Global objects

These objects are available in all modules. The following variables may appear to be global but are not. They exist only in the scope of modules, see the module system documentation:

- `__dirname`
- `__filename`
- `exports`
- `module`
- `require()`

The objects listed here are specific to Node.js. There are built-in objects that are part of the JavaScript language itself, which are also globally accessible.

Class: `AbortController`

A utility class used to signal cancelation in selected `Promise`-based APIs. The API is based on the Web API `AbortController`.

```
const ac = new AbortController();

ac.signal.addEventListener('abort', () => console.log('Aborted!'),
                           { once: true });

ac.abort();

console.log(ac.signal.aborted);  // Prints True
```

`abortController.abort([reason])`

- `reason` {any} An optional reason, retrievable on the `AbortSignals` `reason` property.

Triggers the abort signal, causing the `abortController.signal` to emit the `'abort'` event.

`abortController.signal`

- Type: {AbortSignal}

Class: `AbortSignal`

- Extends: {EventTarget}

The `AbortSignal` is used to notify observers when the `abortController.abort()` method is called.

Static method: `AbortSignal.abort([reason])`

- `reason`: {any}
- Returns: {AbortSignal}

Returns a new already aborted `AbortSignal`.

Static method: `AbortSignal.timeout(delay)`

- `delay` {number} The number of milliseconds to wait before triggering the AbortSignal.

Returns a new `AbortSignal` which will be aborted in `delay` milliseconds.

Event: `'abort'` The `'abort'` event is emitted when the `abortController.abort()` method is called. The callback is invoked with a single object argument with a single `type` property set to `'abort'`:

```
const ac = new AbortController();

// Use either the onabort property...
ac.signal.onabort = () => console.log('aborted!');

// Or the EventTarget API...
ac.signal.addEventListener('abort', (event) => {
  console.log(event.type);  // Prints 'abort'
}, { once: true });

ac.abort();
```

The `AbortController` with which the `AbortSignal` is associated will only ever trigger the `'abort'` event once. We recommended that code check that the `abortSignal.aborted` attribute is `false` before adding an `'abort'` event listener.

Any event listeners attached to the `AbortSignal` should use the `{ once: true }` option (or, if using the `EventEmitter` APIs to attach a listener, use the `once()` method) to ensure that the event listener is removed as soon as the `'abort'` event is handled. Failure to do so may result in memory leaks.

`abortSignal.aborted`

- Type: {boolean} True after the `AbortController` has been aborted.

`abortSignal.onabort`

- Type: {Function}

An optional callback function that may be set by user code to be notified when the `abortController.abort()` function has been called.

`abortSignal.reason`

- Type: {any}

An optional reason specified when the `AbortSignal` was triggered.

```
const ac = new AbortController();
ac.abort(new Error('boom!'));
console.log(ac.signal.reason);  // Error('boom!');
```

`abortSignal.throwIfAborted()` If `abortSignal.aborted` is `true`, throws `abortSignal.reason`.

Class: `Blob`

See {Blob}.

Class: `Buffer`

- {Function}

Used to handle binary data. See the buffer section.

Class: `ByteLengthQueuingStrategy`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ByteLengthQueuingStrategy`.

`__dirname`

This variable may appear to be global but is not. See `__dirname`.

`__filename`

This variable may appear to be global but is not. See `__filename`.

`atob(data)`

> Stability: 3 - Legacy. Use `Buffer.from(data, 'base64')` instead.

Global alias for `buffer.atob()`.

`BroadcastChannel`

See {BroadcastChannel}.

`btoa(data)`

> Stability: 3 - Legacy. Use `buf.toString('base64')` instead.

Global alias for `buffer.btoa()`.

`clearImmediate(immediateObject)`

`clearImmediate` is described in the timers section.

`clearInterval(intervalObject)`

`clearInterval` is described in the timers section.

`clearTimeout(timeoutObject)`

`clearTimeout` is described in the timers section.

Class: `CompressionStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `CompressionStream`.

`console`

- {Object}

Used to print to stdout and stderr. See the `console` section.

Class: `CountQueuingStrategy`

> Stability: 1 - Experimental.

A browser-compatible implementation of `CountQueuingStrategy`.

`Crypto`

> Stability: 1 - Experimental. Enable this API with the `--experimental-global-webcrypto` CLI flag.

A browser-compatible implementation of {Crypto}. This global is available only if the Node.js binary was compiled with including support for the `crypto` module.

`crypto`

> Stability: 1 - Experimental. Enable this API with the `--experimental-global-webcrypto` CLI flag.

A browser-compatible implementation of the Web Crypto API.

`CryptoKey`

> Stability: 1 - Experimental. Enable this API with the `--experimental-global-webcrypto` CLI flag.

A browser-compatible implementation of {CryptoKey}. This global is available only if the Node.js binary was compiled with including support for the `crypto` module.

Class: `DecompressionStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `DecompressionStream`.

`Event`

A browser-compatible implementation of the `Event` class. See `EventTarget` and `Event` API for more details.

`EventTarget`

A browser-compatible implementation of the `EventTarget` class. See `EventTarget` and `Event` API for more details.

`exports`

This variable may appear to be global but is not. See `exports`.

`fetch`

> Stability: 1 - Experimental. Disable this API with the `--no-experimental-fetch` CLI flag.

A browser-compatible implementation of the `fetch()` function.

Class `FormData`

> Stability: 1 - Experimental. Disable this API with the `--no-experimental-fetch` CLI flag.

A browser-compatible implementation of {FormData}.

`global`

- {Object} The global namespace object.

In browsers, the top-level scope is the global scope. This means that within the browser `var something` will define a new global variable. In Node.js this is different. The top-level scope is not the global scope; `var something` inside a Node.js module will be local to that module.

Class `Headers`

> Stability: 1 - Experimental. Disable this API with the `--no-experimental-fetch` CLI flag.

A browser-compatible implementation of {Headers}.

`MessageChannel`

The `MessageChannel` class. See `MessageChannel` for more details.

`MessageEvent`

The `MessageEvent` class. See `MessageEvent` for more details.

`MessagePort`

The `MessagePort` class. See `MessagePort` for more details.

`module`

This variable may appear to be global but is not. See `module`.

`performance`

The `perf_hooks.performance` object.

`process`

- {Object}

The process object. See the `process` object section.

`queueMicrotask(callback)`

- `callback` {Function} Function to be queued.

The `queueMicrotask()` method queues a microtask to invoke `callback`. If `callback` throws an exception, the `process` object `'uncaughtException'` event will be emitted.

The microtask queue is managed by V8 and may be used in a similar manner to the `process.nextTick()` queue, which is managed by Node.js. The `process.nextTick()` queue is always processed before the microtask queue within each turn of the Node.js event loop.

```
// Here, `queueMicrotask()` is used to ensure the 'load' event is always
// emitted asynchronously, and therefore consistently. Using
// `process.nextTick()` here would result in the 'load' event always emitting
// before any other promise jobs.

DataHandler.prototype.load = async function load(key) {
  const hit = this._cache.get(key);
  if (hit !== undefined) {
    queueMicrotask(() => {
      this.emit('load', hit);
    });
    return;
  }

  const data = await fetchData(key);
  this._cache.set(key, data);
  this.emit('load', data);
};
```

Class: `ReadableByteStreamController`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ReadableByteStreamController`.

Class: `ReadableStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ReadableStream`.

Class: `ReadableStreamBYOBReader`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ReadableStreamBYOBReader`.

Class: `ReadableStreamBYOBRequest`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ReadableStreamBYOBRequest`.

Class: `ReadableStreamDefaultController`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ReadableStreamDefaultController`.

Class: `ReadableStreamDefaultReader`

> Stability: 1 - Experimental.

A browser-compatible implementation of `ReadableStreamDefaultReader`.

`require()`

This variable may appear to be global but is not. See `require()`.

`Response`

> Stability: 1 - Experimental. Disable this API with the `--no-experimental-fetch` CLI flag.

A browser-compatible implementation of {Response}.

`Request`

> Stability: 1 - Experimental. Disable this API with the `--no-experimental-fetch` CLI flag.

A browser-compatible implementation of {Request}.

`setImmediate(callback[, ...args])`

`setImmediate` is described in the timers section.

`setInterval(callback, delay[, ...args])`

`setInterval` is described in the timers section.

`setTimeout(callback, delay[, ...args])`

`setTimeout` is described in the timers section.

`structuredClone(value[, options])`

The WHATWG `structuredClone` method.

`SubtleCrypto`

> Stability: 1 - Experimental. Enable this API with the `--experimental-global-webcrypto` CLI flag.

A browser-compatible implementation of {SubtleCrypto}. This global is available only if the Node.js binary was compiled with including support for the `crypto` module.

`DOMException`

The WHATWG `DOMException` class. See `DOMException` for more details.

`TextDecoder`

The WHATWG `TextDecoder` class. See the `TextDecoder` section.

Class: `TextDecoderStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `TextDecoderStream`.

`TextEncoder`

The WHATWG `TextEncoder` class. See the `TextEncoder` section.

Class: `TextEncoderStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `TextEncoderStream`.

Class: `TransformStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `TransformStream`.

Class: `TransformStreamDefaultController`

> Stability: 1 - Experimental.

A browser-compatible implementation of `TransformStreamDefaultController`.

`URL`

The WHATWG `URL` class. See the `URL` section.

`URLSearchParams`

The WHATWG `URLSearchParams` class. See the `URLSearchParams` section.

`WebAssembly`

- {Object}

The object that acts as the namespace for all W3C WebAssembly related functionality. See the Mozilla Developer Network for usage and compatibility.

Class: `WritableStream`

> Stability: 1 - Experimental.

A browser-compatible implementation of `WritableStream`.

Class: `WritableStreamDefaultController`

> Stability: 1 - Experimental.

A browser-compatible implementation of `WritableStreamDefaultController`.

Class: `WritableStreamDefaultWriter`

> Stability: 1 - Experimental.

A browser-compatible implementation of `WritableStreamDefaultWriter`.