Response: arrayBuffer() method

- Syntax
  - Parameters
  - Return value
  - Exceptions
- Examples
  - Playing music
  - Reading files
- Specifications
- Browser compatibility
- See also

The `arrayBuffer()` method of the Response interface takes a Response stream and reads it to completion. It returns a promise that resolves with an ArrayBuffer.

Syntax

```
arrayBuffer()
```

Parameters

None.

Return value

A promise that resolves with an ArrayBuffer.

Exceptions

- `AbortError` DOMException
  - : The request was aborted.
- TypeError
  - : Thrown for one of the following reasons:
    - The response body is disturbed or locked.
    - There was an error decoding the body content (for example, because the Content-Encoding header is incorrect).
- RangeError
  - : Thrown if there is a problem creating the associated `ArrayBuffer` (for example, if the data size is too large).

Examples

Playing music

In our fetch array buffer live, we have a Play button. When pressed, the `getData()` function is run. Note that before playing full audio file will be downloaded. If you need to play ogg during downloading (stream it) - consider HTMLAudioElement:

```
new Audio("music.ogg").play();
```

In `getData()` we create a new request using the Request() constructor, then use it to fetch an OGG music track. We also use AudioContext.createBufferSource to create an audio buffer source. When the fetch is successful, we read an ArrayBuffer out of the response using `arrayBuffer()`, decode the audio data using AudioContext.decodeAudioData(), set the decoded data as the audio buffer source's buffer (`source.buffer`), then connect the source up to the AudioContext.destination.

Once `getData()` has finished running, we start the audio source playing with `start(0)`, then disable the play button so it can't be clicked again when it is already playing (this would cause an error.)

```
function getData() {
  const audioCtx = new AudioContext();

  return fetch("viper.ogg")
    .then((response) => {
      if (!response.ok) {
        throw new Error(`HTTP error, status = ${response.status}`);
      }
      return response.arrayBuffer();
    })
    .then((buffer) => audioCtx.decodeAudioData(buffer))
    .then((decodedData) => {
      const source = new AudioBufferSourceNode(audioCtx);
      source.buffer = decodedData;
      source.connect(audioCtx.destination);
      return source;
    });
}

// wire up buttons to stop and play audio

play.onclick = () => {
  getData().then((source) => {
    source.start(0);
    play.setAttribute("disabled", "disabled");
  });
};
```

Reading files

The Response() constructor accepts Files and Blobs, so it may be used to read a File into other formats.

```
<input type="file" />
```

```
function readFile(file) {
  return new Response(file).arrayBuffer();
}
```

```
document
  .querySelector("input[type=file]")
  .addEventListener("change", (event) => {
    const file = event.target.files[0];
    const buffer = readFile(file);
  });
```

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