What's new in Chrome 120 for Extensions

Chrome 120 is now available in beta and includes many exciting updates for Chrome Extension developers.

Sebastian Benz

It’s been a busy year for the Chrome Extensions team. With yesterday’s [Chrome 120 Beta release], the extensions platform is making another big step forward. For an overview on what’s happened this year, check out our quarterly updates from [July] and [October]. Read on for what's new in Chrome 120 for Extensions.

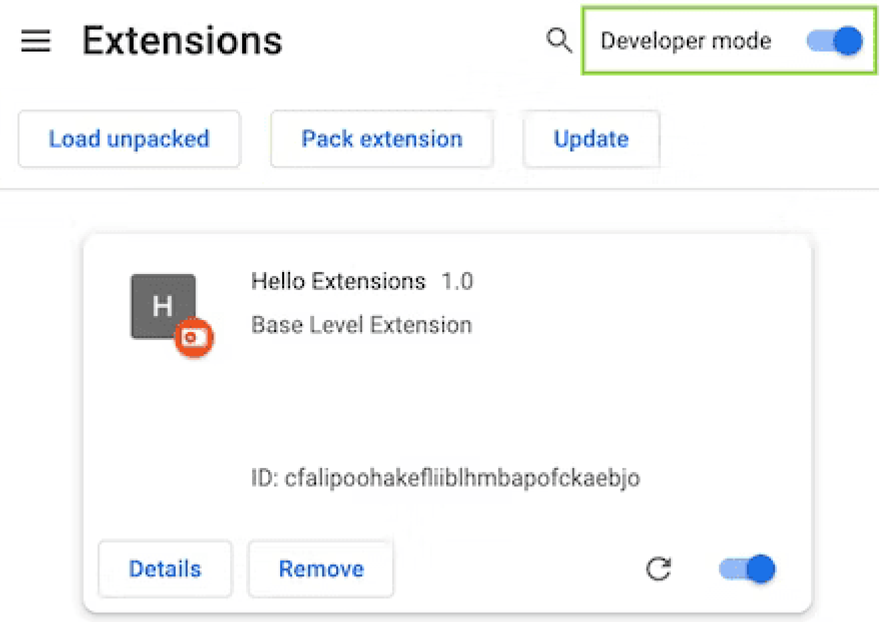
Closing the platform gap

With the release of Chrome 120, we will close the remaining platform gaps listed on our [Manifest V3 known issues page]. The new userScript API as well as support for file handling on ChromeOS have been the two remaining items on the list that we can now cross off, Together with the changes described in the previous quarterly update we are really happy about the current state of the Chrome Extension platform and what we’ve accomplished over the past year.

New userScripts API

User script support has landed! User scripts are (usually relatively small) snippets of code that extensions can inject into web pages in order to modify the page's appearance or behavior. They can be created directly by the user or discovered in a number of different user script repositories around the web. Starting with Chrome 120 Manifest V3 extensions can now manage the collection of user scripts and determine when and how to inject them on web pages.

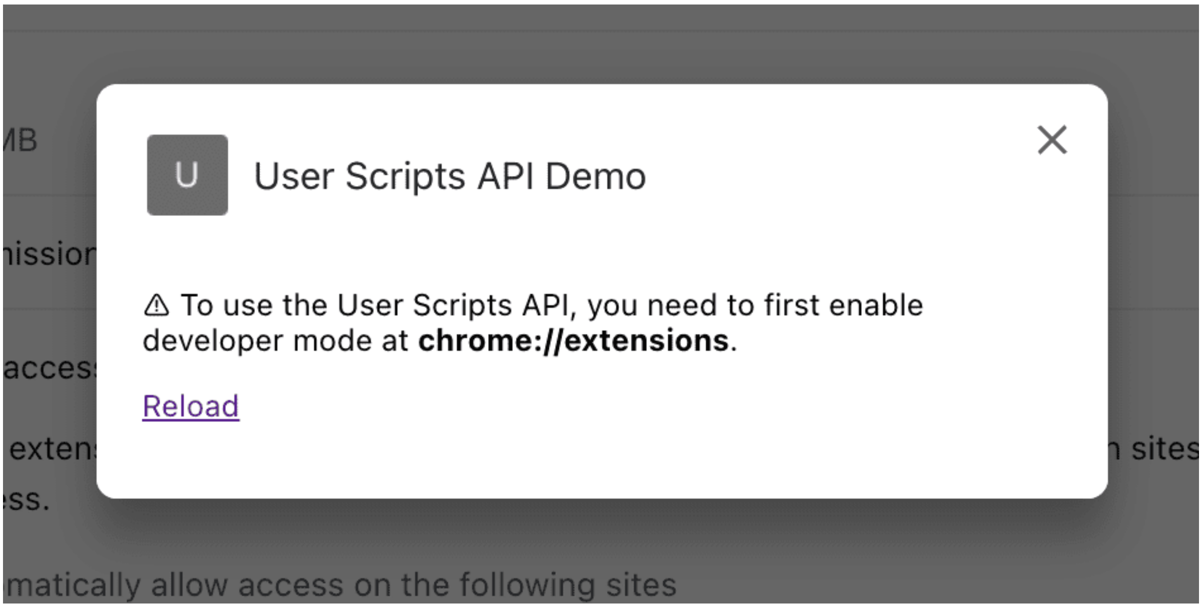
There is one significant difference between user script support in Manifest V2 and Manifest V3. As user scripts are powerful and require high trust in the author of the user script, the Chrome team decided that users must opt into Developer mode before they can run a user script.



Extensions page

(chrome://extensions)

Our new userScript sample demonstrates a simple approach for detecting whether Developer mode is enabled and providing a simple onboarding flow.



⚠ To use the User Scripts API, you need to first enable developer mode at **chrome://extensions**.

Onboard users when Developer mode is disabled.

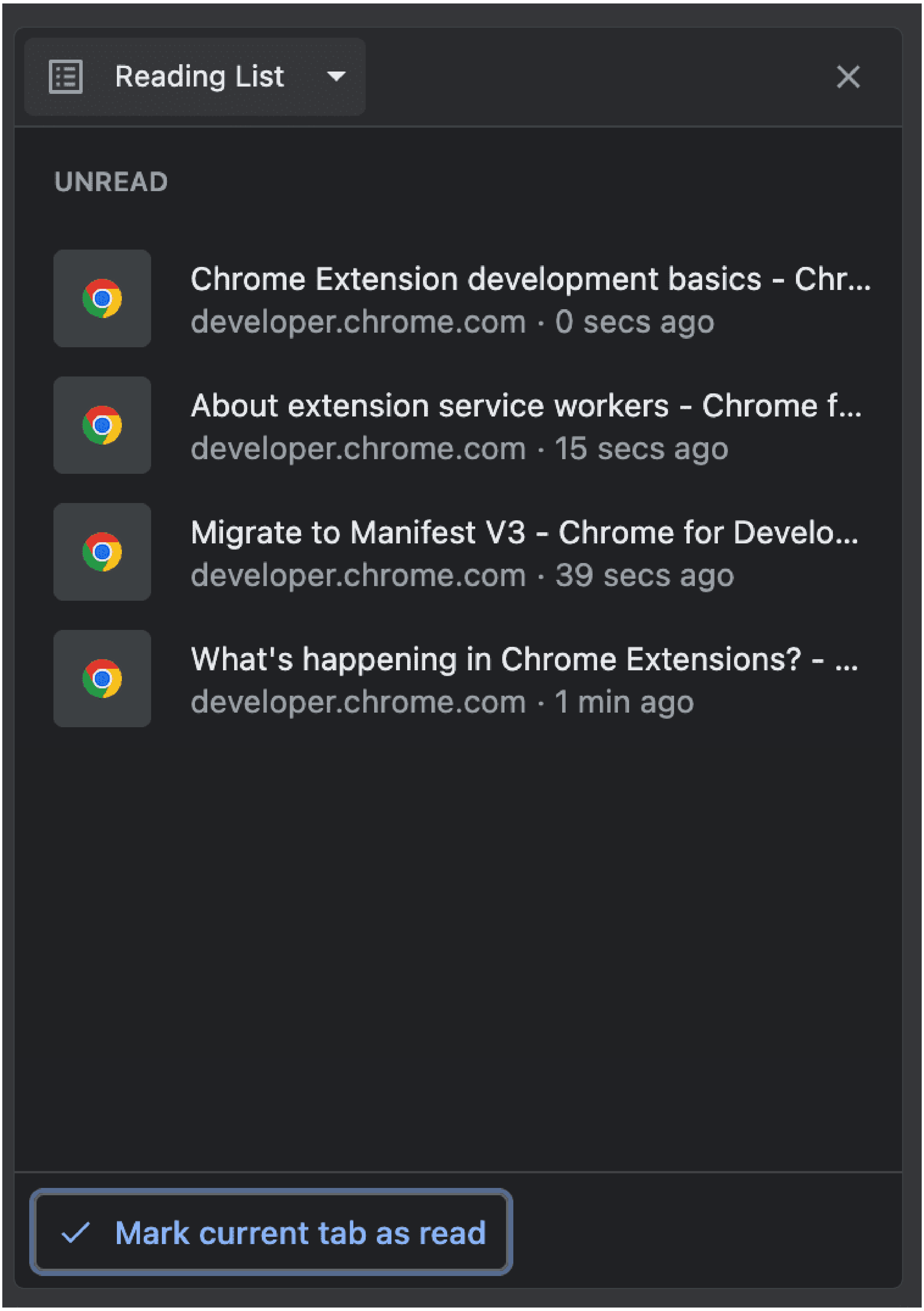
To get started check out the documentation or take a look at the official sample.

Higher static DNR ruleset limits

We significantly increased the limit on enabled static rulesets from 10 to 50. Additionally, we increased the total number of allowed static rulesets from 50 to 100. This is in response to feedback we received in the Web Extensions Community Group.

New ReadingList API

Chrome introduced the reading list in 2021. Last year, the Chrome team made access to the reading list even easier via the side panel. With Chrome 120 we are adding the ability for Chrome Extensions to create, read, update, and delete reading list entries. To learn more, checkout the API docs and our new sample.



UNREAD

Chrome Extension development basics - Chr...
developer.chrome.com · 0 secs ago

About extension service workers - Chrome f...
developer.chrome.com · 15 secs ago

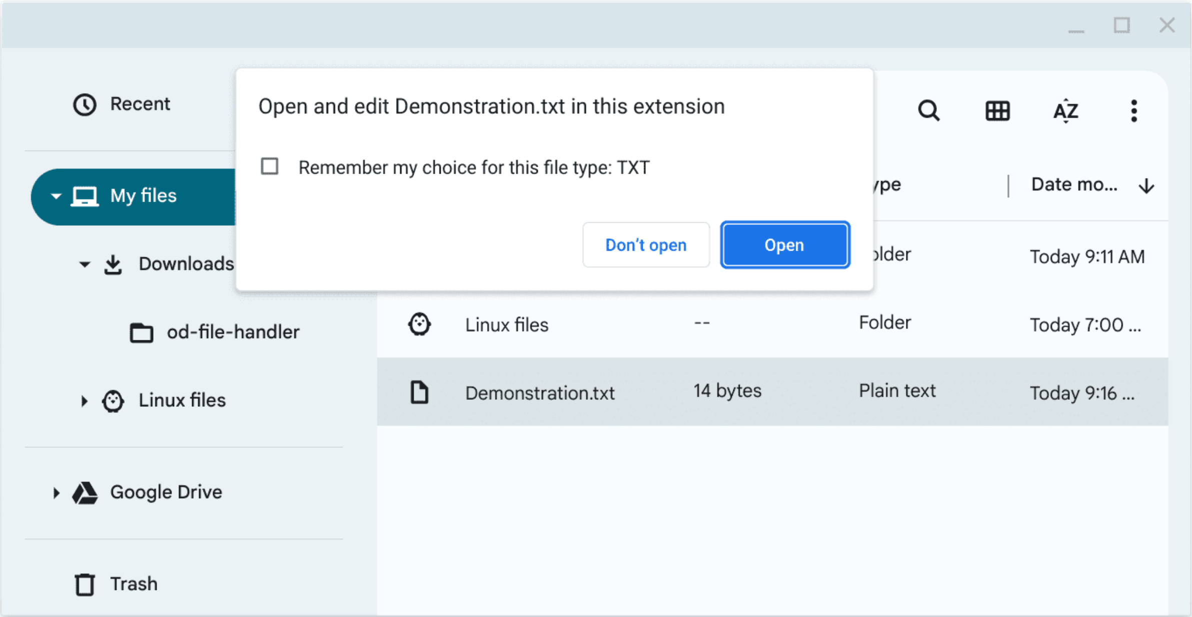
Migrate to Manifest V3 - Chrome for Develo...
developer.chrome.com · 39 secs ago

What's happening in Chrome Extensions? - ...
developer.chrome.com · 1 min ago

The reading list side panel in Chrome.

File handling on ChromeOS

File handling lets extensions open files with specified MIME types and file extensions in a similar manner to web platform file handling. Check out File handling on ChromeOS for more on how to use it.



Open and edit Demonstration.txt in this extension

□ Remember my choice for this file type: TXT

		ype	Date mo...
		older	Today 9:11 AM
Linux files	--	Folder	Today 7:00 ...
Demonstration.txt	14 bytes	Plain text	Today 9:16 ...

Open files in an extension on ChromeOS.

Trigger an alarm in 30 seconds

This is a small update, but addresses an important gap in the service worker lifecycle. Due to the event driven nature of service workers, the recommended way to fire an event in the future is to use `chrome.alarms`. The Alarms API ensures that the event gets fired even if the service worker shuts down in the meantime.

There's a catch though. Before Chrome 120, the shortest time span to trigger an alarm was one minute. However, service workers shut down after 30 seconds of inactivity. So there was no straightforward way to schedule an alarm to fire in 45 seconds, because when using `setTimeout()` to set an event in 45 seconds, the service worker could potentially be shut down before the event fired.

Starting with Chrome 120, you can now either fire an event in:

- less than 30 seconds using `setTimeout()`.
- anything longer than or equal to 30 seconds using `chrome.alarms`:

```
await chrome.alarms.create('demo-default-alarm', {
   periodInMinutes: 0.45
 });
```

Summary

We are really excited about the progress the extension platform has made over the past year. Chrome 120 is another big step with increasing DNR limits and user script support.