Filter

Differences between API implementations

The browser extensions API is an [emerging standard](). As a result, while it is supported by most major browsers – including Firefox, Chrome, Edge, and Opera – there are differences between the various implementations. This means some changes may be necessary to implement your extension for multiple browsers.

Among the various browsers supporting the extensions API, Firefox is the most compliant with the emerging standard and, therefore, is your best place to start when developing browser extensions.

The differences between browsers' API implementations fall into four areas: namespace, asynchronous event handling, API coverage, and manifest keys.

Namespace

You reference all extensions API functions using a namespace. For example, `browser.alarms.create({delayInMinutes});` creates an alarm in Firefox that goes off after the time specified in `delayInMinutes`.

There are two API namespaces in use:

- `chrome` used in Chrome, Edge, and Opera.
- `browser` used in Firefox and Safari.

Asynchronous event handling

JavaScript provides several ways to handle asynchronous events. The proposed extensions API standard is to use promises. The promises approach offers significant advantages when dealing with chained asynchronous event calls.

Firefox and Safari implement promises for the extensions API. All other browsers use callbacks. In Manifest V3, Chrome, Edge, and Opera provided for promises on most appropriate methods. (cf. Chrome bug 328932)

If you are unfamiliar with how JavaScript can handle asynchronous events or promises, look at Getting to know asynchronous JavaScript: Callbacks, Promises and Async/Await or the MDN Using promises page.

API coverage

The differences in the extensions API function implementations among the browsers fall into three broad categories:

- Lack of support for an entire function.
- Variations in the support for features within a function. For example, at the time of writing, Firefox doesn't support the `notification` function method `onButtonClicked`, while Firefox is the only browser that supports `onShown`.
- Proprietary functions supporting browser-specific features. For example, at the time of writing, containers is a Firefox-specific feature supported by the `contextualIdentities` function.

Manifest keys

The differences in the supported `manifest.json` keys among the browsers fall broadly into two categories:

- Extension information attributes. For example, at the time of writing, Firefox and Opera include the `developer` key (in addition to the

Browser extensions

▶ Getting started

▼ Concepts

JavaScript APIs

Content scripts

Background scripts

Match patterns

Work with files

Internationalization

Content Security Policy

Native messaging

Differences between API implementations

Chrome incompatibilities

▶ User interface

▶ How to

▶ JavaScript APIs

▶ Manifest keys

▶ Extension Workshop

Contact us

▶ Channels

`author` key) to record details about the extension developer.

- Extension features. For example, at the time of writing, only Firefox supports the `protocol_handlers` key (which registers web-based protocol handlers, applications that know how to handle particular types of links).

More information

You can find more detailed information about the differences in the supported browser extensions API features in:

- Chrome incompatibilities
- Browser support for JavaScript APIs
- Browser compatibility for manifest.json

Found a content problem with this page?

- Edit the page on GitHub.
- Report the content issue.
- View the source on GitHub.

Want to get more involved? Learn how to contribute.

This page was last modified on Oct 2, 2023 by MDN contributors.