Webmention

W3C Recommendation 12 January 2017

This version:
https://www.w3.org/TR/2017/REC-webmention-20170112/

Latest published version:
https://www.w3.org/TR/webmention/

Latest editor's draft:
https://webmention.net/draft/

Test suite:
https://webmention.rocks/

Implementation report:
https://webmention.net/implementation-reports/

Previous version:
https://www.w3.org/TR/2016/PR-webmention-20161101/

Editor:
Aaron Parecki

Repository:
Github
Issues
Commits

Please check the **errata** for any errors or issues reported since publication.

The English version of this specification is the only normative version. Non-normative translations may also be available.

Abstract

Webmention is a simple way to notify any URL when you mention it on your site. From the receiver's perspective, it's a way to request notifications when other sites mention it.

Author's Note

This section is non-normative.

This specification was contributed to the W3C from the IndieWeb community. More history and evolution of Webmention can be found on the IndieWeb wiki.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. 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/.

This document was published by the Social Web Working Group as a Recommendation. If you wish to make comments regarding this document, please send them to public-socialweb@w3.org (subscribe, archives). All comments are welcome.

Please see the Working Group's implementation report.

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

This document was produced by a group operating under the 5 February 2004 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.

W3C expects the functionality specified in this Recommendation will not be affected by changes to Fetch.

This document is governed by the 1 September 2015 W3C Process Document.

Table of Contents

1. Introduction

A Webmention is a notification that one URL links to another. For example, Alice writes an interesting post on her blog. Bob then writes a response to her post on his own site, linking back to Alice's original post. Bob's publishing software sends a Webmention to Alice notifying that her article was replied to, and Alice's software can show that reply as a comment on the original post.

Sending a Webmention is not limited to blog posts, and can be used for additional kinds of content and responses as well. For example, a response can be an RSVP to an event, an indication that someone "likes" another post, a "bookmark" of another post, and many others. Webmention enables these interactions to happen across different websites, enabling a distributed social web.

1.1 Social Web Working Group

This section is non-normative.

Webmention is one of several related specifications being produced by the Social Web Working Group. Implementers interested in alternative approaches and complimentary protocols should start by reading the overview document [social-web-protocols].

1.2 Background

This section is non-normative.

The Webmention spec began as a simplified version of the [Pingback] spec. Where Pingback required sending the source and target URLs in an XML-RPC payload, Webmention simplified that to a form-encoded payload, which meant it could easily be used in HTML forms, was easier to work with since more tools exist for form-encoded payloads, and was not vulnerable to accidentally exposing other parts of a system's code via XML-RPC.

Webmention then continued on to more thoroughly specify the details of sending and verifying the request, and expanded to support sending notifications when a source document was updated or deleted. More information can be found in the Webmention FAQ on the IndieWeb wiki.

1.3 Overview

This section is non-normative.

A typical Webmention flow is as follows:

1. Alice posts some interesting content on her site (which is set up to receive Webmentions).
2. Bob sees this content and comments about it on his site, linking back to Alice's original post.
3. Using Webmention, Bob's publishing software automatically notifies Alice's server that her post has been linked to by the URL of Bob's post.
4. Alice's publishing software verifies that Bob's post actually contains a mention of her post and then includes this information on her site.

1.4 Protocol Summary

This section is non-normative.

Webmentions are sent "from" a source URL "to" a target URL to notify the target that it has been mentioned at the source URL.

1. User Aaron writes a post on his blog.
2. User Barnaby writes a post on his blog that links to Aaron's post.
3. After publishing the post (i.e., after it has a URL), Barnaby's server notices the mention of Aaron's post as part of the publishing process.

4. Barnaby's server does Webmention discovery on Aaron's post to find its Webmention endpoint (if not found, process stops).
5. Barnaby's server sends a Webmention notification to Aaron's post's Webmention endpoint with:
    - `source` set to Barnaby's post's permalink
    - `target` set to Aaron's post's permalink.
6. Aaron's server receives the Webmention.
7. Aaron's server verifies that `target` in the Webmention is a valid permalink on Aaron's blog (if not, processing stops).
8. Aaron's server verifies that the `source` in the Webmention (when retrieved, after following redirects [FETCH]) contains a hyperlink to the `target` (if not, processing stops).

2. Conformance

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", " SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.1 Conformance Classes

Webmention implementations are either senders or receivers. This section describes the conformance criteria for both.

Listed below are known types of Webmention implementations.

Senders

A Webmention Sender is an implementation that sends Webmentions. In order for a Sender to send a Webmention, there must first be a document at a URL that is accessible to the Receiver.

The conformance criteria for Webmention senders is described in Sending Webmentions

Listed below are some known types of Webmention Senders.

- Blogging or microblogging software
- Plugins for existing CMS software

- Browser plugins to send a Webmention from a context menu on behalf of the page you're viewing
- Web APIs to offload the Webmention endpoint discovery and Webmention delivery
- Proxy services that converts existing social media posts to HTML posts and sends Webmentions

Receivers

A Webmention Receiver is an implementation that receives Webmentions to one or more target URLs on which the Receiver's Webmention endpoint is advertised. In order to receive a Webmention, there must be a URL that advertises the Receiver's Webmention endpoint. The URL is not considered part of the Receiver's implementation, as it may exist in an entirely different system or domain.

The conformance criteria for Webmention receivers is described in Receiving Webmentions

Listed below are some known types of Webmention Receivers.

- Blogging or microblogging software
- Plugins for existing CMS software
- Web services and APIs to receive Webmentions and display them as comments via Javascript embed

2.2 Test Suite and Reporting

Please submit your implementation reports at http://webmention.net/implementation-reports/. Instructions are provided at the URL. The implementation report template references the tests available at webmention.rocks.

webmention.rocks provides many test cases you can use to live-test your implementation. It also is a useful tool to use while developing a Webmention implementation, as it provides detailed responses when errors are encountered.

3. Webmention Protocol

This specification uses the link rel registry as defined by [HTML5] for both HTML and HTTP link relations.

3.1 Sending Webmentions

3.1.1 Create a source document that mentions the target

Webmentions are sent "from" a source URL "to" a target URL to notify the target that it has been mentioned at the source URL. Before a Webmention can be sent, there needs to be a source URL to send the Webmention "from", often a blog post but may be any type of content.

For example, the URL at https://waterpigs.example/post-by-barnaby may contain the following HTML that has a link to Aaron's post.

EXAMPLE 1

```
<!doctype html>
<html>
  <body>
    <a href="https://aaronpk.example/post-by-aaron">This is a great post</a>
  </body>
</html>
```

3.1.2 Sender discovers receiver Webmention endpoint

The sender MUST fetch the target URL (and follow redirects [FETCH]) and check for an HTTP Link header [RFC5988] with a rel value of `webmention`. If the content type of the document is HTML, then the sender MUST look for an HTML `<link>` and `<a>` element with a rel value of `webmention`. If more than one of these is present, the first HTTP Link header takes precedence, followed by the first `<link>` or `<a>` element in document order. Senders MUST support all three options and fall back in this order.

The endpoint MAY be a relative URL, in which case the sender MUST resolve it relative to the `target` URL according to [URL].

The endpoint MAY contain query string parameters, which MUST be preserved as query string parameters and MUST NOT be sent as POST body parameters when sending the Webmention request.

Senders MAY initially make an HTTP HEAD request [RFC7231] to check for the Link header before making a GET request.

EXAMPLE 2

```
GET /post-by-aaron HTTP/1.1
Host: aaronpk.example
HTTP/1.1 200 OK
Link: <http://aaronpk.example/webmention-endpoint>; rel="webmention"

<html>
<head>
...
<link href="http://aaronpk.example/webmention-endpoint" rel="webmention" />
...
</head>
<body>
....
<a href="http://aaronpk.example/webmention-endpoint" rel="webmention">webmentio
...
</body>
</html>
```

Senders MAY customize the HTTP User Agent [RFC7231] used when fetching the target URL in order to indicate to the recipient that this request is made as part of Webmention discovery. In this case, it is recommended to include the string "Webmention" in the User Agent. This provides people with a pointer to find out why the discovery request was made.

3.1.3 Sender notifies receiver

The sender MUST post x-www-form-urlencoded [HTML5] `source` and `target` parameters to the Webmention endpoint, where `source` is the URL of the sender's page containing a link, and `target` is the URL of the page being linked to.

Note that if the Webmention endpoint URL contains query string parameters, the query string parameters MUST be preserved, and MUST NOT be sent in the POST body.

The Webmention endpoint will validate and process the request, and return an HTTP status code [RFC7231]. Most often, `202 Accepted` or `201 Created` will be returned, indicating that the request is queued and being processed asynchronously to prevent DoS (Denial of Service) attacks. If the response code is 201, the `Location` header will include a URL that can be used to monitor the status of the request.

Any 2xx response code MUST be considered a success.

EXAMPLE 3

```
POST /webmention-endpoint HTTP/1.1
Host: aaronpk.example
Content-Type: application/x-www-form-urlencoded

source=https://waterpigs.example/post-by-barnaby&
target=https://aaronpk.example/post-by-aaron


HTTP/1.1 202 Accepted
```

3.1.4 Sending Webmentions for updated posts

If the source URL was updated, the sender SHOULD re-send any previously sent Webmentions, (including re-sending a Webmention to a URL that may have been removed from the document), and SHOULD send Webmentions for any new links that appear at the URL.

This allows the recipients of Webmentions to update their display of the source document, or otherwise notify the recipient that a post that mentioned one of their URLs was updated.

When sending a Webmention when a post is updated, the sender MUST re-discover the Webmention endpoint of each target URL, in case the target has updated their Webmention endpoint.

If a response to the source URL is shown on the source URL page (e.g. as a comment), then sender SHOULD treat that as an update of the source URL and re-send any previously sent Webmentions.

3.1.5 Sending Webmentions for deleted posts

If the source URL was deleted, the sender SHOULD return an HTTP `410 Gone` status code for the URL, and SHOULD display a "tombstone" representation of the deleted post, typically by blanking out the values of any properties in the post, and/or replacing the primary content of the post (e.g. the name and/or content of [h-entry]) with "Deleted". The sender SHOULD then re-send Webmentions for every previously sent Webmention for that document.

This allows receivers which may have displayed a previously received Webmention as a comment or other interaction to remove it from view if they support deletes, while providing a reasonable fallback

for receivers which only support updates.

3.2 Receiving Webmentions

Upon receipt of a POST request containing the `source` and `target` parameters, the receiver SHOULD verify the parameters (see Request Verification below) and then SHOULD queue and process the request asynchronously, to prevent DoS attacks. There are three possible responses to the request, depending on how the receiver processes it.

If the receiver creates a status page which the sender can use to check the status, the receiver MUST reply with an `HTTP 201 Created` response with a `Location` header pointing to the status URL. The response body MAY contain content.

> EXAMPLE 4
>
> ```
> HTTP/1.1 201 Created
> Location: http://aaronpk.example/webmention/DEhB9Jme
> ```

If the receiver processes the request asynchronously but does not return a status URL, the receiver MUST reply with an `HTTP 202 Accepted` response. The response body MAY contain content, in which case a human-readable response is recommended.

> EXAMPLE 5
>
> ```
> HTTP/1.1 202 Accepted
> ```

If the receiver chooses to process the request and perform the verification step synchronously (not recommended), it MUST respond with a `200 OK` status on success.

3.2.1 Request Verification

The receiver MUST check that `source` and `target` are valid URLs [URL] and are of schemes that are supported by the receiver. (Most commonly this means checking that the `source` and `target` schemes are http or https).

The receiver MUST reject the request if the source URL is the same as the target URL.

The receiver SHOULD check that `target` is a valid resource for which it can accept Webmentions. This check SHOULD happen synchronously to reject invalid Webmentions before more in-depth verification begins. What a "valid resource" means is up to the receiver. For example, some receivers may accept Webmentions for multiple domains, others may accept Webmentions for only the same domain the endpoint is on.

Note that a target URL may contain a fragment identifier, and if the receiver limits which URLs can receive Webmentions, the fragment SHOULD be ignored when checking if the URL is supported.

3.2.2 Webmention Verification

Webmention verification SHOULD be handled asynchronously to prevent DoS (Denial of Service) attacks.

If the receiver is going to use the Webmention in some way, (displaying it as a comment on a post, incrementing a "like" counter, notifying the author of a post), then it MUST perform an HTTP GET request on source, following any HTTP redirects (and SHOULD limit the number of redirects it follows) to confirm that it actually mentions the target. The receiver SHOULD include an HTTP Accept header indicating its preference of content types that are acceptable.

The receiver SHOULD use per-media-type rules to determine whether the source document mentions the target URL. For example, in an [HTML5] document, the receiver should look for `<a href="*">`, `<img href="*">`, `<video src="*">` and other similar links. In a JSON ([RFC7159]) document, the receiver should look for properties whose values are an exact match for the URL. If the document is plain text, the receiver should look for the URL by searching for the string. Other content types may be handled at the implementer's discretion. The source document MUST have an exact match of the `target` URL provided in order for it to be considered a valid Webmention.

At this point, the receiver MAY publish content from the source page on the target page or other pages, along with any other data it picks up from the source. For example, the receiver may display the contents of the source as a comment on the post, or may display the author's profile photo in a list of others who have sent similar Webmentions, e.g. showing a list of people who have all "liked" a post.

> NOTE
>
> When republishing content from the source page, care should be taken to ensure you are not unintentionally changing the visibility of the content. For example, if the source document is visible only to a limited audience, you should ensure the republished content is not visible to the general public. The source document may be restricted to a specific audience either by some form of authentication, or because the source URL was behind a firewall that the receiver is also behind.

3.2.3 Error Responses

If the Webmention was not successful because of something the *sender* did, it MUST return a `400 Bad Request` status code and MAY include a description of the error in the response body.

Possible sender-related errors that can be returned synchronously before making a GET request to the source:

- Specified `target` URL not found.
- Specified `target` URL does not accept Webmentions.
- `source` URL was malformed or is not a supported URL scheme (e.g. a mailto: link)

Possible sender-related errors that can occur after fetching the contents of the source URL:

- `source` URL not found.
- `source` URL does not contain a link to the `target` URL.

If the Webmention was not successful because of an error on the receiver's server, it SHOULD return a `500 Internal Server Error` status code and MAY include a description of the error in the response body.

3.2.4 Updating existing Webmentions

If receiver had received a Webmention in the past with the same `source` and `target` then,

- If both the verification steps are successful, it SHOULD update any existing data it picked from `source` for the existing Webmention.
    - When a Webmention implementation does support updating (i.e., a "Webmention update

implementation"), it MUST support updating data from properties of the primary object of the source. (e.g. properties of the [h-entry] of the page).

    - A Webmention update implementation MAY support updating data from children, or other descendant objects of the primary object (e.g. a comment h-entry inside the h-entry of the page). Note: Implementations that support this may wish to consider supporting it according to the [Salmention] extension.

- If it received a `410 Gone` status code on step 2 (performing a GET request on source), or received a `200 OK` status code and does not find a mention of `target` on `source`, it SHOULD delete the existing Webmention, or mark it as deleted.
- Processing Webmentions SHOULD be idempotent. That is, receiving multiple Webmentions for the same `source` and `target` with no content changes should not show as multiple replies.

4. Security Considerations

4.1 Preventing Abuse

- The verification process SHOULD be queued and processed asynchronously to prevent DoS attacks per section 3.2.
- Receivers MUST verify Webmentions per section 3.2.2.
- If a receiver chooses to display data it picks up from source, it MUST ensure that the data is encoded and/or filtered to prevent [XSS] and [CSRF] attacks.
- Receivers MAY moderate Webmentions before publishing them.
- Receivers MAY periodically re-verify Webmentions and update them.

4.2 Limits on GET requests

The Webmention protocol relies on the sender making a GET (or HEAD) request to discover the receiver's endpoint, followed by the receiver making a GET request to the sender's web page to verify the link. This means a sender can cause a receiver to make GET requests to arbitrary URLs, opening up a potential DoS vector.

Receivers MAY make an initial HTTP HEAD request when verifying the link and decide whether to make a full GET request after initially inspecting the content type, length, or other HTTP headers returned.

Receivers SHOULD place limits on the number of HTTP redirects they follow, for example limiting the number to 20, in order to prevent being stuck in a redirect loop if the sender continues to send redirects.

Receivers SHOULD place limits on the amount of data and time they spend fetching unverified source URLs. For example, if a source URL doesn't respond within 5 seconds, it can treat that as a failure. Similarly, the receiver can fetch only the first 1mb of the page, since any reasonable HTML or JSON page will be smaller than that.

4.3 Avoid sending Webmentions to localhost

When the sender discovers the receiver's Webmention endpoint, there are few legitimate reasons for the endpoint to be localhost or any other loopback address. If the sender has any services that listen on localhost that don't require authentication, it's possible for a malicious Webmention receiver to craft a Webmention endpoint that could cause the sender to make an arbitrary POST request to itself.

During the discovery step, if the sender discovers the endpoint is localhost or a loopback IP address (127.0.0.0/8), it SHOULD NOT send the Webmention to that endpoint.

4.4 Cross-Site Request Forgery

This specification does not define any special handling of a Webmention request that may contain additional headers or parameters such as authentication headers or session cookies. However, if a Webmention endpoint does accept requests with additional headers, it SHOULD protect itself against Cross-Site Request Forgery (CSRF) attacks. One way to prevent CSRF attacks is by including a CSRF token in a query string parameter of the Webmention endpoint, so that a Webmention sender finds the token when discovering the endpoint.

For example, the target URL could advertise a Webmention endpoint that includes a CSRF token:

> EXAMPLE 6
>
> ```
> GET /post-by-aaron HTTP/1.1
> Host: aaronpk.example
> HTTP/1.1 200 OK
> Link: <http://aaronpk.example/webmention?csrf=Q0NTVhYjI0NTVkNDA3M>; rel="webmen
> ```

Then, when the Webmention endpoint is processing a request, it can first check the validity of the

CSRF token before any other processing.

4.5 Limit access to protected resources

It is possible for an attacker to advertise a Webmention endpoint that points to an arbitrary URL. As such, if you install software that sends Webmentions on a server that is behind a firewall or otherwise has access to normally protected resources, you should be aware that an attacker can cause the server to send a POST request to an internal server. You SHOULD take precautions to ensure this server cannot access protected resources by either:

- running the sender in an environment that does not have access to such internal resources
- proxying all HTTP requests from this sender through a proxy that blocks access to internal resources

4.6 Security and Privacy Review

These questions provide an overview of security and privacy considerations for this specification as guided by Self-Review Questionnaire: Security and Privacy ([security-privacy-questionnaire]).

Does this specification deal with personally-identifiable information?
The only potentially personally-identifiable information involved in Webmention are the source and target URLs.

Does this specification deal with high-value data?
No, there is no authentication or other credentials involved.

Does this specification introduce new state for an origin that persists across browsing sessions?
No

Does this specification expose persistent, cross-origin state to the web?
The Webmention receiver may create a temporary resource with information about the Webmention request.

Does this specification expose any other data to an origin that it doesn't currently have access to?
No

Does this specification enable new script execution/loading mechanisms?
No

Does this specification allow an origin access to a user's location?
No

Does this specification allow an origin access to sensors on a user's device?

No

Does this specification allow an origin access to aspects of a user's local computing environment?

No

Does this specification allow an origin access to other devices?

No

Does this specification allow an origin some measure of control over a user agent's native UI?

No

Does this specification expose temporary identifiers to the web?

No

Does this specification distinguish between behavior in first-party and third-party contexts?

No

How should this specification work in the context of a user agent's "incognito" mode?

Webmention does not maintain any state so there are no considerations when in "incognito" mode.

Does this specification persist data to a user's local device?

No

Does this specification allow downgrading default security characteristics?

No

5. Other Considerations

5.1 Sending Webmentions from non-HTML content

If your source document is not HTML, (such as a PDF), or is otherwise restricted from fetching the raw source via a plain HTTP GET request, (such as behind a paywall, or requires a click-through license agreement), you will need to set up an HTML "landing page" that lists all the targets you wish to send Webmentions to. After creating this HTML landing page, you can use its URL as the source URL when sending Webmentions. This gives receivers a URL they can fetch to verify the link to their target URL, while avoiding making the complete source document public.

Creating an HTML landing page can help increase the number of inbound links to your content, by providing people with a useful place to link to when referencing otherwise restricted content such as scholarly articles. In the case of a scholarly article, the landing page should include the list of references in the HTML, so that you can use them as Webmention target URLs.

5.2 Sending Webmentions for large datasets

If your Webmention source document is particularly large (such as a large HTML page with thousands of records in a table, or a very large JSON file), you will likely want to avoid sending a Webmention with that source URL. Doing so may cause receivers to download the entire dataset, using a lot of your bandwidth, or receivers may download only the first portion of the file in order to restrict their own bandwidth usage, potentially causing the verification to fail. If the receiver republishes a link to your source document, other visitors may follow that link and inadvertently download the entire dataset as well.

Similar to how it is considered good practice to break up large datasets into pages when being viewed in a browser, when sending Webmentions, you should send Webmentions only from smaller pages of data if your source document is large. If your dataset is HTML, then you can use your (likely) existing HTML pages as source URLs. If your dataset is JSON, you may need to create URLs with smaller JSON documents in pages that you can use as source URLs.

5.3 Respecting cache headers on discovery

When performing Webmention discovery, Senders SHOULD respect the HTTP cache headers [RFC7234] returned by the target URL and avoid fetching the target URL more often than is indicated by the headers.

6. IANA Considerations

The link relation type below has been registered by IANA per Section 6.2.1 of [RFC5988]:

Relation Name:
webmention

Description:
Identifies a target URI that supports the Webmention protocol. This allows clients that mention a resource in some form of publishing process to contact that endpoint and inform it that this resource has been mentioned.

Reference:
W3C Webmention Specification (http://www.w3.org/TR/webmention/)

Notes:
This is a similar "Linkback" mechanism to the ones of Refback, Trackback, and Pingback. It uses a different protocol, though, and thus should be discoverable through its own link relation type.

A. URIs for Form-Encoded Properties

If your implementation wants to treat the `source` and `target` parameters as URIs, you can prefix the terms with `http://www.w3.org/ns/webmention#`.

B. Extensions

This section is non-normative.

The following Webmention Extension Specifications have 2+ interoperable implementations live on the web and are thus listed here:

B.1 Vouch

The [Vouch] protocol is an anti-spam extension to Webmention.

B.2 Salmention

The [Salmention] protocol is an extension to Webmention to propagate comments and other interactions upstream.

B.3 Private Webmention

The [Private-Webmention] protocol is an extension to Webmention that supports sending and verifying Webmentions for posts that have access control.

C. Resources

This section is non-normative.

- Test Suite and Debug Tool
- FAQ
- Implementation Guide
- Implementation Details

- Brainstorming

C.1 Articles

This section is non-normative.

You can find a list of articles about Webmention on the IndieWeb wiki.

C.2 Implementations

This section is non-normative.

You can find a list of Webmention implementations on webmention.net

D. Acknowledgements

The editor wishes to thank Sandeep Shetty for contributing the original draft of the Webmention specification.

Additionally, the editor wishes to thank the IndieWeb community and other implementers for their support, encouragement and enthusiasm, including but not limited to: Amy Guy, Benjamin Roberts, Ben Werdmüller, Dave Wilkinson, Rob Sanderson, and Tantek Çelik.

E. Change Log

This section is non-normative.

E.1 Changes from 01 November PR to REC

- Editorial clarification around sending to loopback addresses
- Updated JSON reference to RFC7159 from the obsolete RFC4627
- Added informative note about breaking up large documents into pages when sending Webmentions from large datasets
- Added note about republishing potentially restricted visibilty content

- Added informative reference to OWASP documents about XSS and CSRF
- Clarified "publish" vs "display" in security considerations
- Added an item to 4.1 security considerations to reiterate verifying Webmentions
- Re-ordered 4.1 security considerations based on chronological order
- Editorial nits

F. References

F.1 Normative references

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

[HTML5]
Ian Hickson; Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Theresa O'Connor; Silvia Pfeiffer. W3C. *HTML5*. 28 October 2014. W3C Recommendation. URL: https://www.w3.org/TR/html5/

[RFC2119]
S. Bradner. IETF. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[RFC5988]
M. Nottingham. IETF. *Web Linking*. October 2010. Proposed Standard. URL: https://tools.ietf.org/html/rfc5988

[RFC7231]
R. Fielding, Ed.; J. Reschke, Ed.. IETF. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7231

[URL]

> **Note:** URLs can be used in numerous different manners, in many differing contexts. For the purpose of producing strict URLs one may wish to consider [RFC3986] and [RFC3987]. The URL specification defines the term URL, various algorithms for dealing with URLs, and an API for constructing, parsing, and resolving URLs. Developers are advised to keep abreast of the latest URL developments by tracking the progress of https://url.spec.whatwg.org/.
>
> As a word of caution, there are notable differences in the manner in which Web browsers and other software stacks outside the HTML context handle URLs. While no changes would be accepted to URL processing that would break existing Web content, some important parts of URL processing should therefore be considered as implementation-defined (e.g. parsing `file:` URLs or operating on URLs that would be syntax errors under the [RFC3986] [RFC3987] syntax).

Anne van Kesteren. WHATWG. *URL Standard*. Continually Updated Specification. URL: https://url.spec.whatwg.org/

F.2 Informative references

[CSRF]
OWASP. *Cross-Site Request Forgery*. Living Document. URL: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[Pingback]
Stuart Langridge; Ian Hickson. hixie.ch. *Pingback 1.0*. Stable Specification. URL: http://www.hixie.ch/specs/pingback/pingback

[Private-Webmention]
Aaron Parecki. IndieWeb. *Private Webmention*. Living Specification. URL: https://indieweb.org/Private-Webmention

[RFC7159]
T. Bray, Ed.. IETF. *The JavaScript Object Notation (JSON) Data Interchange Format*. March 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7159

[RFC7234]
R. Fielding, Ed.; M. Nottingham, Ed.; J. Reschke, Ed.. IETF. *Hypertext Transfer Protocol (HTTP/1.1): Caching*. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7234

[Salmention]
Ben Roberts; Tantek Çelik. IndieWeb. *Salmention*. Living Specification. URL: https://indieweb.org/Salmention

[Vouch]

Aaron Parecki; Tantek Çelik. IndieWeb. *Vouch*. Living Specification. URL: https://indieweb.org/Vouch

[XSS]

OWASP. *Cross-Site Scripting*. Living Document. URL: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[h-entry]

Tantek Çelik. microformats.org. *h-entry*. Living Specification. URL: http://microformats.org/wiki/h-entry

[security-privacy-questionnaire]

Mike West. W3C. *Self-Review Questionnaire: Security and Privacy*. 10 December 2015. W3C Note. URL: https://www.w3.org/TR/security-privacy-questionnaire/

[social-web-protocols]

Amy Guy. W3C. *Social Web Protocols*. 2 November 2016. W3C Working Draft. URL: https://www.w3.org/TR/social-web-protocols/

↥