

Geolocation API

W3C Recommendation 01 September 2022



▼ More details about this document

This version:

<https://www.w3.org/TR/2022/REC-geolocation-20220901/>

Latest published version:

<https://www.w3.org/TR/geolocation/>

Latest editor's draft:

<https://w3c.github.io/geolocation-api/>

History:

<https://www.w3.org/standards/history/geolocation>

[Commit history](#)

Test suite:

<https://wpt.live/geolocation-API/>

Implementation report:

https://w3c.github.io/geolocation-api/reports/PR_imp_report.html

Editors:

Marcos Cáceres ([W3C](#))

Reilly Grant ([Google](#))

Former editor:

Andrei Popescu ([Google Inc.](#))

Feedback:

[GitHub w3c/geolocation-api](#) ([pull requests](#), [new issue](#), [open issues](#))

Errata:

[Errata exists.](#)

Browser support:

caniuse.com

See also [translations](#).

Copyright © 2022 [W3C](#)[®] ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C [liability](#), [trademark](#) and [permissive document license](#) rules apply.

Abstract

The Geolocation API provides access to geographical location information associated with the hosting device.

Status of This Document

This section describes the status of this document at the time of its publication. A list of current [W3C publications](#) and the latest revision of this technical report can be found in the [W3C technical reports index](#) at <https://www.w3.org/TR/>.

The Devices and Sensors Working Group is updating this specification in the hope of making it a "living standard". As such, we've dropped the "Editions" and aim to continue publishing updated [W3C Recommendations](#) of this specification as we add new features or fix bugs.

This document was published by the [Devices and Sensors Working Group](#) as a Recommendation using the [Recommendation track](#).

[W3C](#) recommends the wide deployment of this specification as a standard for the Web.

A [W3C Recommendation](#) is a specification that, after extensive consensus-building, is endorsed by [W3C](#) and its Members, and has commitments from Working Group members to [royalty-free licensing](#) for implementations. Future updates to this Recommendation may incorporate [new features](#).

This document was produced by a group operating under the [W3C Patent Policy](#). [W3C](#) maintains a [public list of any patent disclosures](#) made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains [Essential Claim\(s\)](#) must disclose the information in accordance with [section 6 of the W3C Patent Policy](#).

This document is governed by the [2 November 2021 W3C Process Document](#).

Table of Contents

Abstract

Status of This Document

1. Introduction

- 1.1 Scope
- 1.2 Change log

2. Examples

- 2.1 Get current position
- 2.2 Watch a position
- 2.3 Stop watching a position
- 2.4 Handling errors
- 2.5 Using `maximumAge` as cache control
- 2.6 Using `timeout`
- 2.7 Enabling the API in third-party contexts

- 3. Privacy considerations**
 - 3.1 User consent
 - 3.2 Privacy considerations for recipients of location information
 - 3.3 Implementation considerations
 - 3.4 Checking permission to use the API
- 4. Security considerations**
- 5. Extensions to the Navigator interface**
- 6. Geolocation interface and callbacks**
 - 6.1 Internal slots
 - 6.2 `getCurrentPosition()` method
 - 6.3 `watchPosition()` method
 - 6.4 `clearWatch()` method
 - 6.5 Request a position
 - 6.6 Acquire a position
 - 6.7 Call back with error
- 7. PositionOptions dictionary**
 - 7.1 `enableHighAccuracy` member
 - 7.2 `timeout` member
 - 7.3 `maximumAge` member
- 8. GeolocationPosition interface**
 - 8.1 `coords` attribute
 - 8.2 `timestamp` attribute
 - 8.3 Internal slots
 - 8.4 Task sources
- 9. GeolocationCoordinates interface**
 - 9.1 `latitude`, `longitude`, and `accuracy` attributes
 - 9.2 `altitude` and `altitudeAccuracy` attributes
 - 9.3 `heading` attribute
 - 9.4 `speed` attribute
 - 9.5 Constructing a `GeolocationPosition`
- 10. GeolocationPositionError interface**
 - 10.1 Constants
 - 10.2 `code` attribute
 - 10.3 `message` attribute
- 11. Permissions policy**

12. Conformance

A. IDL Index

B. Index

B.1 Terms defined by this specification

B.2 Terms defined by reference

C. Acknowledgments

D. References

D.1 Normative references

D.2 Informative references

§ 1. Introduction

This section is non-normative.

The *Geolocation API* defines a high-level interface to location information associated only with the device hosting the implementation. Common sources of location information include Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs, as well as user input. The API itself is agnostic of the underlying location information sources, and no guarantee is given that the API returns the device's actual location.

If an end user [grants permission](#), the *Geolocation API*:

- Provides location data as latitude, longitude, altitude, speed, and heading, as well as the accuracy of the acquired location data, and the approximate time for when the position was acquired via the [GeolocationPosition](#) interface.
- Supports "one-shot" position updates via the [getCurrentPosition\(\)](#) method and the ability to receive updates for when the position of the hosting device significantly changes via the [watchPosition\(\)](#) method.
- Using the [PositionOptions](#)'s [maximumAge](#), allows an application to request a cached position whose age is no greater than a specified value (only the last position is cached).
- Provides a way for the application to receive updates about errors, as a [GeolocationPositionError](#), that have occurred while [acquiring a position](#).
- And through [enableHighAccuracy](#), supports requesting "high accuracy" position data, though the request can be ignored by the user agent.

§ 1.1 Scope

This section is non-normative.

This specification is limited to providing a scripting API for retrieving geographic position information associated with a hosting device. The geographic position information is provided in terms of World Geodetic System coordinates [WGS84]. It does not include providing a markup language of any kind, nor does not include defining a new URL scheme for building URLs that identify geographic locations.

§ 1.2 Change log

This section is non-normative.

Since First Public Working Draft in 2021, the *Geolocation API* has received the following normative changes:

- [fix "Queue a task" / "in parallel" usage \(#118\)](#)
- [Suggest permission lifetime \(#108\)](#)
- [Handle OS-level permission change \(#109\)](#)
- [Switch DOMTimeStamp to EpochTimeStamp \(#104\)](#)
- [Return 0 when watchPosition\(\) errors \(#100\)](#)
- [Callback with error if doc is not fully active \(#97\)](#)
- [Gracefully handle documents that are not fully active \(#90\)](#)
- [Add "geolocation task queue" \(#87\)](#)

Since publication of the Second Edition in 2016, this specification has received the following changes:

- [Request a position](#) only proceeds when a document is visible, or the document becomes visible.
- Clarified how caching works as part of [acquiring a position](#): only last position is cached, and can be evicted at any time.
- Now relies on the [Permissions](#) specification to handle permission grants, and user interface requirements.
- The errorCallback is now nullable.
- The API can be controlled by [Permissions Policy](#), which restricts how/where the API is exposed to web pages.
- The callbacks are no longer treated as "EventHandler" objects (i.e., objects that have a `.handleEvent()` method), but are now exclusively treated as IDL callback functions.
- The API is now only exposed in Secure Contexts (i.e., only available in HTTPS).
- The interfaces no longer use [WebIDL]'s legacy [NoInterfaceObject], so Geolocation and other interface of this spec are now in the global scope. Also, the interfaces were renamed from NavigatorGeolocation* to just Geolocation*.

See the [commit history](#) for a complete list of changes.

§ 2. Examples

This section is non-normative.

The API is designed to enable both "one-shot" position requests and repeated position updates. The following examples illustrate common use cases.

§ 2.1 Get current position

This section is non-normative.

Request the user's current location. If the user allows it, you will get back a position object.

EXAMPLE 1: A one-shot position request

```
navigator.geolocation.getCurrentPosition(position => {  
  const { latitude, longitude } = position.coords;  
  // Show a map centered at Latitude / Longitude.  
});
```

§ 2.2 Watch a position

This section is non-normative.

Request the ability to watch user's current location. If the user allows it, you will get back continuous updates of the user's position.

EXAMPLE 2: Watching a position for repeated updates

```
const watchId = navigator.geolocation.watchPosition(position => {  
  const { latitude, longitude } = position.coords;  
  // Show a map centered at Latitude / Longitude.  
});
```

§ 2.3 Stop watching a position

This section is non-normative.

Stop watching for position changes by calling the [clearWatch\(\)](#) method.

EXAMPLE 3: Using clearWatch()

```
const watchId = navigator.geolocation.watchPosition(
  position => console.log(position)
);

function buttonClickHandler() {
  // Cancel the updates when the user clicks a button.
  navigator.geolocation.clearWatch(watchId);
}
```

And a HTML button that when pressed stops watching the position.

```
<button onclick="buttonClickHandler()">
  Stop watching location
</button>
```

§ 2.4 Handling errors

This section is non-normative.

When an error occur, the second argument of the [watchPosition\(\)](#) or [getCurrentPosition\(\)](#) method gets called with a [GeolocationPositionError](#) error, which can help you figure out what might have gone wrong.

EXAMPLE 4: Handling errors

```
// Request repeated updates.
const watchId = navigator.geolocation.watchPosition(
  scrollMap, handleError
);

function scrollMap(position) {
  const { latitude, longitude } = position.coords;
  // Scroll map to latitude / longitude.
}

function handleError(error) {
  // Display error based on the error code.
  const { code } = error;
  switch (code) {
    case GeolocationPositionError.TIMEOUT:
      // Handle timeout.
      break;
    case GeolocationPositionError.PERMISSION_DENIED:
      // User denied the request.
      break;
    case GeolocationPositionError.POSITION_UNAVAILABLE:
      // Position not available.
      break;
  }
}
```

§ 2.5 Using `maximumAge` as cache control

This section is non-normative.

By default, the API always attempts to return a cached position so long as it has a previously acquired position. In this example, we accept a position whose age is no greater than 10 minutes. If the user agent does not have a fresh enough cached position object, it automatically acquires a new position.

EXAMPLE 5: Getting cached position

```
navigator.geolocation.getCurrentPosition(  
  successCallback,  
  console.error,  
  { maximumAge: 600_000 }  
);  
  
function successCallback(position) {  
  // By using the 'maximumAge' member above, the position  
  // object is guaranteed to be at most 10 minutes old.  
}
```

§ 2.6 Using timeout

If you require location information in a time sensitive manner, you can use the [PositionOptions timeout](#) member to limit the amount of time you are willing to wait to [acquire a position](#).

EXAMPLE 6: Timing out a position request

```
// Request a position. We are only willing to wait 10
// seconds for it.
navigator.geolocation.getCurrentPosition(
    successCallback,
    errorCallback,
    { timeout: 10_000 }
);

function successCallback(position) {
    // Request finished in under 10 seconds...
}

function errorCallback(error) {
    switch (error.code) {
        case GeolocationPositionError.TIMEOUT:
            // We didn't get it in a timely fashion.
            doFallback();
            // Acquire a new position object,
            // as long as it takes.
            navigator.geolocation.getCurrentPosition(
                successCallback, errorCallback
            );
            break;
        case "...": // treat the other error cases.
    }
}

function doFallback() {}
```

§ 2.7 Enabling the API in third-party contexts

This section is non-normative.

The default allowlist of 'self' allows Geolocation API usage in same-origin nested frames but prevents third-party content from using the API.

Third-party usage can be selectively enabled by adding the allow="geolocation" attribute to an iframe element:

EXAMPLE 7: Enabling the Geolocation API in an iframe

```
<iframe
  src="https://third-party.com"
  allow="geolocation">
</iframe>
```

Alternatively, the API can be disabled in a first-party context by specifying an HTTP response header:

EXAMPLE 8: Permissions Policy over HTTP

```
Permissions-Policy: geolocation=()
```

See [Permissions Policy](#) for more details about the Permissions-Policy HTTP header.

§ 3. Privacy considerations

This section is non-normative.

The API defined in this specification is used to retrieve the geographic location of a hosting device. In almost all cases, this information also discloses the location of the user of the device, thereby potentially compromising the user's privacy.

§ 3.1 User consent

This section is non-normative.

The *Geolocation API* is a powerful feature that requires express permission from an end-user before any location data is shared with a web application. This requirement is normatively enforced by the [check permission](#) steps on which the [getCurrentPosition\(\)](#) and [watchPosition\(\)](#) methods rely.

An end-user will generally give express permission through a user interface, which usually present a range of permission lifetimes that the end-user can choose from. The choice of lifetimes vary across user agents, but they are typically time-based (e.g., "a day"), or until browser is closed, or the user might even be given the choice for the permission to be granted indefinitely. The permission lifetimes dictate how long a user agent grants a permission before that permission is automatically reverted back to its default permission state, prompting the end-user to make a new choice upon subsequent use.

Although the granularity of the permission lifetime varies across user-agents, this specification urges user agents to limit the lifetime to a single browsing session by default (see [3.4 Checking permission to use the API](#) for normative requirements).

§ 3.2 Privacy considerations for recipients of location information

This section is non-normative.

NOTE: Developers' responsibility with this sensitive data

This section applies to "recipients", which generally means developers utilizing the *Geolocation API*. Although it's impossible for the user agent, or this specification, to enforce these requirements, developers need to read this section carefully and do their best to adhere to the suggestions below. Developers need to be aware that there might be privacy laws in their jurisdictions that can govern the usage and access to users' location data.

Recipients ought to only request position information when necessary, and only use the location information for the task for which it was provided to them. Recipients ought to dispose of location information once that task is completed, unless expressly permitted to retain it by the user. Recipients need to also take measures to protect this information against unauthorized access. If location information is stored, users need to be allowed to update and delete this information.

The recipients of location information need to refrain from retransmitting the location information without the user's express permission. Care needs to be taken when retransmitting and the use of encryption is encouraged.

Recipients ought to clearly and conspicuously disclose the fact that they are collecting location data, the purpose for the collection, how long the data is retained, how the data is secured, how the data is shared if it is shared, how users can access, update and delete the data, and any other choices that users have with respect to the data. This disclosure needs to include an explanation of any exceptions to the guidelines listed above.

§ 3.3 Implementation considerations

This section is non-normative.

Implementers are advised to consider the following aspects that can negatively affect the privacy of their users: in certain cases, users can inadvertently grant permission to the user agent to disclose their location to websites. In other cases, the content hosted at a certain URL changes in such a way that the previously granted location permissions no longer apply as far as the user is concerned. Or the users might simply change their minds.

Predicting or preventing these situations is inherently difficult. Mitigation and in-depth defensive measures are an implementation responsibility and not prescribed by this specification. However, in designing these measures,

implementers are advised to enable user awareness of location sharing, and to provide access to user interfaces that enable revocation of permissions.

§ 3.4 Checking permission to use the API

► MDN 

The *Geolocation API* is a default powerful feature identified by the name "***geolocation***".

When ***checking permission*** to use the API, a user agent *MAY* suggest time-based permission lifetimes, such as "24 hours", "1 week", or choose to remember the permission grant indefinitely. However, it is *RECOMMENDED* that a user agent prioritize restricting the permission lifetime to a single session: This can be, for example, until the realm is destroyed, the end-user navigates away from the origin, or the relevant browser tab is closed.

§ 4. Security considerations

There are no security considerations associated with Geolocation API at the time of publication. However, readers are advised to read the [3. Privacy considerations](#).

§ 5. Extensions to the Navigator interface

► MDN 

WebIDL

```
partial interface Navigator {  
  [SameObject] readonly attribute Geolocation geolocation;  
};
```

§ 6. Geolocation interface and callbacks

► MDN 

WebIDL

```
[Exposed=Window]  
interface Geolocation {  
  undefined getCurrentPosition (  
    PositionCallback successCallback,  
    optional PositionErrorCallback? errorCallback = null,  
    optional PositionOptions options = {}  
  );  
};
```

```

long watchPosition (
    PositionCallback successCallback,
    optional PositionErrorCallback? errorCallback = null,
    optional PositionOptions options = {}
);

undefined clearWatch (long watchId);
};

callback PositionCallback = undefined (
    GeolocationPosition position
);

callback PositionErrorCallback = undefined (
    GeolocationPositionError positionError
);

```

§ 6.1 Internal slots

Instances of [Geolocation](#) are created with the internal slots in the following table:

Internal slot	Description
[[cachedPosition]]	A GeolocationPosition , initialized to null. It's a reference to the last acquired position and serves as a cache. A user agent <i>MAY</i> evict [[cachedPosition]] by resetting it to null at any time for any reason.
[[watchIDs]]	Initialized as an empty list of unsigned Long items.

§ 6.2 [getCurrentPosition\(\)](#) method

► MDN 

The ***getCurrentPosition(successCallback, errorCallback, options)*** method steps are:

1. If the current settings object's relevant global object's associated Document is not fully active:
 1. [Call back with error](#) *errorCallback* and [POSITION_UNAVAILABLE](#).
 2. Terminate this algorithm.
2. In parallel, [request a position](#) passing *successCallback*, *errorCallback*, and *options*.

The **`watchPosition`**(*successCallback*, *errorCallback*, *options*) method steps are:

1. If the current settings object's relevant global object's associated Document is not fully active:
 1. Call back with error passing *errorCallback* and `POSITION_UNAVAILABLE`.
 2. Return 0.
2. Let *watchId* be an implementation-defined unsigned Long that is greater than zero.
3. Append *watchId* to this's `[[watchIDs]]`.
4. In parallel, request a position passing *successCallback*, *errorCallback*, *options*, and *watchId*.
5. Return *watchId*.


§ 6.4 `clearWatch()` method

When **`clearWatch()`** is invoked, the user agent *MUST*:

1. Remove *watchId* from this's `[[watchIDs]]`.

§ 6.5 Request a position

To ***request a position***, pass a PositionCallback *successCallback*, a PositionErrorCallback? *errorCallback*, PositionOptions *options*, and an optional *watchId*:

1. Let *watchIDs* be this's `[[watchIDs]]`.
2. Let *document* be the current settings object's relevant global object's associated Document.
3. If *document* is not allowed to use the "geolocation" feature:
 1. If *watchId* was passed, remove *watchId* from *watchIDs*.
 2. Call back with error passing *errorCallback* and `PERMISSION_DENIED`.
 3. Terminate this algorithm.
4. If *document*'s visibility state is "hidden", wait for the following page visibility change steps to run:
 1.  Assert: *document*'s visibility state is "visible".
 2. Continue to the next steps below.
5. Let *descriptor* be a new `PermissionDescriptor` whose name is "geolocation".
6. Set *permission* to request permission to use *descriptor*.

7. If *permission* is "denied", then:

1. If *watchId* was passed, remove *watchId* from *watchIDs*.
2. [Call back with error](#) passing *errorCallback* and [PERMISSION_DENIED](#).
3. Terminate this algorithm.

8. Wait to [acquire a position](#) passing *successCallback*, *errorCallback*, *options*, and *watchId*.

9. If *watchId* was not passed, terminate this algorithm.

10. While *watchIDs* contains *watchId*:

1. Wait for a significant change of geographic position. What constitutes a significant change of geographic position is left to the implementation. User agents *MAY* impose a rate limit on how frequently position changes are reported.
2. If *document* is not fully active or visibility state is not "visible", go back to the previous step and again [wait for a significant change of geographic position](#).

NOTE: Position updates are exclusively for fully-active visible documents

The desired effect here being that position updates are exclusively delivered to fully active documents that are visible; Otherwise the updates get silently "dropped on the floor". Only once a document again becomes fully active and visible (e.g., an *iframe* gets reattached to a parent document), do the position updates once again start getting delivered.

3. Wait to [acquire a position](#) passing *successCallback*, *errorCallback*, *options*, and *watchId*.

§ 6.6 Acquire a position

To **acquire a position**, passing [PositionCallback](#) *successCallback*, a [PositionErrorCallback?](#) *errorCallback*, [PositionOptions](#) *options*, and an optional *watchId*.

1. If *watchId* was passed and this's [\[\[watchIDs\]\]](#) does not contain *watchId*, terminate this algorithm.
2. Let *acquisitionTime* be a new *EpochTimeStamp* that represents now.
3. Let *timeoutTime* be the sum of *acquisitionTime* and *options.timeout*.
4. Let *cachedPosition* be this's [\[\[cachedPosition\]\]](#).
5. Create an implementation-specific *timeout* task that elapses at *timeoutTime*, during which it tries to acquire the device's position by running the following steps:
 1. Let *permission* be get the current permission state of ["geolocation"](#).
 2. If *permission* is "denied":
 1. Stop *timeout*.
 2. Do the [user or system denied permission](#) failure case step.

3. If *permission* is "granted":

1. Let *position* be null.
2. If *cachedPosition* is not null, and *options*.[maximumAge](#) is greater than 0:
 1. Let *cacheTime* be *acquisitionTime* minus the value of the *options*.[maximumAge](#) member.
 2. If *cachedPosition*'s [timestamp](#)'s value is greater than *cacheTime*, and *cachedPosition*.
[\[\[isHighAccuracy\]\]](#) equals *options*.[enableHighAccuracy](#), set *position* to *cachedPosition*.
3. Otherwise, if *position* is not *cachedPosition*, try to acquire position data from the underlying system, optionally taking into consideration the value of *options*.[enableHighAccuracy](#) during acquisition.
4. If the *timeout* elapses during acquisition, or acquiring the device's position results in failure:
 1. Stop the *timeout*.
 2. Go to [dealing with failures](#).
 3. Terminate this algorithm.
5. If acquiring the position data from the system succeeds:
 1. Set *position* be [a new GeolocationPosition](#) passing *acquisitionTime* and *options*.[enableHighAccuracy](#).
 2. Set this's [\[\[cachedPosition\]\]](#) to *position*.
6. Stop the *timeout*.
7. Queue a task on the [geolocation task source](#) with a step that invokes *successCallback* with *position*.

↪ **Dealing with failures:**

- If acquiring a position fails, do one of the following based on the condition that matches the failure:

↪ **User or system denied permission:**

[Call back with error](#) passing *errorCallback* and [PERMISSION_DENIED](#).

NOTE: Browser permission VS OS permission

On certain platforms, there can be a circumstance where the user has granted the user agent permission to use Geolocation at the browser-level, but the permission to access location services has been denied at the OS level.

↪ **Timeout elapsed:**

[Call back with error](#) with *errorCallback* and [TIMEOUT](#).

↪ **Data acquisition error or any other reason:**

[Call back with error](#) passing *errorCallback* and [POSITION_UNAVAILABLE](#).

§ 6.7 Call back with error

When instructed to *call back with error*, given an [PositionErrorCallback?](#) *callback* and an [unsigned short](#) *code*:

1. If *callback* is null, return.
2. Let *error* be a newly created [GeolocationPositionError](#) instance whose [code](#) attribute is initialized to *code*.
3. Queue a task on the [geolocation task source](#) with a step that invokes *callback* with *error*.

§ 7. *PositionOptions* dictionary

WebIDL

```
dictionary PositionOptions {  
    boolean enableHighAccuracy = false;  
    [Clamp] unsigned long timeout = 0xFFFFFFFF;  
    [Clamp] unsigned long maximumAge = 0;  
};
```

§ 7.1 *enableHighAccuracy* member

The *enableHighAccuracy* member provides a hint that the application would like to receive the most accurate location data. The intended purpose of this member is to allow applications to inform the implementation that they do not require high accuracy geolocation fixes and, therefore, the implementation *MAY* avoid using geolocation providers that consume a significant amount of power (e.g., GPS).

NOTE: A word of warning about *enableHighAccuracy*

The *enableHighAccuracy* member can result in slower response times or increased power consumption. The user might also disable this capability, or the device might not be able to provide more accurate results than if the flag wasn't specified.

§ 7.2 *timeout* member

The ***timeout*** member denotes the maximum length of time, expressed in milliseconds, before [acquiring a position](#) expires.

NOTE: When is the timeout calculated?

The time spent waiting for the document to become visible and for [obtaining permission to use the API](#) is not included in the period covered by the [timeout](#) member. The [timeout](#) member only applies when [acquiring a position](#) begins.

NOTE: Immediate cancellation

An *options*.[timeout](#) value 0 can cause immediate failures.

§ 7.3 maximumAge member

The ***maximumAge*** member indicates that the web application is willing to accept a cached position whose age is no greater than the specified time in milliseconds.

§ 8. GeolocationPosition interface

► MDN 

WebIDL

```
[Exposed=Window, SecureContext]
interface GeolocationPosition {
    readonly attribute GeolocationCoordinates coords;
    readonly attribute EpochTimeStamp timestamp;
};
```

§ 8.1 coords attribute

► MDN 

The ***coords*** attribute contains geographic coordinates.

§ 8.2 timestamp attribute

► MDN 

The *timestamp* attribute represents the time when the geographic position of the device was acquired.

§ 8.3 Internal slots

Instances of [GeolocationPositionError](#) are created with the internal slots in the following table:

Internal slot	Description
<i>[[isHighAccuracy]]</i>	A boolean that records the value of the enableHighAccuracy member when this GeolocationPosition is created .

§ 8.4 Task sources

The following task source is defined by this specifications.

The *geolocation task source*

Used by this specification to queue up non-blocking [PositionCallback](#) and [PositionErrorCallback](#) when performing [position requests](#).

§ 9. *GeolocationCoordinates* interface

► MDN

WebIDL

```
[Exposed=Window, SecureContext]
interface GeolocationCoordinates {
  readonly attribute double accuracy;
  readonly attribute double latitude;
  readonly attribute double longitude;
  readonly attribute double? altitude;
  readonly attribute double? altitudeAccuracy;
  readonly attribute double? heading;
  readonly attribute double? speed;
};
```

§ 9.1 latitude, longitude, and accuracy attributes

► MDN

► MDN

The ***latitude*** and ***longitude*** attributes are geographic coordinates specified in decimal degrees.

► MDN 

The ***accuracy*** attribute denotes the accuracy level of the latitude and longitude coordinates in meters (e.g., 65 meters).

§ 9.2 altitude and altitudeAccuracy attributes

► MDN 

► MDN 

The ***altitude*** attribute denotes the height of the position, specified in meters above the [WGS84] ellipsoid.

The ***altitudeAccuracy*** attribute represents the altitude accuracy in meters (e.g., 10 meters).

§ 9.3 heading attribute

► MDN 

The ***heading*** attribute denotes the direction of travel of the hosting device and is specified in degrees, where $0^\circ \leq \text{heading} < 360^\circ$, counting clockwise relative to the true north.

§ 9.4 speed attribute

► MDN 

The ***speed*** attribute denotes the magnitude of the horizontal component of the hosting device's current velocity in meters per second.

§ 9.5 Constructing a GeolocationPosition

A new ***GeolocationPosition*** is constructed with EpochTimeStamp *timestamp* and boolean *isHighAccuracy* by performing the following steps:

1. Let *coords* be a newly created [GeolocationCoordinates](#) instance:
 1. Initialize *coord*'s [latitude](#) attribute to a geographic coordinate in decimal degrees.
 2. Initialize *coord*'s [longitude](#) attribute to a geographic coordinate in decimal degrees.
 3. Initialize *coord*'s [accuracy](#) attribute to a non-negative real number. The value *SHOULD* correspond to a 95% confidence level with respect to the longitude and latitude values.
 4. Initialize *coord*'s [altitude](#) attribute in meters above the [WGS84] ellipsoid, or null if the implementation cannot provide altitude information.
 5. Initialize *coord*'s [altitudeAccuracy](#) attribute as non-negative real number, or to null if the implementation cannot provide altitude information. If the altitude accuracy information is provided, it

SHOULD correspond to a 95% confidence level.

6. Initialize *coord*'s [speed](#) attribute to a non-negative real number, or as null if the implementation cannot provide speed information.
 7. Initialize *coord*'s [heading](#) attribute in degrees, or null if the implementation cannot provide heading information. If the hosting device is stationary (i.e., the value of the [speed](#) attribute is 0), then initialize the [heading](#) to NaN.
2. Return a newly created [GeolocationPosition](#) instance with its [coords](#) attribute initialized to *coords* and [timestamp](#) attribute initialized to *timestamp*, and its [\[\[isHighAccuracy\]\]](#) internal slot set to *isHighAccuracy*.

§ 10. [GeolocationPositionError](#) interface

► MDN

WebIDL

```
[Exposed=Window]
interface GeolocationPositionError {
  const unsigned short PERMISSION\_DENIED = 1;
  const unsigned short POSITION\_UNAVAILABLE = 2;
  const unsigned short TIMEOUT = 3;
  readonly attribute unsigned short code;
  readonly attribute DOMString message;
};
```

§ 10.1 Constants

PERMISSION_DENIED (numeric value 1)

[Request a position](#) failed because the user denied permission to use the API.

POSITION_UNAVAILABLE (numeric value 2)

[Acquire a position](#) failed.

TIMEOUT (numeric value 3)

The length of time specified by the [timeout](#) member has elapsed before the user agent could successfully [acquire a position](#).

§ 10.2 [code](#) attribute

► MDN 

The ***code*** attribute returns the value it was [initialized to](#) (see [10.1 Constants](#) for possible values).

The *message* attribute is a developer-friendly textual description of the [code](#) attribute.

NOTE: Don't show `.message` to users!

The purpose of [GeolocationPositionError](#)'s [message](#) attribute is to assist developers with debugging. For legacy reasons, this attribute does not supply language or base direction metadata and is only localized into English. Developers are advised to use [GeolocationPositionError](#)'s [code](#) attribute to create a localized experience.

§ 11. Permissions policy

The *Geolocation API* defines a policy-controlled feature identified by the token string "[geolocation](#)". Its default allowlist is `'self'`.

§ 12. Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words *MAY*, *MUST*, *RECOMMENDED*, and *SHOULD* in this document are to be interpreted as described in [BCP 14](#) [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

§ A. IDL Index

WebIDL

```
partial interface Navigator {  
  [SameObject] readonly attribute Geolocation geolocation;  
};
```

```
[Exposed=Window]  
interface Geolocation {  
  undefined getCurrentPosition (  
    PositionCallback successCallback,
```

```

    optional PositionErrorCallback? errorCallback = null,
    optional PositionOptions options = {}
);

long watchPosition (
    PositionCallback successCallback,
    optional PositionErrorCallback? errorCallback = null,
    optional PositionOptions options = {}
);

undefined clearWatch (long watchId);
};

callback PositionCallback = undefined (
    GeolocationPosition position
);

callback PositionErrorCallback = undefined (
    GeolocationPositionError positionError
);

dictionary PositionOptions {
    boolean enableHighAccuracy = false;
    [Clamp] unsigned long timeout = 0xFFFFFFFF;
    [Clamp] unsigned long maximumAge = 0;
};

[Exposed=Window, SecureContext]
interface GeolocationPosition {
    readonly attribute GeolocationCoordinates coords;
    readonly attribute EpochTimeStamp timestamp;
};

[Exposed=Window, SecureContext]
interface GeolocationCoordinates {
    readonly attribute double accuracy;
    readonly attribute double latitude;
    readonly attribute double longitude;
    readonly attribute double? altitude;
    readonly attribute double? altitudeAccuracy;
    readonly attribute double? heading;
    readonly attribute double? speed;
};

[Exposed=Window]
interface GeolocationPositionError {
    const unsigned short PERMISSION\_DENIED = 1;
    const unsigned short POSITION\_UNAVAILABLE = 2;
};

```



```
const unsigned short TIMEOUT = 3;
readonly attribute unsigned short code;
readonly attribute DOMString message;
};
```

§ B. Index

§ B.1 Terms defined by this specification

[A new GeolocationPosition](#) §9.5

[accuracy](#) attribute for GeolocationCoordinates §9.1

[acquire a position](#) §6.6

[altitude](#) attribute for GeolocationCoordinates §9.2

[altitudeAccuracy](#) attribute for
GeolocationCoordinates §9.2

[\[\[cachedPosition\]\]](#) internal slot for Geolocation §6.1

[call back with error](#) §6.7

[checking permission](#) §3.4

[clearWatch\(\)](#) method for Geolocation §6.4

[code](#) attribute for GeolocationPositionError §10.2

[coords](#) attribute for GeolocationPosition §8.1

[enableHighAccuracy](#) member for PositionOptions §7.1

[geolocation](#) attribute for Navigator §5.

[Geolocation](#) interface §6.

[geolocation task source](#) §8.4

["geolocation"](#) §3.4

[GeolocationCoordinates](#) interface §9.

[GeolocationPosition](#) interface §8.

[GeolocationPositionError](#) interface §10.

[getCurrentPosition](#) method for Geolocation §6.2

[heading](#) attribute for GeolocationCoordinates §9.3

[\[\[isHighAccuracy\]\]](#) internal slot for
GeolocationPosition §8.3

[latitude](#) attribute for GeolocationCoordinates §9.1

[longitude](#) attribute for GeolocationCoordinates §9.1

[maximumAge](#) member for PositionOptions §7.3

[message](#) attribute for GeolocationPositionError §10.3

[PERMISSION_DENIED](#) §10.1

[POSITION_UNAVAILABLE](#) §10.1

[PositionCallback](#) §6.

[PositionErrorCallback](#) §6.

[PositionOptions](#) dictionary §7.

[request a position](#) §6.5

[speed](#) attribute for GeolocationCoordinates §9.4

[timeout](#) member for PositionOptions §7.2

[TIMEOUT](#) §10.1

[timestamp](#) attribute for GeolocationPosition §8.2

[\[\[watchIDs\]\]](#) internal slot for Geolocation §6.1

[watchPosition](#) method for Geolocation §6.3

§ B.2 Terms defined by reference

[PERMISSIONS] defines the following:

Permissions

[PERMISSIONS-POLICY] defines the following:

Permissions Policy

[WEBIDL] defines the following:

unsigned long type

§ C. Acknowledgments

This section is non-normative.

This specification builds upon earlier work in the industry, including research by Aza Raskin, Google Gears Geolocation API, and LocationAware.org.

Thanks also to Alec Berntson, Alissa Cooper, Steve Block, Greg Bolsinga, Lars Erik Bolstad, Aaron Boodman, Dave Burke, Chris Butler, Max Froumentin, Shyam Habarakada, Marcin Hanclik, Ian Hickson, Brad Lassey, Angel Machin, Cameron McCormack, Daniel Park, Stuart Parmenter, Olli Pettay, Chris Prince, Arun Ranganathan, Carl Reed, Thomas Roessler, Dirk Segers, Allan Thomson, Martin Thomson, Doug Turner, Erik Wilde, Matt Womer, and Mohamed Zergaoui.

§ D. References

§ D.1 Normative references

[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: <https://www.rfc-editor.org/rfc/rfc2119>

[RFC8174]

Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: <https://www.rfc-editor.org/rfc/rfc8174>

[webidl]

Web IDL Standard. Edgar Chen; Timothy Gu. WHATWG. Living Standard. URL: <https://webidl.spec.whatwg.org/>

[WGS84]

National Imagery and Mapping Agency Technical Report 8350.2, Third Edition. National Imagery and Mapping Agency. 3 January 2000.

§ D.2 Informative references

[Permissions]

[Permissions](https://www.w3.org/TR/permissions/). Marcos Caceres; Mike Taylor. W3C. 30 July 2022. W3C Working Draft. URL:
<https://www.w3.org/TR/permissions/>

[Permissions-Policy]

[Permissions Policy](https://www.w3.org/TR/permissions-policy-1/). Ian Clelland. W3C. 16 July 2020. W3C Working Draft. URL:
<https://www.w3.org/TR/permissions-policy-1/>

