Response: clone() method

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

The `clone()` method of the Response interface creates a clone of a response object, identical in every way, but stored in a different variable.

Like the underlying ReadableStream.tee api, the body of a cloned `Response` will signal backpressure at the rate of the *faster* consumer of the two bodies, and unread data is enqueued internally on the slower consumed `body` without any limit or backpressure. Backpressure refers to the mechanism by which the streaming consumer of data (in this case, the code that reads the body) slows down the producer of data (such as the TCP server) so as not to load large amounts of data in memory that is waiting to be used by the application. If only one cloned branch is consumed, then the entire body will be buffered in memory. Therefore, `clone()` is one way to read a response twice in sequence, but you should not use it to read very large bodies in parallel at different speeds.

`clone()` throws a TypeError if the response body has already been used. In fact, the main reason `clone()` exists is to allow multiple uses of body objects (when they are one-use only.)

Syntax

```
clone()
```

Parameters

None.

Return value

A Response object.

Examples

In our Fetch Response clone example (see Fetch Response clone live) we create a new Request object using the Request() constructor, passing it a JPG path. We then fetch this request using fetch(). When the fetch resolves successfully, we clone it, extract a blob from both responses using two Response.blob calls, create object URLs out of the blobs using URL.createObjectURL(), and display them in two separate img elements.

```
const image1 = document.querySelector(".img1");
const image2 = document.querySelector(".img2");

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

fetch(myRequest).then((response) => {
  const response2 = response.clone();

  response.blob().then((myBlob) => {
    const objectURL = URL.createObjectURL(myBlob);
    image1.src = objectURL;
  });

  response2.blob().then((myBlob) => {
    const objectURL = URL.createObjectURL(myBlob);
    image2.src = objectURL;
  });
});
```

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 17:34:56 UTC