

Document: requestStorageAccessFor() method



- [Syntax](#)
 - [Parameters](#)
 - [Return value](#)
 - [Exceptions](#)
 - [Description](#)
 - [Examples](#)
 - [Specifications](#)
 - [Browser compatibility](#)
 - [See also](#)
-

{{APIRef("Storage Access API")}} {{non-standard_header}}

The `requestStorageAccessFor()` method of the {{domxref("Document")}} interface allows top-level sites to request third-party cookie access on behalf of embedded content originating from another site in the same [related website set](#). It returns a {{jsxref("Promise")}} that resolves if the access was granted, and rejects if access was denied.

Syntax

```
requestStorageAccessFor(requestedOrigin)
```

Parameters

- `requestedOrigin`
 - : A string representing the URL of the origin you are requesting third-party cookie access for.

Return value

A `{{jsxref("Promise")}}` that fulfills with `undefined` if the access to third-party cookies was granted and rejects if access was denied.

`requestStorageAccessFor()` requests are automatically denied unless the top-level content is currently processing a user gesture such as a tap or click (`{{Glossary("transient activation")}}`), or unless permission was already granted previously. If permission was not previously granted, they must run inside a user gesture-based event handler. The user gesture behavior depends on the state of the promise:

- If the promise resolves (i.e., permission was granted), then the user gesture has not been consumed, so the script can subsequently call APIs requiring a user gesture.
- If the promise is rejected (i.e., permission was not granted), then the user gesture has been consumed, so the script can't do anything that requires a gesture. This prevents scripts from calling `requestStorageAccessFor()` again if permission is denied.

Exceptions

- `InvalidStateError` `{{domxref("DOMException")}}`
 - : Thrown if the current `{{domxref("Document")}}` is not yet active.
- `NotAllowedError` `{{domxref("DOMException")}}`
 - : Thrown if:
 - The document's window is not a [secure context](#).
 - The document is not the top-level document.
 - The document has a `null` origin.
 - The supplied `requestedOrigin` is [opaque](#).
 - The top-level and embedded sites are not in the same [related website set](#).
 - The embedding `{{htmlelement("iframe")}}` is sandboxed, and the `allow-storage-access-by-user-activation` token is not set.
 - Usage is blocked by a `{{httpheader("Permissions-Policy/storage-access", "storage-access")}}` [Permissions Policy](#).
 - Usage is denied by the user agent's permission request to use the API.
- `{{jsxref("TypeError")}}`
 - : Thrown if `requestedOrigin` is not a valid URL.

Description

The `requestStorageAccessFor()` method addresses challenges in adopting the Storage Access API on top-level sites that use cross-site images or scripts requiring cookies. It is relevant to user agents that by default block access to [third-party](#), [unpartitioned](#) cookies to improve privacy (e.g., to prevent tracking), and is a proposed extension of the [Storage Access API](#).

`requestStorageAccessFor()` can enable third-party cookie access for cross-site resources directly embedded into a top-level site that are unable to request storage access themselves, for example `{{htmlElement("img")}}` elements. Cross-site content embedded in `<iframe>`s that has its own logic and resources and needs third-party cookie access should request storage access via `{{domxref("Document.requestStorageAccess()")}}`.

To check whether permission to access third-party cookies has already been granted via `requestStorageAccessFor()`, you can call `{{domxref("Permissions.query()")}}`, specifying the feature name `"top-level-storage-access"`. This is different from the feature name used for the regular `{{domxref("Document.requestStorageAccess()")}}` method, which is `"storage-access"`.

The `Permissions.query()` call must specify the embedded origin; for example:

```
navigator.permissions.query({
  name: "top-level-storage-access",
  requestedOrigin: "https://www.example.com",
});
```

“ [!NOTE] Usage of this feature may be blocked by a `{{httpheader("Permissions-Policy/storage-access", "storage-access")}}` [Permissions Policy](#) set on your server (the same one that controls the rest of the Storage Access API). In addition, the document must pass additional browser-specific checks such as allowlists, blocklists, on-device classification, user settings, or anti-[clickjacking](#) heuristics.

Examples

```
function rSAFor() {
  if ("requestStorageAccessFor" in document) {
    document.requestStorageAccessFor("https://example.com").then(
```

```

    (res) => {
      // Use storage access
      doThingsWithCookies();
    },
    (err) => {
      // Handle errors
    },
  );
}
}

```

After a successful `requestStorageAccessFor()` call, cross-site requests will include cookies if they include [CORS](#) / [crossorigin](#), so sites may want to wait before triggering a request. Such requests must use the `credentials: "include"` option and resources must include the `crossorigin="use-credentials"` attribute.

For example:

```

function checkCookie() {
  fetch("https://example.com/getcookies.json", {
    method: "GET",
    credentials: "include",
  })
  .then((response) => response.json())
  .then((json) => {
    // Do something
  });
}

```

“ [!NOTE] See [Using the Storage Access API](#) for a more complete example.

Specifications

{{Specifications}}

Browser compatibility

{{Compat}}

See also

- {{domxref("Document.hasStorageAccess()")}},
{{domxref("Document.hasUnpartitionedCookieAccess()")}},
{{domxref("Document.requestStorageAccess()")}}
 - [Using the Storage Access API](#)
 - [Introducing Storage Access API](#) (WebKit blog)
-

<div><div></div><div></div></div>	<div><div></div><div></div></div>	#2	
<div><div></div><div></div></div>	<div><div></div><div></div></div>		2026-10-31 03:27:53 UTC
<div><div></div><div></div></div>	<div><div></div><div></div></div>		2026-10-31 03:27:53 UTC