Standard Standard Ecma-XXX

1st Edition / 6 October 2025

Minimum Common Web Platform API

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

Ecma International
Rue du Rhone 114 CH-1204 Geneva
Tel: +41 22 849 6000
Fax: +41 22 849 6001
Web: https://www.ecma-international.org

Table of Contents

page

Introduction

Contributing to this Specification

This version:
https://min-common-api.proposal.wintertc.org/

Issue Tracking:
GitHub

Editor:
James M Snell (Cloudflare)

Introduction

There is a wide base of JavaScript runtime environments being used beyond web browsers, specifically in web server and edge platforms. An important part of the reason for this is the fact that JavaScript can be used both in the server and in the client side, which reduces the specialization needed to work in different parts of a single codebase, and allows reusing code across the server and client side.

But since code running in web browsers makes up the vast majority of JavaScript code, this is helped along if such runtimes support the same APIs as web browsers. So, unsurprisingly, more and more runtime environments have started supporting web platform APIs. However, the increase in such environments, as well as the different choices of web platform APIs, resulted in poor interoperability across such environments.

As such, this Ecma Standard defines the Minimum Common Web API specification, which lists a curated minimum subset of web platform APIs for server-side and edge runtimes to implement if they aim to be web-interoperable. This is the first edition of the standard, corresponding to the 2025 snapshot, and an additional snapshot will be published every year.

1. Scope

This Standard defines the 2025 snapshot of the Minimum Common Web Platform API, a curated subset of APIs defined by web platform standards from W3C and WHATWG, which is intended to define a minimum set of capabilities common to Browser and Non-Browser JavaScript-based runtime environments.

2. Conformance

A conforming implementation of the Minimum Common Web Platform API shall conform to ECMA-262, and additionally shall provide the interfaces and properties listed in this specification, according to their definition in the corresponding W3C or WHATWG standard.

Runtime-specific extensions to any Web Platform API may be implemented by conforming runtimes. Such extensions shall be defined so that their use neither contradicts, nor causes the non-conformance of, normative functionality of any Web Platform API. It is important to carefully consider use of such extensions, as it reduces interoperability and portability of code across runtimes.

This specification does not prohibit implementing additional Web Platform APIs beyond those listed here.

Note: For example, the `Performance` API could be extended with additional methods or properties beyond those defined in the [HR-TIME] specifications, such as those defined in the [PERFORMANCE-TIMELINE] or [USER-

TIMING] specifications.

3. Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

References

Normative References

[COMPRESSION]
Adam Rice. *Compression Standard*. Living Standard. URL: https://compression.spec.whatwg.org/

[CONSOLE]
Dominic Farolino; Robert Kowalski; Terin Stock. *Console Standard*. Living Standard. URL: https://console.spec.whatwg.org/

[DOM]
Anne van Kesteren. *DOM Standard*. Living Standard. URL: https://dom.spec.whatwg.org/

[ECMASCRIPT]
ECMAScript Language Specification. URL: https://tc39.es/ecma262/multipage/

[ENCODING]
Anne van Kesteren. *Encoding Standard*. Living Standard. URL: https://encoding.spec.whatwg.org/

[FETCH]
Anne van Kesteren. *Fetch Standard*. Living Standard. URL: https://fetch.spec.whatwg.org/

[FILEAPI]
Marijn Kruisselbrink. *File API*. URL: https://w3c.github.io/FileAPI/

[HR-TIME]
Yoav Weiss. *High Resolution Time*. URL: https://w3c.github.io/hr-time/

[HTML]
Anne van Kesteren; et al. *HTML Standard*. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[RFC7231]
R. Fielding, Ed.; M. Nottingham, Ed.; J. Reschke, Ed.. *HTTP Semantics*. June 2022. Internet Standard. URL: https://httpwg.org/specs/rfc9110.html

[STREAMS]
Adam Rice; et al. *Streams Standard*. Living Standard. URL: https://streams.spec.whatwg.org/

[URL]
Anne van Kesteren. *URL Standard*. Living Standard. URL: https://url.spec.whatwg.org/

[URLPATTERN]
Ben Kelly; Jeremy Roman; 宍戸俊哉 (Shunya Shishido). *URL Pattern Standard*. Living Standard. URL: https://urlpattern.spec.whatwg.org/

[WASM-JS-API-2]
. Ms2ger. *WebAssembly JavaScript Interface*. URL: https://webassembly.github.io/spec/js-api/

[WASM-WEB-API-2]
. Ms2ger. *WebAssembly Web API*. URL: https://webassembly.github.io/spec/web-api/

[WEBCRYPTO]
Daniel Huigens. *Web Cryptography Level 2*. URL: https://w3c.github.io/webcrypto/

[WEBIDL]
Edgar Chen; Timothy Gu. *Web IDL Standard*. Living Standard. URL: https://webidl.spec.whatwg.org/

[XHR]
Anne van Kesteren. *XMLHttpRequest Standard*. Living Standard. URL: https://xhr.spec.whatwg.org/

Informative References

[PERFORMANCE-TIMELINE]
Nicolas Pena Moreno. *Performance Timeline*. URL: https://w3c.github.io/performance-timeline/

[USER-TIMING]
Nicolas Pena Moreno. *User Timing Level 3*. URL: https://w3c.github.io/user-timing/

4. Terms and definitions

4.1. *Web Platform*

The Web Platform is the combination of technology standards defined by organizations such as the W3C, the WHATWG, and others as implemented by Web Browsers.

4.2. *Web-interoperable Runtime*

Any ECMAScript-based application runtime environment that implements the subset of Web Platform APIs outlined in this specification.

While this term is intentionally broad to also encompass Web Browsers, the primary focus here is on outlining expectations for non-browser runtimes.

5. Common API Index

All Web-interoperable Runtimes conforming to this specification shall implement each of the following Web Platform APIs. These should be implemented in accordance with their normative requirements except where modified here. Where any conforming runtime environment chooses (either by necessity or otherwise) to diverge from a normative requirement of the specification, clear explanations of such divergence shall be made clearly and readily available in the documentation.

All of the following interfaces shall be exposed on the global object accessible through **`globalThis`**, unless otherwise specified in this specification:

- **`AbortController`** [DOM]
- **`AbortSignal`** [DOM]
- **`Blob`** [FILEAPI]
- **`ByteLengthQueuingStrategy`** [STREAMS]
- **`CompressionStream`** [COMPRESSION]
- **`CountQueuingStrategy`** [STREAMS]

- **`Crypto`** [WEBCRYPTO]
- **`CryptoKey`** [WEBCRYPTO]
- **`CustomEvent`** [HTML]
- **`DecompressionStream`** [COMPRESSION]
- **`DOMException`** [WEBIDL]
- **`ErrorEvent`** [HTML]
- **`Event`** [DOM]
- **`EventTarget`** [DOM]
- **`File`** [FILEAPI]
- **`FormData`** [XHR]
- **`Headers`** [FETCH]
- **`MessageChannel`** [HTML]
- **`MessageEvent`** [HTML]
- **`MessagePort`** [HTML]
- **`Performance`** [HR-TIME]
- **`PromiseRejectionEvent`** [HTML]
- **`ReadableByteStreamController`** [STREAMS]
- **`ReadableStream`** [STREAMS]
- **`ReadableStreamBYOBReader`** [STREAMS]
- **`ReadableStreamBYOBRequest`** [STREAMS]
- **`ReadableStreamDefaultController`** [STREAMS]
- **`ReadableStreamDefaultReader`** [STREAMS]
- **`Request`** [FETCH]
- **`Response`** [FETCH]
- **`SubtleCrypto`** [WEBCRYPTO]
- **`TextDecoder`** [ENCODING]
- **`TextDecoderStream`** [ENCODING]
- **`TextEncoder`** [ENCODING]
- **`TextEncoderStream`** [ENCODING]
- **`TransformStream`** [STREAMS]

- `TransformStreamDefaultController` [STREAMS]
- `URL` [URL]
- `URLPattern` [URLPATTERN]
- `URLSearchParams` [URL]
- `WebAssembly.Global` [WASM-JS-API-2]
- `WebAssembly.Instance` [WASM-JS-API-2]
- `WebAssembly.Memory` [WASM-JS-API-2]
- `WebAssembly.Module` [WASM-JS-API-2]
- `WebAssembly.Table` [WASM-JS-API-2]
- `WebAssembly.Tag` [WASM-JS-API-2]
- `WebAssembly.Exception` [WASM-JS-API-2]
- `WebAssembly.CompileError` [WASM-JS-API-2]
- `WebAssembly.LinkError` [WASM-JS-API-2]
- `WebAssembly.RuntimeError` [WASM-JS-API-2]
- `WritableStream` [STREAMS]
- `WritableStreamDefaultController` [STREAMS]
- `WritableStreamDefaultWriter` [STREAMS]

All of the following methods and properties shall be exposed on the global object accessible through **`globalThis`**, unless otherwise specified in this specification:

- `globalThis` [ECMASCRIPT]
- **`globalThis`**`.atob()` [HTML]
- **`globalThis`**`.btoa()` [HTML]
- **`globalThis`**`.clearTimeout()` [HTML]
- **`globalThis`**`.clearInterval()` [HTML]
- **`globalThis`**`.console` [CONSOLE]
- **`globalThis`**`.crypto` [WEBCRYPTO]
- **`globalThis`**`.fetch()` [FETCH]
- **`globalThis`**`.navigator.userAgent` [HTML]
- **`globalThis`**`.onerror` [HTML]
- **`globalThis`**`.onunhandledrejection` [HTML]
- **`globalThis`**`.onrejectionhandled` [HTML]

- **globalThis.performance** [HR-TIME]
- **globalThis.queueMicrotask()** [HTML]
- **globalThis.reportError()** [HTML]
- **globalThis.self** [HTML]
- **globalThis.setTimeout()** [HTML]
- **globalThis.setInterval()** [HTML]
- **globalThis.structuredClone()** [HTML]
- **globalThis.WebAssembly.compile()** [WASM-JS-API-2]
- **globalThis.WebAssembly.compileStreaming()** [WASM-WEB-API-2]
- **globalThis.WebAssembly.instantiate()** [WASM-JS-API-2]
- **globalThis.WebAssembly.instantiateStreaming()** [WASM-WEB-API-2]
- **globalThis.WebAssembly.JSTag** [WASM-JS-API-2]
- **globalThis.WebAssembly.validate()** [WASM-JS-API-2]

Web-interoperable runtimes that support workers shall also expose **onerror**, **onunhandledrejection**, **onrejectionhandled** and **self** on the worker's **globalThis**, unless otherwise specified in this specification. [HTML]

6. The Global Scope

The exact type of the global scope (**globalThis**) can vary across runtimes. Most Web Platform APIs are defined in terms that assume Web Browser environments that specifically expose types like **Window**, **WorkerGlobalScope**, and so forth. To simplify conformance, all interfaces, methods, and properties defined by this specification shall be exposed on the runtime's relevant global scope (e.g., **globalThis.crypto**, **globalThis.ReadableStream**, etc).

With many runtimes, adding a new global-scoped property can introduce breaking changes when the new global conflicts with existing application code. Many Web Platform APIs define global properties using the **readonly** attribute. [WEBIDL] To avoid introducing breaking changes, runtimes conforming to this specification may choose to ignore the **readonly** attribute for properties being added to the global scope. This allows users of these runtimes to delete or overwrite these properties if they conflict with existing application code.

The global object on **Window**-like and worker environments should always be an instance of **EventTarget**. Web-interoperable runtimes should follow the report an exception algorithm, and the JavaScript HostPromiseRejectionTracker host hook, as defined in [HTML]. This includes firing the **error**, **unhandledrejection** and **rejectionhandled** events on the global object.

In cases where it is not possible to have the global object be an instance of **EventTarget** due to legacy reasons, the relevant events shall still be fired through a suitable alternative mechanism available at the global scope. This mechanism shall provide at least the same information that is provided by the relevant event interfaces if the global object were to be an **EventTarget**. Such runtimes shall not support the **onerror**, **onunhandledrejection** and **onrejectionhandled** global properties. Such runtimes are not required to implement the **ErrorEvent** and **PromiseRejectionEvent** interfaces.

Note: For example, in Node.js the global object does not implement **EventTarget**, and the relevant events are fired on the **globalThis.process** object with the names **uncaughtException**, **unhandledRejection** and

rejectionHandled, respectively.

7. Requirements for default User-Agent value

The default `User-Agent` value is provided such that application code can reliably identify the runtime within which it is running. The value shall be a string conforming to the **User-Agent** construction in [RFC7231]:

```
User-Agent      = product *( RWS ( product / comment ) )
product         = token ["/" product-version]
product-version = token
```

The default `User-Agent` value should be treated as a single, complete, opaque, unstructured value. It is recommended that the value be limited to a single **product** token excluding the optional **product-version**. The value should not include any **comment** components.

Note: For instance, **navigator.userAgent** could be set to **'MyRuntime'**.

Index

Terms defined by this specification

Terms defined by reference

- [HTML] defines the following terms:
  - ErrorEvent
  - MessageChannel
  - MessageEvent
  - MessagePort
  - PromiseRejectionEvent
  - Window
  - WorkerGlobalScope
  - atob(data)
  - btoa(data)
  - clearInterval(id)
  - clearTimeout(id)
  - error
  - navigator
  - onerror (for GlobalEventHandlers)
  - onerror (for WorkerGlobalScope)
  - onrejectionhandled (for WindowEventHandlers)
  - onrejectionhandled (for WorkerGlobalScope)
  - onunhandledrejection (for WindowEventHandlers)
  - onunhandledrejection (for WorkerGlobalScope)
  - queueMicrotask(callback)
  - rejectionhandled
  - report an exception
  - reportError(e)
  - self (for Window)
  - self (for WorkerGlobalScope)
  - setInterval(handler, timeout, ...arguments)
  - setTimeout(handler, timeout, ...arguments)
  - structuredClone(value, options)
  - unhandledrejection
  - userAgent
- [STREAMS] defines the following terms:
  - ByteLengthQueuingStrategy
  - CountQueuingStrategy
  - ReadableByteStreamController
  - ReadableStream
  - ReadableStreamBYOBReader
  - ReadableStreamBYOBRequest
  - ReadableStreamDefaultController
  - ReadableStreamDefaultReader
  - TransformStream
  - TransformStreamDefaultController
  - WritableStream
  - WritableStreamDefaultController
  - WritableStreamDefaultWriter
- [URL] defines the following terms:
  - URL
  - URLSearchParams
- [URLPATTERN] defines the following terms:
  - URLPattern
- [WASM-JS-API-2] defines the following terms:
  - CompileError
  - Exception
  - Global
  - Instance
  - JSTag
  - LinkError
  - Memory
  - Module
  - RuntimeError
  - Table

    - Tag
    - WebAssembly
    - compile(bytes)
    - instantiate(bytes)
    - validate(bytes)
- [WASM-WEB-API-2] defines the following terms:
    - compileStreaming(source)
    - instantiateStreaming(source)
- [WEBCRYPTO] defines the following terms:
    - Crypto
    - CryptoKey
    - SubtleCrypto
    - crypto
- [WEBIDL] defines the following terms:
    - DOMException
    - read only
- [XHR] defines the following terms:
    - FormData

Copyright & Software License

Ecma International
Rue du Rhone 114
CH-1204 Geneva
Tel: +41 22 849 6000
Fax: +41 22 849 6001
Web: https://ecma-international.org/

Copyright Notice

Software License

FOR INFORMATION REGARDING THE LICENSING OF PATENT CLAIMS THAT ARE REQUIRED TO IMPLEMENT ECMA INTERNATIONAL STANDARDS.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the authors nor Ecma International may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE ECMA INTERNATIONAL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ECMA INTERNATIONAL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.