Built-in AI Early Preview Program

The Language Detection API

Authors: Thomas Steiner, Kenji Baheux
Contact: See this section

Last-updated: Sep 25, 2024
See changelog.

News

- Sep 25, 2024: The Language Detection API is **now available for live experimentation** through an origin trial. The linked article will be the canonical reference from there on.

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.

In this update, the Chrome team is thrilled to give you a sneak peek at the language detection API. It's one of the task APIs we are working on, and we are excited to show you how to take it for a spin locally. We'll also share implementation details of this first iteration.

Language Detection API

Purpose

The language detection API is provided for local experimentation. Its purpose is to let you detect the language(s) a given piece of text was written in. Our implementation currently supports 100+ languages and variants.

Early Preview Goals

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

1. The **quality of the language detection,** 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.

    - Note: the current implementation is based on an old version of the API design.

Availability

The language detection API is available, behind an experimental flag, from Chrome 129+ on desktop platforms and Android.

- You'll need Version 129.0.6639.0 or above
- We recommend using Chrome Canary or Chrome dev channel.

Requirements

There are no particular hardware requirements for the language detection API, and the API is supported on all platforms (e.g. Android, ChromeOS, Windows, Mac, Linux) with the exception of iOS where Apple's WebKit engine is mandatory.

Setup

Prerequisites

1. Make sure that you are using Chrome on one of these platforms: Android, ChromeOS, Windows, Mac or Linux.
2. Download Chrome Canary, and confirm that your version is equal to or newer than 129.0.6639.0.

Enable the Language Detection API

Follow these steps to enable the language detection API flag for local experimentation:

1. Open a new tab in Chrome, go to `chrome://flags/#language-detection-api`
2. Select `Enabled`.
3. Relaunch Chrome.

Confirm availability of the Language Detection API

1. Open DevTools and send `await translation.canDetect();` in the console. If this returns `"readily"`, then you are all set.
    - Otherwise, double check that you have followed the prerequisites and the enabling steps.
    - If this still fails, please reach out with details about your environment and any error messages you may have.

API overview

Sample code

Note: the current implementation doesn't correspond to the API as documented in the explainer. The following sample code represents the current implementation, and will eventually break as we adjust the code to be in sync with the explainer.

Explainer, explained.

An explainer 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.

Checking if the language detector is available

JavaScript

```
const canDetect = await translation.canDetect();
let detector;
if (canDetect !== 'no') {
  if (canDetect === 'readily') {
    // The language detector can immediately be used.
    detector = await translation.createDetector();
  } else {
    // The language detector can be used after the model download.
    detector = await translation.createDetector();
    detector.addEventListener('downloadprogress', (e) => {
      console.log(e.loaded, e.total);
    });
    await detector.ready;
  }
} else {
    // The language detector can't be used at all.
}
```

Detecting the language of a string

JavaScript

```
const someUserText = 'Hallo und herzlich willkommen im Early Preview Program!';
const results = await detector.detect(someUserText);
for (const result of results) {
  // Show the full list of potential languages with their likelihood
  // In practice, one would pick the top language(s) crossing a high enough
threshold.
  console.log(result.detectedLanguage, result.confidence);
}
```

Caveats

Here are known temporary limitations:

- **Dependency with Chrome's translation feature:** The API will only function after the Chrome translate feature has processed the page (typically after the page finishes loading). The text you want to analyze with the API doesn't have to be on the page during this initialization step.
- **Performance with long text:** the API may block the main thread for 10s of ms when used with very long strings of text.

Appendix

Full API surface

The full API surface is described below. See Web IDL for details on the language.

```
Unset
[Exposed=(Window,Worker)]
interface Translation {
  Promise<TranslationAvailability> canDetect();
  Promise<LanguageDetector> createDetector();
};

[Exposed=(Window,Worker)]
interface LanguageDetector : EventTarget {
  readonly attribute Promise<undefined> ready;
  attribute EventHandler ondownloadprogress;

  Promise<sequence<LanguageDetectionResult>> detect(DOMString input);
};

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

dictionary LanguageDetectionResult {
  DOMString? detectedLanguage;
  double confidence;
};
```

Supported languages

af	ca	eo	gd	hu	ka	lt	my	ru	sr	uk
am	ceb	es	gl	hy	kk	lv	ne	ru-Latn	st	ur
ar	co	et	gu	id	km	mg	nl	sd	su	uz

ar-Latn	cs	eu	ha	ig	kn	mi	no	si	sv	vi
az	cy	fa	haw	is	ko	mk	ny	sk	sw	xh
be	da	fi	hi	it	ku	ml	pa	sl	ta	yi
bg	de	fil	hi-Latn	iw	ky	mn	pl	sm	te	yo
bg-Latn	el	fr	hmn	ja	la	mr	ps	sn	tg	zh
bn	el-Latn	fy	hr	ja-Latn	lb	ms	pt	so	th	zh-Latn
bs	en	ga	ht	jv	lo	mt	ro	sq	tr	zu

See BCP 47 codes for details.

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`

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:

- 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.

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!

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
Aug 8, 2024	• First version.
Sep 25, 2024	• Announcement of the origin trial.