy to 768x768 pixels.*
 - **Aspect ratio:** While perfectly square (1:1) images are not mandatory, those with extreme aspect ratios (e.g., very wide banners or tall, narrow images) may suffer from significant distortion or undesirable cropping when the model adapts them to the 768x768 pixels square. Aim for aspect ratios *relatively* close to 1:1.
 - **Avoid extraneous content:** Submit images that primarily feature the subject or area of interest. For instance, instead of sending an entire uncropped screenshot, crop to the relevant portion. This helps the model concentrate its 768x768 pixels processing power on the details that matter most for your use case.
- **Supported image sources:** Image content can be any of the following:
 - The image content can be either an ImageData, ImageBitmap, OffscreenCanvas, HTMLImageElement, or HTMLCanvasElement.
 - Raw bytes through [SourceBuffer](#), [ArrayBuffer](#), or typed arrays.
- **Requirement for live image resources:** The image must be fully decoded before it's used in a prompt. One way to ensure this happens is to use `img.decode()`. For example:

JavaScript

```
const img = document.querySelector('img');
const session = await LanguageModel.create({expectedInputs:[{type:'image'}]});
await img.decode();
const promptResult = await session.prompt([
  'Generate an Alt attribute for this following image to improve accessibility.',
  { type:'image', content: img }
]);
```

Token limitations

You can estimate the token length of the input prompt. For example, a session created with `LanguageModel.create()` returns an `AILanguageModel` object:

None

```
{inputQuota: 6144
inputUsage: 0
onquotaoverflow: null
temperature: 1
topK: 3}
```

Use the properties `inputQuota` and `inputUsage` to count the prompt token length. The default context token size is 6144.

You may send a prompt that causes the context window to overflow. In such cases, the initial portions of the conversation are removed, one prompt and response pair at a time, until enough tokens are available to process the new prompt. The exception is the [system prompt](#), which is never removed.

If it's impossible to remove enough tokens from conversation history to process a new prompt, `prompt()` and `promptStreaming()` fail with a `QuotaExceededError` DOMException and nothing will be removed.

Such overflows can be detected by listening for the `"quotaoverflow"` event on the session:

```
JavaScript
session.addEventListener("quotaoverflow", () => {
  console.log("We've gone past the quota, and some inputs will be dropped!");
});
```

Appendix

General feedback

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

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 API

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](#).

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). 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, send an email to chrome-ai-dev-preview+unsubscribe@chromium.org.

Opt-in

If you know someone who would like to join the program, ask them to complete the sign-up form at goo.gl/chrome-ai-dev-preview-join.

Other updates

Links to all previous updates and surveys we've sent can be found in [The Context Index](#).

Changelog

Date	Changes
May 15, 2025	<ul style="list-style-type: none">• First version.
📅 Date	<ul style="list-style-type: none">•