Streams API

The Streams API allows JavaScript to programmatically access streams of data received over the network and process them as desired by the developer.

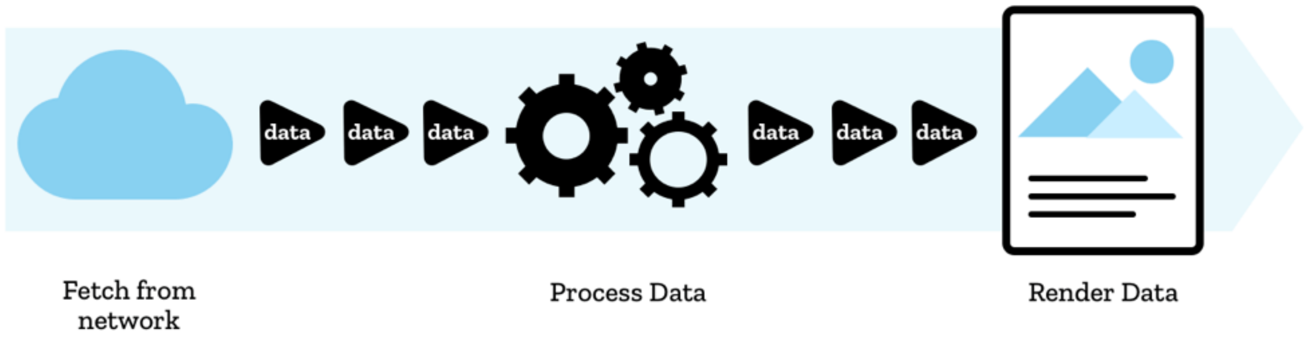
> **Note:** This feature is available in Web Workers

Concepts and usage

Streaming involves breaking a resource that you want to receive over a network down into small chunks, then processing it bit by bit. This is something browsers do anyway when receiving assets to be shown on webpages — videos buffer and more is gradually available to play, and sometimes you'll see images display gradually as more is loaded.

But this has never been available to JavaScript before. Previously, if we wanted to process a resource of some kind (be it a video, or a text file, etc.), we'd have to download the entire file, wait for it to be deserialized into a suitable format, then process the whole lot after it is fully received.

With Streams being available to JavaScript, this all changes — you can now start processing raw data with JavaScript bit by bit as soon as it is available on the client-side, without needing to generate a buffer, string, or blob.



There are more advantages too — you can detect when streams start or end, chain streams together, handle errors and cancel streams as required, and react to the speed the stream is being read at.

The basic usage of Streams hinges around making responses available as streams. For example, the response body returned by a successful fetch request can be exposed as a `ReadableStream`, and you can then read it using a reader created with `ReadableStream.getReader()`, cancel it with `ReadableStream.cancel()`, etc.

More complicated uses involve creating your own stream using the `ReadableStream()` constructor, for example to process data inside a service worker.

You can also write data to streams using `WritableStream`.

> **Note:** You can find a lot more details about the theory and practice of streams in our articles — Streams API concepts, Using readable streams, and Using writable streams.

Stream interfaces

Readable streams

`ReadableStream`

Represents a readable stream of data. It can be used to handle response streams of the Fetch API, or developer-defined streams (e.g. a custom `ReadableStream()` constructor).

`ReadableStreamDefaultReader`

Represents a default reader that can be used to read stream data supplied from a network (e.g. a fetch request).

`ReadableStreamDefaultController`

Represents a controller allowing control of a `ReadableStream`'s state and internal queue. Default controllers are for streams that are not byte streams.

Writable streams

`WritableStream`

Provides a standard abstraction for writing streaming data to a destination, known as a sink. This object comes with built-in backpressure and queuing sink. This object comes with built-in backpressure and queuing.

`WritableStreamDefaultWriter`

Represents a default writable stream writer that can be used to write chunks of data to a writable stream.

`WritableStreamDefaultController`

Represents a controller allowing control of a `WritableStream`'s state. When constructing a `WritableStream`, the underlying sink is given a corresponding `WritableStreamDefaultController` instance to manipulate.

Transform Streams

`TransformStream`

Represents a set of transformable data.

`TransformStreamDefaultController`

Provides methods to manipulate the `ReadableStream` and `WritableStream` associated with a transform stream.

Related stream APIs and operations

`ByteLengthQueuingStrategy`

Provides a built-in byte length queuing strategy that can be used when constructing streams.

`CountQueuingStrategy`

Provides a built-in chunk counting queuing strategy that can be used when constructing streams.

Extensions to other APIs

`Request`

When a new `Request` object is constructed, you can pass it a `ReadableStream` in the `body` property of its `RequestInit` dictionary. This `Request` could then be passed to a `fetch()` to commence fetching the stream.

`Response.body`

The response body returned by a successful fetch request is exposed by default as a

The response body returned by a successful fetch request is exposed by default as a `ReadableStream`, and can have a reader attached to it, etc.

ByteStream-related interfaces

Warning: these are not implemented anywhere as yet, and questions have been raised as to whether the spec details are in a finished enough state for them to be implemented. This may change over time.

`ReadableStreamBYOBReader`

Represents a BYOB ("bring your own buffer") reader that can be used to read stream data supplied by the developer (e.g. a custom `ReadableStream()` constructor).

`ReadableByteStreamController`

Represents a controller allowing control of a `ReadableStream`'s state and internal queue. Byte stream controllers are for byte streams.

`ReadableStreamBYOBRequest`

Represents a pull into request in a `ReadableByteStreamController`.

Examples

We have created a directory of examples to go along with the Streams API documentation — see mdn/dom-examples/streams. The examples are as follows:

- Simple stream pump: This example shows how to consume a ReadableStream and pass its data to another.
- Grayscale a PNG: This example shows how a ReadableStream of a PNG can be turned into grayscale.
- Simple random stream: This example shows how to use a custom stream to generate random strings, enqueue them as chunks, and then read them back out again.
- Simple tee example: This example extends the Simple random stream example, showing how a stream can be teed and both resulting streams can be read independently.
- Simple writer: This example shows how to write to a writable stream, then decode the stream and write the contents to the UI.
- Unpack chunks of a PNG: This example shows how `pipeThrough()` can be used to

transform a ReadableStream into a stream of other data types by transforming a data of a PNG file into a stream of PNG chunks.

Examples from other developers:

- Progress Indicators with Streams, Service Workers, & Fetch.

Specifications

Specification
Streams Living Standard

Browser compatibility

ReadableStream

Report problems with this compatibility data on GitHub

`ReadableStream`	
Chrome	43
Edge	14
Firefox	65
Internet Explorer	**No**
Opera	30
Safari	10.1
WebView Android	43
Chrome Android	43
Firefox for Android	65
Opera Android	30
Safari on iOS	10.3
Samsung Internet	4.0
Deno	1.0

`ReadableStream()` constructor	
Chrome	43
Edge	79
Firefox	65
Internet Explorer	**No**
Opera	30
Safari	10.1
WebView Android	43
Chrome Android	43
Firefox for Android	65
Opera Android	30
Safari on iOS	10.3
Samsung Internet	4.0
Deno	1.0

`cancel`	
Chrome	43
Edge	14
Firefox	65
Internet Explorer	**No**
Opera	30
Safari	10.1
WebView Android	43
Chrome Android	43
Firefox for Android	65
Opera Android	30
Safari on iOS	10.3

Samsung Internet	4.0
Deno	1.0
`getReader`	
Chrome	43
Edge	14
Firefox	65
Internet Explorer	**No**
Opera	30
Safari	10.1
WebView Android	43
Chrome Android	43
Firefox for Android	65
Opera Android	30
Safari on iOS	10.3
Samsung Internet	4.0
Deno	1.0
`locked`	
Chrome	43
Edge	14
Firefox	65
Internet Explorer	**No**
Opera	30
Safari	10.1
WebView Android	43
Chrome Android	43
Firefox for Android	65
Opera Android	30

Safari on iOS	10.3
Samsung Internet	4.0
Deno	1.0
`pipeThrough`	
Chrome	59
Edge	79
Firefox	**No**
Internet Explorer	**No**
Opera	46
Safari	10.1
WebView Android	59
Chrome Android	59
Firefox for Android	**No**
Opera Android	43
Safari on iOS	10.3
Samsung Internet	7.0
Deno	1.0
`pipeTo`	
Chrome	59
Edge	79
Firefox	**No**
Internet Explorer	**No**
Opera	46
Safari	10.1
WebView Android	59
Chrome Android	59
Firefox for Android	**No**

Opera Android	43
Safari on iOS	10.3
Samsung Internet	7.0
Deno	1.0
`tee`	
Chrome	43
Edge	79
Firefox	65
Internet Explorer	**No**
Opera	30
Safari	10.1
WebView Android	43
Chrome Android	43
Firefox for Android	65
Opera Android	30
Safari on iOS	10.3
Samsung Internet	4.0
Deno	1.0

🟩	Full support
🟥	No support

WritableStream

Report problems with this compatibility data on GitHub

`WritableStream`	
Chrome	59
Edge	16
Firefox	**No**
Internet Explorer	**No**
Opera	47
Safari	14.1
WebView Android	59
Chrome Android	59
Firefox for Android	**No**
Opera Android	44
Safari on iOS	14.5
Samsung Internet	7.0
Deno	1.0
`WritableStream()` constructor	
Chrome	59
Edge	16
Firefox	**No**
Internet Explorer	**No**
Opera	47
Safari	14.1
WebView Android	59
Chrome Android	59
Firefox for Android	**No**
Opera Android	44

Safari on iOS	14.5
Samsung Internet	7.0
Deno	1.0
`abort`	
Chrome	59
Edge	16
Firefox	**No**
Internet Explorer	**No**
Opera	47
Safari	14.1
WebView Android	59
Chrome Android	59
Firefox for Android	**No**
Opera Android	44
Safari on iOS	14.5
Samsung Internet	7.0
Deno	1.0
`close`	
Chrome	81
Edge	81
Firefox	**No**
Internet Explorer	**No**
Opera	68
Safari	14.1
WebView Android	81
Chrome Android	81
Firefox for Android	**No**

Opera Android	58
Safari on iOS	14.5
Samsung Internet	13.0
Deno	1.0
`getWriter`	
Chrome	59
Edge	16
Firefox	**No**
Internet Explorer	**No**
Opera	47
Safari	14.1
WebView Android	59
Chrome Android	59
Firefox for Android	**No**
Opera Android	44
Safari on iOS	14.5
Samsung Internet	7.0
Deno	1.0
`locked`	
Chrome	59
Edge	16
Firefox	**No**
Internet Explorer	**No**
Opera	47
Safari	14.1
WebView Android	59
Chrome Android	59

Firefox for Android	**No**
Opera Android	44
Safari on iOS	14.5
Samsung Internet	7.0
Deno	1.0

🟩	Full support
🟥	No support

See also

- Streams API concepts
- Using readable streams
- Using writable streams

Last modified: Aug 31, 2021, by MDN contributors