Built-in AI Early Preview Program

Welcome and about the Prompt API

Authors: Kenji Baheux, Thomas Steiner, Alexandra Klepper

Contact: See this section

Last-updated: Nov 6, 2024
See changelog.

Latest news 📯

- Oct 15, 2024 : **Breaking changes** in Chrome Canary 131.0.6776.0+ **for the Prompt API**; tThis document has been updated with the latest.
- Aug 27, 2024 : **Breaking changes** in Chrome Canary 129.0.6667.0+ **for the Prompt API**; this document has been updated with the latest.
- Summarization API and Language detection API are now available for experimentation.
- Check goo.gle/chrome-ai-dev-preview-index for the full list of updates!

Intro

Welcome, and thank you for participating in our Early Preview Program for built-in AI capabilities (article, talk at Google I/O 2024). Your involvement is invaluable as we explore opportunities to improve or augment web experiences with AI!

The Built-in AI Early Preview Program has several goals:

- **Listening:** We're eager to hear your feedback on early-stage APIs. Help us better understand the problems you're trying to solve. This will ensure that we design the APIs you need, in the right way.
- **Exploring:** We want to facilitate the discovery of promising applications for built-in AI, which will directly inform our roadmap and prioritization discussions. To this extent, we'll provide access to exploratory APIs such as the Prompt API.
- **Supporting:** Your insights and interest will inform discussions with other browser vendors, as we work towards common standards for AI integration in web browsers.

📣	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.

In this first update, we're excited to provide details about the upcoming exploratory Prompt API, designed to facilitate the discovery of AI use cases through local prototyping. More concretely, this API will let you interact with Gemini Nano, on-device, directly in your development environment.

> 📝 **Exploratory APIs** are intended for local prototyping only. With these exploratory APIs, we intend to solicit feedback, confirm assumptions, and determine what task APIs (such as a Translation API) we build in the future. *Consequently, the exploratory APIs **may never launch**.*

As we explore the possibilities of this technology together, it's critical that we prioritize responsible AI development. To help guide our efforts, take a moment to review Google's Generative AI Prohibited Uses Policy. This policy outlines some key considerations for ethical and safe deployment of AI.

Let's learn and build together ✨!

Prompt API

Purpose

The Prompt API is provided for local experimentation, to facilitate the discovery of use cases for built-in AI. With this API, you can send natural language instructions to an instance of Gemini Nano in Chrome.

While the Prompt API offers the most flexibility, it won't necessarily deliver the best results and may not deliver sufficient quality in some cases. That's why we believe that task-specific APIs (such as a Translation API) paired with fine tuning or expert models, will deliver significantly better results. Our hope for the Prompt API is that it helps accelerate the discovery of compelling use cases to inform a roadmap of task-specific APIs.

Timing

The Prompt API is available, behind an experimental flag, from Chrome 127+ for desktop.

- The experimental flag **is available as of** `May 31, 2024 6:09 PM` **.**
- You'll need **Version 128.0.6545.0 or above**.
- We recommend using Chrome Canary or Chrome Dev 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. *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 **deleted again**.*		
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.
	We are interested in feedback about performance aspects to further improve the user experience, and adjust the requirements.
	Not yet supported: • Chrome for Android • Chrome for iOS • Chrome for ChromeOS

Setup

Prerequisites

1. Acknowledge Google's Generative AI Prohibited Uses Policy.
2. Download Chrome Dev channel (or Canary channel), and confirm that your version is equal or newer than 128.0.6545.0.
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 some operating systems may report the actual free disk space differently, for example, by including or not disk space that's occupied by the trash bin. On macOS, use Disk Utility to get the representative free disk space.

Enable Gemini Nano and the Prompt API

Follow these steps to enable Gemini Nano and the Prompt API flags for local experimentation:

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. Go to `chrome://flags/#prompt-api-for-gemini-nano`
4. Select `Enabled`
5. Relaunch Chrome.

Confirm availability of Gemini Nano

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

If this fails, continue as follows:

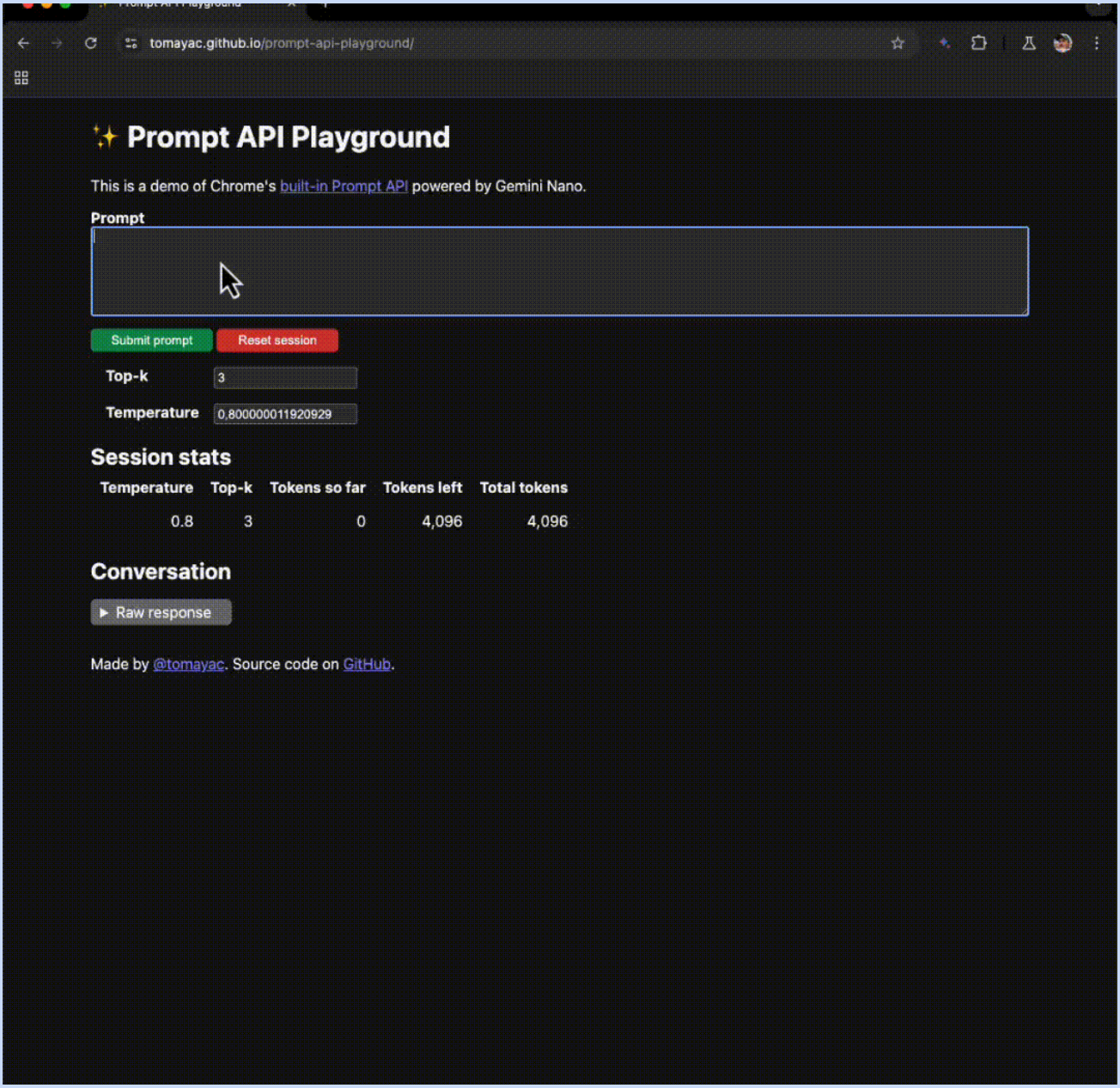
1. 📣🆕Force Chrome to recognize that you want to use this API. To do so, open DevTools and send `await ai.languageModel.create();` in the console. This will likely fail but it's intended.
2. Relaunch Chrome.
3. Open a new tab in Chrome, go to `chrome://components`
4. Confirm that Gemini Nano is either available or is being downloaded
    - You'll want to see the `Optimization Guide On Device Model` present with a **version greater or equal to `2024.5.21.1031`.**
    - If there is no version listed, click on `Check for update` to force the download.
5. Once the model has downloaded and has reached a version greater than shown above, open DevTools and send `(await ai.languageModel.capabilities()).available;` in the console. If this returns **`"readily"`**, then you are all set.
    - Otherwise, relaunch, wait for a little while, and try again from step 1.

If this still fails, please see the troubleshooting section.

Demo

With the Prompt API enabled, head over to this Chrome Dev Playground to try it out:

- https://chrome.dev/web-ai-demos/prompt-api-playground/ (code)



The Prompt API in action

API overview

Sample code

At once version

```
JavaScript
// Start by checking if it's possible to create a session based on the
availability of the model, and the characteristics of the device.
const {available, defaultTemperature, defaultTopK, maxTopK } = await
ai.languageModel.capabilities();

if (available !== "no") {
  const session = await ai.languageModel.create();

  // Prompt the model and wait for the whole result to come back.
  const result = await session.prompt("Write me a poem");
  console.log(result);
}
```

Streaming version

```
JavaScript
const {available, defaultTemperature, defaultTopK, maxTopK } = await
ai.languageModel.capabilities();

if (available !== "no") {
  const session = await ai.languageModel.create();

  // Prompt the model and stream the result:
  const stream = session.promptStreaming("Write me an extra-long poem");
  for await (const chunk of stream) {
    console.log(chunk);
  }
}
```

Tracking model download progress

```JavaScript
const session = await ai.languageModel.create({
  monitor(m) {
    m.addEventListener("downloadprogress", e => {
      console.log(`Downloaded ${e.loaded} of ${e.total} bytes.`);
    });
  }
});
```

Session persistence

Each session keeps track of the context of the conversation.

```JavaScript
const session = await ai.languageModel.create({
  systemPrompt: "You are a friendly, helpful assistant specialized in clothing choices."
});

const result = await session.prompt(`
  What should I wear today? It's sunny and I'm unsure between a t-shirt and a polo.
`);

console.log(result);

const result2 = await session.prompt(`
  That sounds great, but oh no, it's actually going to rain! New advice??
`);
```

Session cloning

To preserve resources, you can clone an existing session. The conversation context will be reset, but the initial prompt or the system prompts will remain intact.

```JavaScript
const clonedSession = await session.clone();
```

Session options

Each session can be customized with topK and temperature. The default values for these parameters are returned from `ai.languageModel.capabilities()`.

```
JavaScript
const capabilities = await ai.languageModel.capabilities();
// Initializing a new session must either specify both topK and temperature, or
neither of them.
const slightlyHighTemperatureSession = await ai.languageModel.create({
  temperature: Math.max(capabilities.defaultTemperature * 1.2, 1.0),
  topK: capabilities.defaultTopK,
});
```

System prompts

Give the language model some context.

```
JavaScript
const session = await ai.languageModel.create({
  systemPrompt: "Pretend to be an eloquent hamster."
});
await session.prompt('Do you like nuts?');
// ' As a hamster of unparalleled linguistic ability, I find myself quite adept
at responding to the question of whether or not I enjoy the consumption of
delectable nuts. Nutty delight indeed!'
```

Session information

A given language model session will have a maximum number of tokens it can process. Developers can check their current usage and progress toward that limit by using the following properties on the session object:

```
JavaScript
console.log(`${session.tokensSoFar}/${session.maxTokens} (${session.tokensLeft}
left)`);
```

Important caveat At the moment, there is a per prompt limit of 1024 tokens, and the session can retain the last 4096 tokens. We are discussing ways to simplify this so that you will be able to use the 4096 tokens as you wish (e.g. in one prompt, or over several prompts without limits on each prompt). Stay tuned.

Terminating a session

Call `destroy()` to free resources if you no longer need a session. When a session is destroyed, it can no longer be used, and any ongoing execution will be aborted. You may want to keep the session around if you intend to prompt the model often since creating a session can take some time.

```
JavaScript
await session.prompt(`
  You are a friendly, helpful assistant specialized in clothing choices.
`);

session.destroy();

// The promise will be rejected with an error explaining that the session is
destroyed.
await session.prompt(`
  What should I wear today? It's sunny and I'm unsure between a t-shirt and a
polo.
`);
```

Exceptions

The Prompt API may receive errors from the AI runtime. See this section for a list of possible errors, and how they are mapped into `DOMException`s.

Caveats

The `(await ai.languageModel.capabilities()).available`'s "after-download" state

The **`"after-download"`** state and behavior is not supported. The API doesn't trigger a download of the model. Instead, Chrome will trigger the download, either as part of the chrome://flags state change, or because of another on-device AI feature.

Streaming

Currently, `promptStreaming()` returns a `ReadableStream` whose chunks successively build on each other.

For example, the following code logs a sequence, such as "Hello," "Hello world," "Hello world I am," "Hello world I am an AI."

```javascript
for await (const chunk of stream) {
  console.log(chunk);
}
```

This is not the desired behavior. We intend to align with other streaming APIs on the platform, where the chunks are successive pieces of a single long stream. This means the output would be a sequence like "Hello", " world", " I am", " an AI".

For now, to achieve the intended behavior, you can implement the following, which will work with both behaviors, the standard and the non-standard behavior:

```javascript
let result = '';
let previousChunk = '';

for await (const chunk of stream) {
  const newChunk = chunk.startsWith(previousChunk)
      ? chunk.slice(previousChunk.length) : chunk;
  console.log(newChunk);
  result += newChunk;
  previousChunk = chunk;
}
console.log(result);
```

Incognito and guest mode

Our implementation of the Prompt API doesn't currently support the Incognito mode, nor the guest mode. This is due to a dependency in the AI runtime layer, and not intended to be a permanent limitation.

Enterprise

This API will **not** work if GenAILocalFoundationalModelSettings is set to "Do **not** download model".

- You can check this setting from `chrome://policy`.

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 our recommendations for Gemini Nano in Chrome.

DOs	DONTs
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.	**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.
Add rules. You can improve the quality of the output by adding rules in your prompt. Examples: "Respond in a friendly tone", "Use the category 'ambiguous' if it's unclear", "Do not provide any explanation."	**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).
Add a persona. This can help the model return something that's more aligned to your needs. Example: "You just bought a product, and are writing a review for said product. Please rephrase [...]"	**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.
(Non-English) **Specify the output language.** If you prompt the model in English but need a non-English response, specify the output language in the prompt. Example: "Please rephrase the following sentence in Spanish. Hola, cómo estás".	
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.	

Emulating stop sequences

The model doesn't support stop sequences natively, but you can emulate them as follows.

```
Unset
const abortController = new AbortController();
const signal = abortController.signal;

const STOP_SEQUENCES = ['world'];

const languageModel = await self.ai.languageModel.create();
const stream = await languageModel.promptStreaming('Say "Hello, world!"', {
  signal,
});

let previousLength = 0;
streamingLoop: for await (const chunk of stream) {
  const newContent = chunk.slice(previousLength);
  console.log(`Chunk: "${newContent}"`);
  for (const stopSequence of STOP_SEQUENCES) {
    if (newContent.toLowerCase().includes(stopSequence.toLowerCase())) {
      console.log(
        `Stop sequence "${stopSequence}" found in chunk "${newContent}".
Aborting.`
      );
      abortController.abort();
      break streamingLoop;
    }
  }
  document.body.insertAdjacentText('beforeEnd', newContent);
  previousLength = chunk.length;
}
```

Share your feedback

Surveys

We'll send surveys on an ongoing basis to get a sense of how the APIs and the task-specific APIs approach is working, to collect signals about popular use cases, issues, etc.

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`

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

If you want to report bugs or other issues related to Chrome's behavior / implementation of the Prompt 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 one of the built-in AI APIs, see if there is any related issue first and if not then file a public spec issue:

- Prompt API spec issues
- Translation API spec issues

Other feedback

For other questions or issues, reach out directly by sending an email to the mailing list owners (`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 (such as adding to the FAQ section).

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.

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!

Non-English language support

The latest model version of Gemini Nano has only thoroughly been tested for quality (e.g., its responses should be factually correct and well written) and security (it should not tell you dangerous things, like how to build a bomb) with English. We're working on lifting this limitation soon. There will be a flag that allows you to bypass the limitation for local testing, but it's not working reliably yet, tracked in this CL.

Compatibility issue on macOS

The use of Rosetta to run the x64 version of Chromium on ARM is neither tested nor maintained, and unexpected behavior will likely result. Please check that all tools that spawn Chromium are ARM-native.

Output quality

The current implementation of the Prompt API is primarily for experimentation and may not reflect the final output quality when integrated with task APIs we intend to ship.

That said, if you see the model generates harmful content or problematic responses, we encourage you to share feedback on output quality. Your reports are invaluable in helping us refine and improve our models, APIs, and components in the AI runtime layer, to ensure safety and responsible use.

What happens when the number of tokens in the prompt exceeds the context window?

The current design only considers the last N tokens in a given input. Consequently, providing too much text in the prompt, may result in the model ignoring the beginning portion of the prompt. For example, if the prompt begins with "Translate the following text to English:...", preceded by thousands of lines of text, the model may not receive the beginning of the prompt and thus fail to provide a translation.

Some prompts stop when using stream execution.

Let us know if you can reproduce the issue: prompt, session options, details about the device, etc. In the meantime, restart Chrome and try again.

Last resort troubleshooting

Alternative steps

Some participants have reported that the following steps helped them get the component to show up:

- Try changing the `chrome://flags/#optimization-guide-on-device-model` to `Enabled` instead of `Enabled BypassPerfRequirement`, then retry the setup steps.

Model download delay

The browser may not start downloading the model right away. If your computer fulfills all the requirements and you don't see the model download start on `chrome://components` after calling `ai.languageModel.create()`, and *Optimization Guide On Device Model* shows *version 0.0.0.0 / New*, leave the browser open for a few minutes to wait for the scheduler to start the download.

Debug logs

If everything fails:

1. Open a new tab
2. Go to `chrome://gpu`
3. Download the gpu report

4. Go to `chrome://histograms/#OptimizationGuide.ModelExecution.OnDeviceModelInstallCriteria.AtRegistration.DiskSpace`
    - If you see records for 0, it means that your device doesn't have enough storage space for the model. Ensure that you have at least 22 GB on the disk with your user profile, and retry the setup steps. If you are still stuck, continue with the other steps below.

    On qualifying systems, the histogram should look similar to the following example:

```
- Histogram:
OptimizationGuide.ModelExecution.OnDeviceModelInstallCriteria.AtRegistration.DiskSpace recorded 1 samples, mean = 1.0 (flags = 0x41)
[#]
```

5. Go to `chrome://histograms/#OptimizationGuide` and download the histograms report
6. Share both reports with the Early Preview Program coordinators.

Crash logs

If you encounter a crash error message such as "`the model process crashed too many times for this version.`", then we'll need a crash ID to investigate the root causes.

1. Ensure that you have enabled crash reporting.
2. Reproduce the issue.
3. Go to `chrome://crashes`
4. Find the most recent crash
    a. Hit `Send now` if necessary.
    b. Then wait and reload until the `Status` line changes from `Not uploaded` to `Uploaded`. Note that this could take a while.
5. Copy the ID next to the `Uploaded Crash Report ID` line.
6. Share the ID with the Early Preview Program coordinators.

In the meantime, you'll need to wait for a new Chrome version to get another chance of trying the Prompt API.

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 AILanguageModelFactory languageModel;
};

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

callback AICreateMonitorCallback = undefined (AICreateMonitor monitor);

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

// LanguageModel

[Exposed=(Window,Worker)]
interface AILanguageModelFactory {
  Promise<AILanguageModel> create(optional AILanguageModelCreateOptions options
= {});
  Promise<AILanguageModelCapabilities> capabilities();
};

[Exposed=(Window,Worker)]
interface AILanguageModel : EventTarget {
  Promise<DOMString> prompt(DOMString input, optional
AILanguageModelPromptOptions options = {});
  ReadableStream promptStreaming(DOMString input, optional
AILanguageModelPromptOptions options = {});

  Promise<unsigned long long> countPromptTokens(DOMString input, optional
AILanguageModelPromptOptions options = {});
  readonly attribute unsigned long long maxTokens;
  readonly attribute unsigned long long tokensSoFar;
  readonly attribute unsigned long long tokensLeft;
  readonly attribute unsigned long topK;
  readonly attribute float temperature;
```

```
  Promise<AILanguageModel> clone();
  undefined destroy();
};

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

  // Always null if available === "no"
  readonly attribute unsigned long? defaultTopK;
  readonly attribute unsigned long? maxTopK;
  readonly attribute float? defaultTemperature;
};

dictionary AILanguageModelCreateOptions {
  AbortSignal signal;
  AICreateMonitorCallback monitor;

  DOMString systemPrompt;
  sequence<AILanguageModelPrompt> initialPrompts;
  [EnforceRange] unsigned long topK;
  float temperature;
};

dictionary AILanguageModelPrompt {
  AILanguageModelPromptRole role;
  DOMString content;
};

dictionary AILanguageModelPromptOptions {
  AbortSignal signal;
};

enum AILanguageModelPromptRole { "system", "user", "assistant" };
```

Full list of exceptions

Methods	DOMExceptions	Error messages	Comments
All methods	**InvalidStateError**	The execution context is not valid.	The JS context is invalid (e.g. a detached iframe). **Remedy:** ensure the API is called from a valid JS context.

create	**OperationError**	Model execution service is not available.	**Remedy:** try again, possibly after relaunching Chrome.
When streaming a response	**UnknownError**	An unknown error occurred: <error code>	**Remedy:** retry, possibly after relaunching Chrome. Report a technical issue if you are stuck and/or can easily reproduce the problem.
	NotSupportedError	The request was invalid.	
	UnknownError	Some other generic failure occurred.	
	NotReadableError	The response was disabled.	
	AbortError	The request was canceled.	
prompt, promptStreaming	**InvalidStateError**	The model execution session has been destroyed.	This happens when calling prompt or promptStreaming after the session has been destroyed. **Remedy:** create a new session.
create	**NotSupportedError**	Initializing a new session must either specify both topK and temperature, or neither of them.	This happens when the optional parameters are partially specified when calling create. **Remedy:** You need to specify both topK and temperature parameters, or none at all, when calling create. Use the values from defaultTextSessionOptions if you only want to change the value of one parameter.
	InvalidStateError	The session cannot be created.	If capabilities' available returned "readily" then this should not happen. **Remedy:** retry, possibly after relaunching Chrome. Report a technical issue if you are stuck and/or can easily reproduce the problem.

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`

Changelog

Date	Changes

May 29, 2024	• Added details about the lack of support for Incognito. • Added a note about GenAI Enterprise policy.
May 30, 2024	• Added a note about storage footprint. • Added a note about Worker support being limited to Dedicated Workers.
May 31, 2024	• Added a note about setting the system language to English (US) in the setup section (temporary requirement). • Minor editorial changes to the flags URLs, the name of the Bypass Perf Requirements option, and the name of the Prompt API flag. • Changed the target date to May 31st to reflect the latest estimate, although it's still May 30th in Baker Islands as of the time of writing. • Changed "yes" to "readily" to reflect the actual implementation. • Added an explanation for BypassPerfRequirements.
Jun 1, 2024	• In the setup steps, clarified that the version can be *equal or* newer than 128.0.6545.0.
Jun 3, 2024	• Added chrome://histograms step in the troubleshooting guide. • Added a reminder that prerequisites must be met, in particular the 22 GB storage space requirement.
Jun 6, 2024	• Added a note about *guest mode* not being supported. • Bumped the minimum storage requirements to account for some OS reporting the remaining storage space in gibibytes.
Jun 9, 2024	• Splitting the sample code into the at-once version, and the streaming version. • Added a code sample for setting the temperature and topK. • Added a note about why keeping a session around is a good idea when the model is expected to be prompted often.
Jun 18, 2024	• Striked out the OS / Chrome language requirement. • Updated support information for workers. • Added a known issue in the troubleshooting section. • Added a regression sidebar in the setup section.
Jun 25, 2024	• Updated ETA for regression fix making it to the dev channel: Jun 26, 2024
Jun 26, 2024	• Updated latest status for the regression (fixed in latest Canary and Dev channels).
Jun 27, 2024	• Updated minimal version to be greater than the 128 release with the regression fix. • Removed the regression sidebar table since it's now resolved in the latest 128 version for both Dev and Canary.

Jun 28, 2024	• Added a requirement for the network connection type. • Updated the steps to take into a recent change requiring the use of `await ai.createTextSession();` to register the desire to use the Prompt API as a condition to trigger the model download.
Jun 28, 2024	• Added screenshot for OptimizationGuide.ModelExecution.OnDeviceModelInstallCriteria.AtRegistration.DiskSpace histogram and deeplinked it.
Jun 29, 2024	• Added a troubleshooting sub-section for reporting crash issues.
Jul 1, 2024	• Added alternative steps in the troubleshooting section based on a report by a participant.
Jul 9, 2024	• Added a note that the disk storage requirement is tied to where the user profile lives.
Jul 11, 2024	• Added “Other updates” section with a link to an index document pointing to all docs and surveys we’ve sent thus far. • Added a note about what to do upon encountering the “too many crashes for this version” error.
Jul 19, 2024	• Added a note about support for session persistence from Chrome 128.0.6602.0. • Added links for reporting implementation issues, and spec issues.
Jul 25, 2024	• Updated status for session persistence. • Noted a fix for the “BAD MESSAGE” crasher.
Aug 8, 2024	• Announcement of two new Task APIs available for local experimentation: summarization and language detection. • Updated info about support for session persistence and cloning. • Updated info about how to change session parameters (temperature and topK).
Aug 9, 2024	• Minor tweak to debug logs section to point folks to the full list of histograms when sharing a report. • Added a note about how to check for group policy state.
Aug 20, 2024	• Added note about model deletion if less than 10 GB of storage remains some time later after an initial successful model download.
Aug 21, 2024	• Heads up about breaking changes in Chrome Canary for the Prompt API.
Aug 27, 2024	• Updated the whole doc to reflect the latest shape and functionality in Chrome canary.
Aug 28, 2024	• Updated two links that pointed to a staging version of the present doc.

Aug 29, 2024	• Added an important caveat about maxToken.
Sep 30, 2024	• Added an approach to emulate stop sequences. • Added information on how a qualifying histogram should look like.
Oct 12, 2024	• Removed control sequence sections since system prompts, N-shots, etc are now taken care of behind the scenes via the API (e.g. see this section of the explainer).
Oct 15, 2024	• Updated the document with the new namespace (`assistant` ⇒ `languageModel`) and reflected the changes in the IDL as well.
Oct 31, 2024	• Added FAQ entry about non-English language support.
Nov 6, 2024	• Updated the `ReadableStream` processing function to work with the non-standard and the standard behavior.