Match patterns

Match patterns are a way to specify groups of URLs: a match pattern matches a specific set of URLs. They are used in WebExtensions APIs in a few places, most notably to specify which documents to load content scripts into, and to specify which URLs to add `webRequest` listeners to.

APIs that use match patterns usually accept a list of match patterns, and will perform the appropriate action if the URL matches any of the patterns. See, for example, the `content_scripts` key in manifest.json.

Match pattern structure

> **Note:** Some browsers don't support certain schemes. Check the Browser compatibility table for details.

All match patterns are specified as strings. Apart from the special `<all_urls>` pattern, match patterns consist of three parts: *scheme*, *host*, and *path*. The scheme and host are separated by `://`.

```
<scheme>://<host><path>
```

scheme

The *scheme* component may take one of two forms:

Form	Matches
`*`	Only "http" and "https" and in some browsers also "ws" and "wss".
One of `http`, `https`, `ws`, `wss`, `ftp`, `data`, `file`, or `(chrome-)extension`.	Only the given scheme.

host

The *host* component may take one of three forms:

Form	Matches
`*`	Any host.
`*.` followed by part of the hostname.	The given host and any of its subdomains.
A complete hostname, without wildcards.	Only the given host.

host must not include a port number.

host is optional only if the *scheme* is "file".

Note that the wildcard may only appear at the start.

path

The *path* component must begin with a `/`.

In this article
Match pattern structure
Examples
Browser compatibility

After that, it may subsequently contain any combination of the `*` wildcard and any of the characters that are allowed in URL paths or query strings. Unlike *host*, the *path* component may contain the `*` wildcard in the middle or at the end, and the `*` wildcard may appear more than once.

The value for the *path* matches against the string which is the URL path plus the URL query string. This includes the `?` between the two, if the query string is present in the URL. For example, if you want to match URLs on any domain where the URL path ends with `foo.bar`, then you need to use an array of Match Patterns like `['*://*/*foo.bar', '*://*/*foo.bar?*']`. The `?*` is needed, rather than just `bar*`, in order to anchor the ending `*` as applying to the URL query string and not some portion of the URL path.

Neither the URL fragment identifier, nor the `#` which precedes it, are considered as part of the *path*.

Note: The path pattern string should not include a port number. Adding a port, as in: `http://localhost:1234/*` causes the match pattern to be ignored. However, `http://localhost:1234` will match with `http://localhost/*`.

<all_urls>

The special value `<all_urls>` matches all URLs under any of the supported schemes: that is "http", "https", "ws", "wss", "ftp", "data", and "file".

Examples

Pattern	Example matches	Example non-matches

`<all_urls>`
 	
Match all URLs.
	`http://example.org/`
	`https://a.org/some/path/`
	`ws://sockets.somewhere.org/`
	`wss://ws.example.com/stuff/`
	`ftp://files.somewhere.org/`
		`resource://a/b/c/`
		(unsupported scheme)
		`ftps://files.somewhere.org/`
		(unsupported scheme)

`*://*/*`
 	
Match all HTTP, HTTPS and WebSocket URLs.
	`http://example.org/`
	`https://a.org/some/path/`
	`ws://sockets.somewhere.org/`
	`wss://ws.example.com/stuff/`
		`ftp://ftp.example.org/`
		(unmatched scheme)
		`file:///a/`
		(unmatched scheme)

	`http://mozilla.org/`	

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
Browser extensions

Pattern	Example matches	Example non-matches

`*://*.mozilla.org/*`
 	
Match all HTTP, HTTPS and WebSocket URLs that are hosted at "mozilla.org" or one of its subdomains.
	`http://mozilla.org/`
	`http://a.mozilla.org/`
	`http://a.b.mozilla.org/`
	`https://b.mozilla.org/path/`
	`ws://ws.mozilla.org/`
	`wss://secure.mozilla.org/something`
		`ftp://mozilla.org/` (unmatched scheme)
		`http://mozilla.com/` (unmatched host)
		`http://firefox.org/` (unmatched host)

`*://mozilla.org/`
 	
Match all HTTP, HTTPS and WebSocket URLs that are hosted at exactly "mozilla.org/".
	`http://mozilla.org/`
	`https://mozilla.org/`
	`ws://mozilla.org/`
	`wss://mozilla.org/`
		`ftp://mozilla.org/` (unmatched scheme)
		`http://a.mozilla.org/` (unmatched host)
		`http://mozilla.org/a` (unmatched path)

`ftp://mozilla.org/` Match only "ftp://mozilla.org/".	`ftp://mozilla.org`	`http://mozilla.org/` (unmatched scheme) `ftp://sub.mozilla.org/` (unmatched host) `ftp://mozilla.org/path` (unmatched path)

`https://*/path`
 	
Match HTTPS URLs on any host, whose path is "path".
	`https://mozilla.org/path`
	`https://a.mozilla.org/path`
	`https://something.com/path`
		`http://mozilla.org/path` (unmatched scheme)
		`https://mozilla.org/path/` (unmatched path)
		`https://mozilla.org/a` (unmatched path)
		`https://mozilla.org/` (unmatched path)
		`https://mozilla.org/path?foo=1` (unmatched path due to URL query string)

		`http://mozilla.org/path/`

Pattern	Example matches	Example non-matches

`https://*/path/`
 	
Match HTTPS URLs on any host, whose path is "path/" and which has no URL query string.
	`https://mozilla.org/path/`
	
	`https://a.mozilla.org/path/`
	
	`https://something.com/path/`
		`https://mozilla.org/path`
		(unmatched path)
		
		`https://mozilla.org/a`
		(unmatched path)
		
		`https://mozilla.org/`
		(unmatched path)
		
		`https://mozilla.org/path/?foo=1`
		(unmatched path due to URL query string)

`https://mozilla.org/*`
 	
Match HTTPS URLs only at "mozilla.org", with any URL path and URL query string.
	`https://mozilla.org/`
	
	`https://mozilla.org/path`
	
	`https://mozilla.org/another`
	
	`https://mozilla.org/path/to/doc`
	
	`https://mozilla.org/path/to/doc?foo=1`
		`http://mozilla.org/path`
		(unmatched scheme)
		
		`https://mozilla.com/path`
		(unmatched host)

`https://mozilla.org/a/b/c/`
 	
Match only this URL, or this URL with any URL fragment.
	`https://mozilla.org/a/b/c/`
	
	`https://mozilla.org/a/b/c/#section1`
		Anything else.

`https://mozilla.org/*/b/*/`
 	
Match HTTPS URLs hosted on "mozilla.org", whose path contains a component "b" somewhere in the middle. Will match URLs with query strings, if the string ends in a `/`.
	`https://mozilla.org/a/b/c/`
	
	`https://mozilla.org/d/b/f/`
	
	`https://mozilla.org/a/b/c/d/`
	
	`https://mozilla.org/a/b/c/d/#section1`
	
	`https://mozilla.org/a/b/c/d/?foo=/`
	
	`https://mozilla.org/a?foo=21314&bar=/b/&extra=c/`
		`https://mozilla.org/b/*/`
		(unmatched path)
		
		`https://mozilla.org/a/b/`
		(unmatched path)
		
		`https://mozilla.org/a/b/c/d/?foo=bar`
		(unmatched path due to URL query string)

`file:///blah/*`
 	
Match any FILE URL whose path begins with "blah".
	`file:///blah/`
	
	`file:///blah/bleh`
		`file:///bleh/`
		(unmatched path)

Pattern	Example matches	Example non-matches

Invalid match patterns

Invalid pattern	Reason
`resource://path/`	Unsupported scheme.
`https://mozilla.org`	No path.
`https://mozilla.*.org/`	"*" in host must be at the start.
`https://*zilla.org/`	"*" in host must be the only character or be followed by ".".
`http*://mozilla.org/`	"*" in scheme must be the only character.
`https://mozilla.org:80/`	Host must not include a port number.
`*://*`	Empty path: this should be "`*://*/*`".
`file://*`	Empty path: this should be "`file:///*`".

Browser compatibility

BCD tables only load in the browser

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 Aug 2, 2023 by MDN contributors.

Developers