Window: fetch() method

- Syntax
  - Parameters
  - Return value
  - Exceptions
- Examples
- Specifications
- Browser compatibility
- See also

The `fetch()` method of the Window interface starts the process of fetching a resource from the network, returning a promise that is fulfilled once the response is available.

The promise resolves to the Response object representing the response to your request.

A `fetch()` promise only rejects when the request fails, for example, because of a badly-formed request URL or a network error. A `fetch()` promise *does not* reject if the server responds with HTTP status codes that indicate errors (`404`, `504`, etc.). Instead, a `then()` handler must check the Response.ok and/or Response.status properties.

The `fetch()` method is controlled by the `connect-src` directive of Content Security Policy rather than the directive of the resources it's retrieving.

> **NOTE:** The `fetch()` method's parameters are identical to those of the Request() constructor.

Syntax

```
fetch(resource)
fetch(resource, options)
```

Parameters

- `resource`
  - : This defines the resource that you wish to fetch. This can either be:
    - A string or any other object with a stringifier — including a URL object — that provides the URL of the resource you want to fetch. The URL may be relative to the base URL, which is the document's baseURI in a window context, or WorkerGlobalScope.location in a worker context.
    - A Request object.
- `options`
  - : A RequestInit object containing any custom settings that you want to apply to the request.

Return value

A Promise that resolves to a Response object.

Exceptions

- `AbortError` DOMException
  - : The request was aborted due to a call to the AbortController abort() method.
- `NotAllowedError` DOMException
  - : Thrown if:
    - Use of the Topics API is specifically disallowed by a browsing-topics Permissions Policy, and `browsingTopics` is set to `true`.
    - Use of Private State Token API operations is specifically disallowed by a private-state-token-issuance or private-state-token-redemption Permissions Policy, and the `privateToken` option is specified, including a disallowed `privateToken.operation` type.
- TypeError
  - : Can occur for the following reasons:
    - The requested URL is invalid.
    - The requested URL includes credentials (username and password).

- The RequestInit object passed as the value of `options` included properties with invalid values.
- The request is blocked by a permissions policy.
- There is a network error (for example, because the device does not have connectivity).
- The `privateToken` init option is specified, including a `privateToken.operation` type of `send-redemption-record`, and the `privateToken.issues` array was empty or not set, or one or more of the specified `issuers` are not trustworthy, HTTPS URLs.

Examples

In our Fetch Request example (see Fetch Request live) we create a new Request object using the relevant constructor, then fetch it using a `fetch()` call. Since we are fetching an image, we run Response.blob() on the response to give it the proper MIME type so it will be handled properly, then create an Object URL of it and display it in an img element.

```
const myImage = document.querySelector("img");

const myRequest = new Request("flowers.jpg");

window
  .fetch(myRequest)
  .then((response) => {
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    return response.blob();
  })
  .then((response) => {
    myImage.src = URL.createObjectURL(response);
  });
```

In our Fetch Request with init example (see Fetch Request init live) we do the same thing except that we pass in an *options* object when we invoke `fetch()`. In this case, we can set a Cache-Control value to indicate what kind of cached responses we're okay with:

```
const myImage = document.querySelector("img");
const reqHeaders = new Headers();

// A cached response is okay unless it's more than a week old
reqHeaders.set("Cache-Control", "max-age=604800");

const options = {
  headers: reqHeaders,
};

// Pass init as an "options" object with our headers.
const req = new Request("flowers.jpg", options);

fetch(req).then((response) => {
  // …
});
```

You could also pass the `init` object in with the `Request` constructor to get the same effect:

```
const req = new Request("flowers.jpg", options);
```

You can also use an object literal as `headers` in `init`:

```
const options = {
  headers: {
    "Cache-Control": "max-age=60480",
  },
};

const req = new Request("flowers.jpg", options);
```

The Using fetch article provides more examples of using `fetch()`.

Specifications

Browser compatibility

See also

- WorkerGlobalScope.fetch()
- Fetch API
- ServiceWorker API
- HTTP access control (CORS)
- HTTP

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:35:07 UTC