Response

- Constructor
- Instance properties
- Static methods
- Instance methods
- Examples
  - Fetching an image
  - A PHP Call
- Specifications
- Browser compatibility
- See also

The **`Response`** interface of the Fetch API represents the response to a request.

You can create a new `Response` object using the Response() constructor, but you are more likely to encounter a `Response` object being returned as the result of another API operation—for example, a service worker FetchEvent.respondWith, or a simple fetch().

Constructor

- Response()
  - : Creates a new `Response` object.

Instance properties

- Response.body
  - : A ReadableStream of the body contents.
- Response.bodyUsed

    - : Stores a boolean value that declares whether the body has been used in a response yet.
- Response.headers
    - : The Headers object associated with the response.
- Response.ok
    - : A boolean indicating whether the response was successful (status in the range `200` – `299`) or not.
- Response.redirected
    - : Indicates whether or not the response is the result of a redirect (that is, its URL list has more than one entry).
- Response.status
    - : The status code of the response. (This will be `200` for a success).
- Response.statusText
    - : The status message corresponding to the status code. (e.g., `OK` for `200`).
- Response.type
    - : The type of the response (e.g., `basic`, `cors`).
- Response.url
    - : The URL of the response.

Static methods

- Response.error()
    - : Returns a new `Response` object associated with a network error.
- Response.redirect()
    - : Returns a new response with a different URL.
- Response.json()
    - : Returns a new `Response` object for returning the provided JSON encoded data.

Instance methods

- Response.arrayBuffer()
    - : Returns a promise that resolves with an ArrayBuffer representation of the response body.
- Response.blob()
    - : Returns a promise that resolves with a Blob representation of the response body.

- Response.bytes()
  - : Returns a promise that resolves with a Uint8Array representation of the response body.
- Response.clone()
  - : Creates a clone of a `Response` object.
- Response.formData()
  - : Returns a promise that resolves with a FormData representation of the response body.
- Response.json()
  - : Returns a promise that resolves with the result of parsing the response body text as JSON.
- Response.text()
  - : Returns a promise that resolves with a text representation of the response body.
- Response.textStream()
  - : Returns a ReadableStream that can be used to read the contents of the response body in chunks of UTF-8.

Examples

Fetching an image

In our basic fetch example (run example live) we use a simple `fetch()` call to grab an image and display it in an img element. The `fetch()` call returns a promise, which resolves to the `Response` object associated with the resource fetch operation.

You'll notice that since we are requesting an image, we need to run Response.blob to give the response its correct MIME type.

```
const image = document.querySelector(".my-image");
fetch("flowers.jpg")
  .then((response) => {
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }
    return response.blob();
  })
```

```
  .then((blob) => {
    const objectURL = URL.createObjectURL(blob);
    image.src = objectURL;
  })
  .catch((error) => {
    console.error("Error fetching the image:", error);
  });
```

You can also use the Response() constructor to create your own custom `Response` object:

```
const response = new Response();
```

A PHP Call

Here we call a PHP program file that generates a JSON string, displaying the result as a JSON value.

```
// Function to fetch JSON using PHP
const getJSON = async () => {
  // Generate the Response object
  const response = await fetch("getJSON.php");
  if (response.ok) {
    // Get JSON value from the response body
    return response.json();
  }
  throw new Error("*** PHP file not found");
};

// Call the function and output value or error message to console
getJSON()
  .then((result) => console.log(result))
  .catch((error) => console.error(error));
```

Specifications

Browser compatibility

See also

- ServiceWorker API
- HTTP access control (CORS)
- HTTP

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:34:57 UTC