

URLs & Navigations

rakina@chromium.org, ...

Jun 2021

Background

There are various URLs that are involved in navigation, session history, etc. Also, there are various parameters in `DidCommitProvisionalLoadParams` that are URLs or related to URLs:

- [url](#)
- [base_url](#) (already removed!)
- [referrer](#)
- [redirects](#) (already removed!)
- [url_is_unreachable](#)
- [original_request_url](#) (already removed!)

We intend to [remove](#) all of them from `DidCommitProvisionalLoadParams` and calculate them with values known by the browser at commit time instead. While looking into how to remove these params, we noticed it's pretty unclear how these URLs interact with each other + which URLs are used at which timing, and also found a few bugs (and potential bugs).

This document is here to clarify how these URLs interact with each other, and also point out some potential problems and solutions.

(CMIW! This is mostly from staring at code/specification and also analyzing `DumpWithoutCrashing` reports for `DidCommitParams` verification)

Types of URLs and how we use them

Note: See [Various URL cases](#) to see illustrations of how these URLs are used in practice.

Navigation URLs & “Original” URLs

The destination/target of the navigation. The navigation URL might change due to server redirects.

In Chrome

- `CommonNavigationParams`' [url](#) tracks the latest destination URL while `CommitNavigationParams`' [original_url](#) tracks the URL the navigation started with (before any server redirects)

- WebNavigationParams' url is [set to the original URL](#) at commit time, and used to [set the DocumentLoader's url](#). DocumentLoader's url won't get [updated](#) until it [handles all redirects](#) when it "starts loading".

In the spec

- Navigations are handled by the [navigate](#) algorithm
 - Fragment navigations are [triggered](#) if the navigation URL is the same as the document's URL (excluding fragments)
 - For most cross-document navigations will end up calling [process a navigate fetch](#), which also handles redirects etc and eventually calls [process a navigate response](#) and [creates a document](#) with URL set to the last response's URL

Problems

- WebNavigationParams' URL is used by a lot of things that probably expected the URL to be the final/commit URL instead of the original URL:
 - Determining the [security origin](#) for whether or not unload timing [can be requested](#)
 - Determining [FrameLoadType](#), used to determine whether same-URL navigation should do replacement or not (also, this compares it with [the old document's history/"unreachable" URL](#) instead of navigation URL)
 - Determining [whether the document is still on an empty document](#) or not since the initial empty document.
 - [Filling static response](#) (about:srcdoc, MHTML, etc) though maybe it's not possible to get server redirects before/after these URLs?
- DocumentLoader's url being the original URL before redirect-processing at commit time seems less problematic, current uses (navigation timing etc seems to expect it to be the ["initial URL"](#)), but it's still not obvious (at least to rakina@ :D)

Proposal: Should we just set WebNavigationParam's URL to the commit URL, and add a new "original URL" param explicitly for use cases that do want to use the original URL? (Same for DocumentLoader, which actually [already tracks the original URL](#))

Document URLs

The URL that most decisions should be based on (at least, according to the spec). Normally set to the URL of the last commit, but can be updated due to document.open() or History API's pushState/replaceState (all of these don't go through the "navigate" path in the spec!). Note: there are [various URLs](#) (creation/current/cookie) that are associated with the document, for navigation purposes we probably only care about "current URL"?

In Chrome

- DocumentLoader's url_ keeps track of the Document URL.

- document.open() will [update](#) the URL in the document, DocumentLoader, and let the browser know so that last_url_in_renderer_ in the browser's RenderFrameHostImpl can be updated.
- History API's pushState/replaceState are treated like same-document navigation and create/replace history entries on top of updating the document's URL.
- Special cases:
 - Error page's document URL is [set to chrome-error://chromewebdata/](#)
 - [loadDataURLWithBaseURL\(\)](#) loads a data: URL but [sets the document URL](#) to the supplied "base URL" instead of the data: URL.

In the spec

- Documents have [URLs](#), and most things in the spec base their decisions off the document's URL (with exceptions, see "[Base URL](#)" and "[History URL](#)" sections...)
- document.open(), History API's pushState/replaceState go through the "[URL and history update steps](#)" which updates the document's URL and create/replace history entries with the specified URL.
 - Currently all of these cases also update the session history entry's URL (see "[History URL](#)" section) but there's an [open PR](#) to make document.open() not update the session history entry's URL, de-syncing the document URL with session history entry URL.

Problems

- Document URLs are not tracked 100% correctly in the browser (even after this CL, we're still missing the error page & loadDataURLWithBaseURL cases), so we can't completely replicate decisions made in the renderer.
 - Solution: Track document URLs correctly in the browser! ([CL](#))
- Maybe some parts in Chrome that use "history URL" for decisions should be migrated to use the "document URL" instead?
 - Case: loadDataURLWithBaseURL() commits a data URL "dataURL" and sets the base URL to "baseURL" and the "unreachable"/history URL to "historyURL", then a document.open() happens which updates the document URL to "documentOpenURL" (but not the "unreachable"/history URL). Which URL should be used for navigation decisions?
 - Currently UrlForHistory is used() so "historyURL" is used, which is violating the spec that bases the decision off the current document URL ("documentOpenURL")
 - But if the current document URL is used, if the document.open() never happens, the document URL will be "baseURL" (instead of "historyURL" like what is used now). Is this OK?
 - Additionally, for error pages, the document URL will be "chrome-error://chromewebdata"
 - Solution: Make document.open() URL update the "unreachable"/history URL too (without actually using it in any session history entry). Also, rename

“UrlForHistory” to “UrlForHistoryOrNavigationDecisions” (or maybe we should just split these into two, saving the “url for navigation decisions” explicitly?)

Base URLs

The value used to resolve relative URLs.

In Chrome

- Base URL typically starts out the same as the document URL at the commit time, and [might change](#) after parsing/loading due to <base> elements, etc.
- Special cases:
 - [loadDataURLWithBaseURL\(\)](#) loads a data: URL but sets the base URL to the supplied URL that is **also [used as the document URL](#)** instead of the data: URL. (+ also sets a completely separate history URL)
 - MHTML main resources might [set the base URL](#) at commit time.
 - [WebBundles](#)
 - [Fallback base URLs](#) for about:srcdoc and about:blank
 - (are there other cases?)

In the spec

- Documents have [base URLs](#), mostly used to [resolve relative URLs](#). The base URL might be [updated dynamically](#), and affect only the base URL and no other URLs.

History URLs & “unreachable” URLs

The value that should be saved in the session history entry.

In Chrome

- In non-error page cases, the navigation commit URL is used as the history URL.
- There’s also “unreachable URL” in WebNavigationParams & DocumentLoader, used for two main cases:
 - Error pages sets it to [the URL that failed to load](#)
 - [loadDataURLWithBaseURL\(\)](#) sets it to [a supplied history URL](#) (see [bug](#) on why it uses “unreachable URL”)
- DocumentLoader’s [UrlForHistory\(\)](#) will return the “unreachable URL” if it exists. It is currently used in two cases:
 - Setting the [HistoryItem URL](#). This is exactly what it is meant for.
 - [Determining if a navigation is same-URL](#). This seems problematic as most decisions should be based on the document URL, especially when considering document.open() URL updates. Also this compares the previous document’s history URL **against the new document’s navigation URL**.
 -
- Note that FrameNavigationEntry, which represents the session history entry in the browser, doesn’t actually use the “history URL”. It’s always [created](#) with the filtered [“loading URL”](#) (~navigation/commit URL) - so history navigations to

loadDataURLWithBaseUrl entry will try to navigate to the data: URL again instead of the history URL (which is probably OK?)

In the spec

- Session history entries have URLs that are [set to the document's URL](#) on navigation.

Problems

- History URLs are not tracked in the browser, so we can't completely replicate decisions made in the renderer.
 - Solution: Track history URLs in the browser! ([CL](#))
- [see "Document URL" on whether decisions should be made based on the document URL instead of the history URL]

"Loading"/"Override"/DidCommit URLs

The URL that is used to set DidCommitProvisionalLoadParams' url. This is probably here to preserve the commit/navigation URL to send back the browser.

- loadDataURLWithBaseUrl() commits will [return the data: URL](#) (not the base/document/history URLs)
- Error page commits will return the [commit URL](#) (instead of the document URL that is set to chrome-error://chromewebdata)

According to [this comment](#) this is supposed to be the URL shown in the URL bar, but that's not really true for some cases, including loadDataURLWithBaseUrl() (see "virtual URL" below)

Virtual URLs

URL shown in the URL bar. For most cases this will be the same as the "Loading"/"Override"/DidCommit URL above, except for some cases:

- loadDataURLWithBaseUrl(), where the [supplied history URL](#) will be used if possible.
- [View-source: URLs](#) to include the "view-source" prefix, because the commit URL of view-source URL is just the vanilla URL without prefix.

Blocked/Filtered URLs

The commit URL can change during commit in the renderer before it goes back to the browser due to a few reasons

- Document policy blocking the navigation
 - See [Reusing NavigationRequest with mismatched url](#) (crbug.com/963396), which mentioned that CSP & XFO might block rendering of a document and change the URL to "about:blank#blocked". Currently it looks like only the document policy case is left, as [DocumentLoader::ReplaceWithEmptyDocument\(\)](#) is only called if the document is [blocked by the document policy](#).

- TODO: Move the document policy computation to the browser so that the navigation will just be an error page navigation instead.
- URL filtering
 - We call [FilterURL](#) when we receive URLs from the renderer (when we finished [committing](#), when we [began](#) renderer-initiated navigation, etc).
 - From [850824 - Self-XSS via modal, window.open, and delayed navigation](#) and <https://crrev.com/c/1257863>, it looks like the browser will always pass a valid URL to the renderer on commit now, except for empty URLs - which will be rewritten to about:blank by the renderer (see [CL](#) that DCHECKs this).
 - **Q:** So.. this should only change the URL at commit time for renderer-initiated navigations that don't go through the browser? (renderer-initiated same-document navigation & the synchronous about:blank navigation)

(Nothing in the spec for this probably)

Redirect URLs

Redirects can happen and change the URL of the navigation. There are “server redirects” and “client redirects”. There can only be one client redirect per navigation, but one navigation can go through multiple server redirects. For more details, see

[Redirections in Chrome & its problems](#) (a bit stale because we've fixed some stuff! Only [problem #2 and some parts of problem #1 \(only for referrers\) remain now...](#))

In Chrome

- DocumentLoader is initially created with the original URL (the URL before redirects, instead of the final commit URL). It processes redirects when it “starts loading” (still during commit time) by calling [HandleRedirect\(\)](#) for every redirect, updating its URLs and does additional actions related to redirects (See “[Navigation URL](#)” section for more details/problems)
- Server redirects affect referrer URLs. Also we still track the [redirect chain](#) in the renderer, currently only used to [calculate the referrer](#) for DidCommitProvisionalLoadParams. This can be removed once we move this calculation to the browser. (see “[Referrer URLs](#)” section for more details)

In the spec

- Server redirects are handled as part of the loop in “[process a navigate fetch](#)”. Fetches after redirect calls [http-redirect-fetch](#) instead of the normal [fetch](#).
- The navigation params track whether a navigation [did a cross-origin redirect or not](#) (currently used only for [unload timing](#)?)
- Not sure if there's anything in the spec specifically for client redirects, but some script-initiated navigation default to “replace” ([location object setter](#), [form submission](#)) unless the document is completely loaded or it involves trusted clicks

Referrer URLs

Referrer URLs are sent in the “Referrer” header of navigation requests. It is typically set to the URL of the document where the navigation is initiated but might be stripped/removed completely depending on the policy.

In Chrome

- The referrer is tracked in CommonNavigationParams’ [referrer](#) and also in CommitNavigationParam’s [redirect_infos](#) ([new_referrer](#), [new_referrer_policy](#)).
 - When a server redirect happens, the network stack [re-computes the referrer](#). Then both referrer and redirect_infos are [updated](#).
- When committing the navigation, the renderer [sets](#) WebNavigationParams’ referrer to the CommonNavigationParams’ referrer
 - The referrer will be sanitized in the renderer. The sanitization code in Blink is a bit different than [what the browser used](#) on server redirects (see the first two steps [here](#) which doesn’t exist in the net/ [version](#))... Is this OK?
- DocumentLoader initially uses WebNavigationParams’ referrer but will update it to [use the referrer from RedirectInfos](#) if there are redirects. This final referrer is used for [HistoryItem](#) in the renderer (and the serialized PageState sent to the browser) and also for document.referrer.
- The referrer returned by the renderer in DidCommitProvisionalLoadParams will be:
 - If the navigation was initiated by client redirect, it will be set to the [first entry in the redirect chain](#) (the URL of the document that initiated the client redirect). Note that this is the full unstripped URL.
 - Otherwise it will be the final referrer URL in the DocumentLoader.
- DidCommitProvisionalLoadParams’ referrer is saved in the FrameNavigationEntry. This value will be used to set CommonNavigationParams’ [referrer](#) on history navigation back to that entry.

In the spec

- The referrer for a navigation request is determined through “[Determine request’s referrer](#)”
 - For navigation that just started (no referrer URL yet), it’s handled through the “client” case, which most of the time sets it to the URL of the document that the navigation is happening on? (even when it’s triggered by another frame and also history navigations?)
 - Otherwise, for redirects, the previous referrer URL is used
- Document’s referrer is [set](#) to the final navigation request’s referrer.

Problems

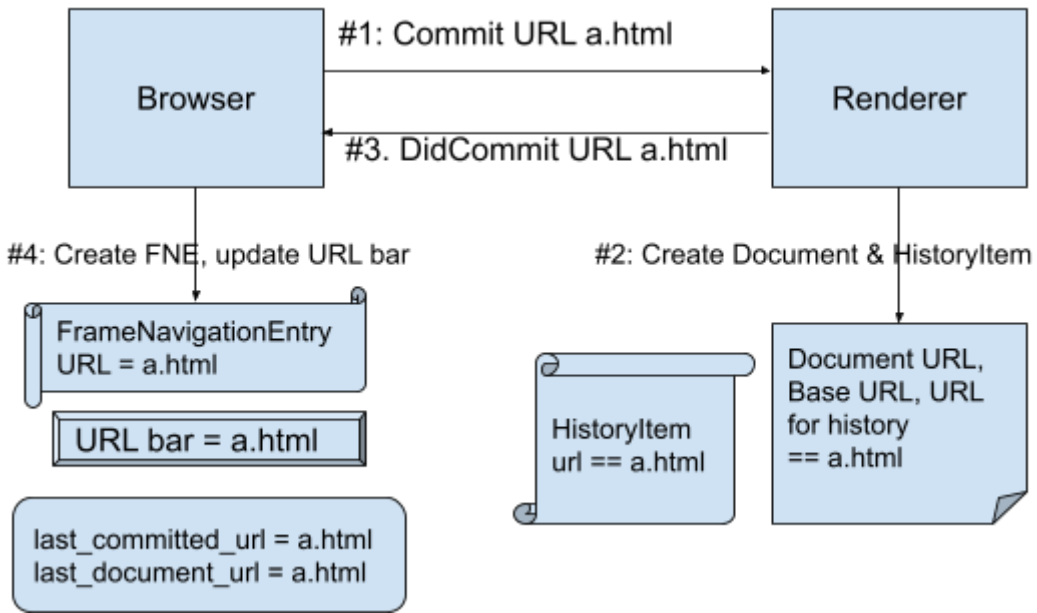
- Referrer URL in DidCommitProvisionalLoadParams might not match the actual final referrer

- If the navigation is a client redirect, it will always [return the full URL](#) of the document that initiated the client redirect, even if the navigation went through cross-origin redirects that strip the actual referrer URL.
- Solution: Maybe just don't do the client redirect special handling? This will make history navigation to these entries start with the potentially-stripped version of the URL, same as the other cases, so this seems OK?

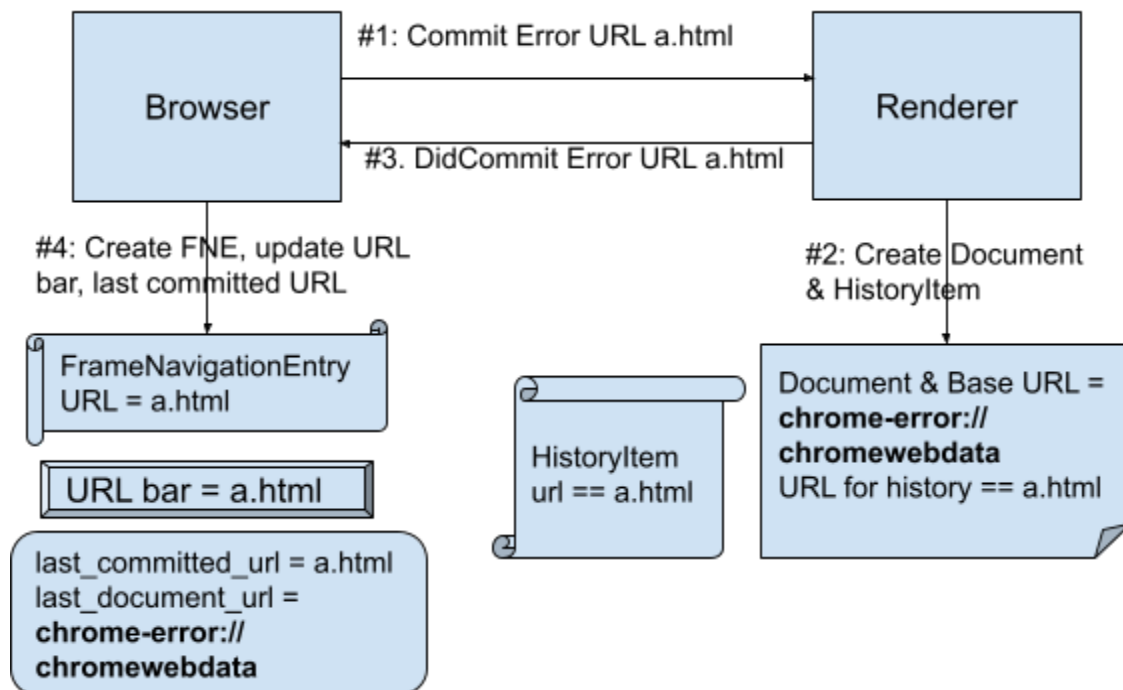
Various URL Cases

Tries to illustrate how various URLs are used in various interesting cases with various URLs

Normal navigation

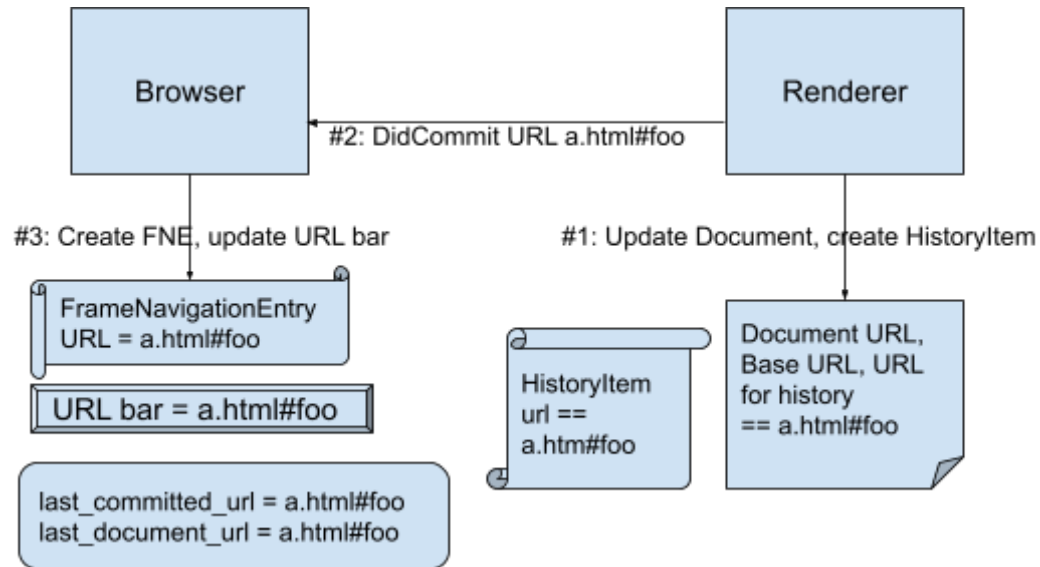


Error page navigation



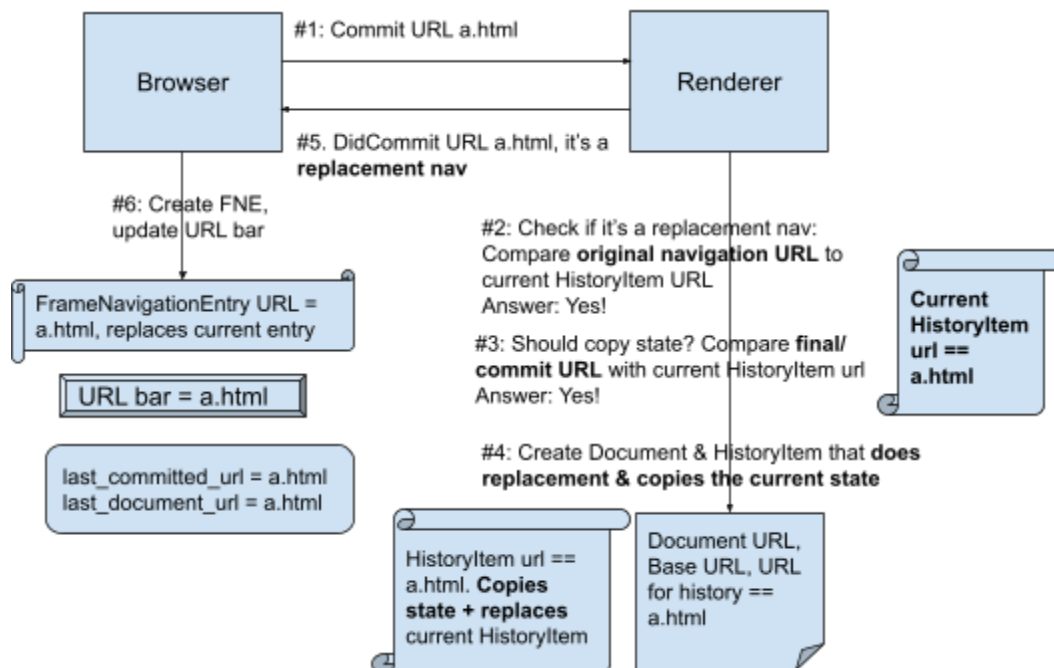
Same document navigation

Renderer-initiated same-document navigations (fragment link clicks, `history.pushState()`, `history.replaceState()`) will update document & history URLs.

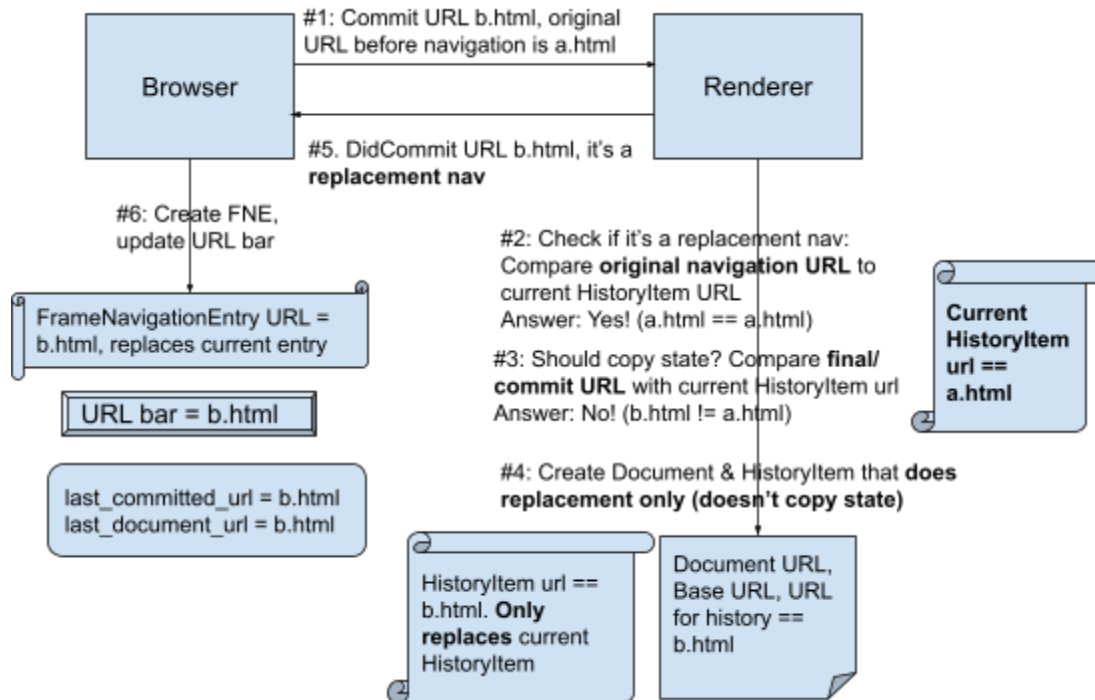


Same URL navigation

Same-URL navigations are treated specially, as they will be [converted to do replacement](#) (if it's not a reload/other non-standard type of navigations) and also [copy various states](#). To figure out if it's a same URL navigation, Chrome currently consults "`UrlForHistory()`" (the URL used for `HistoryItem`) instead of the actual document URL (which is what is used in the spec).

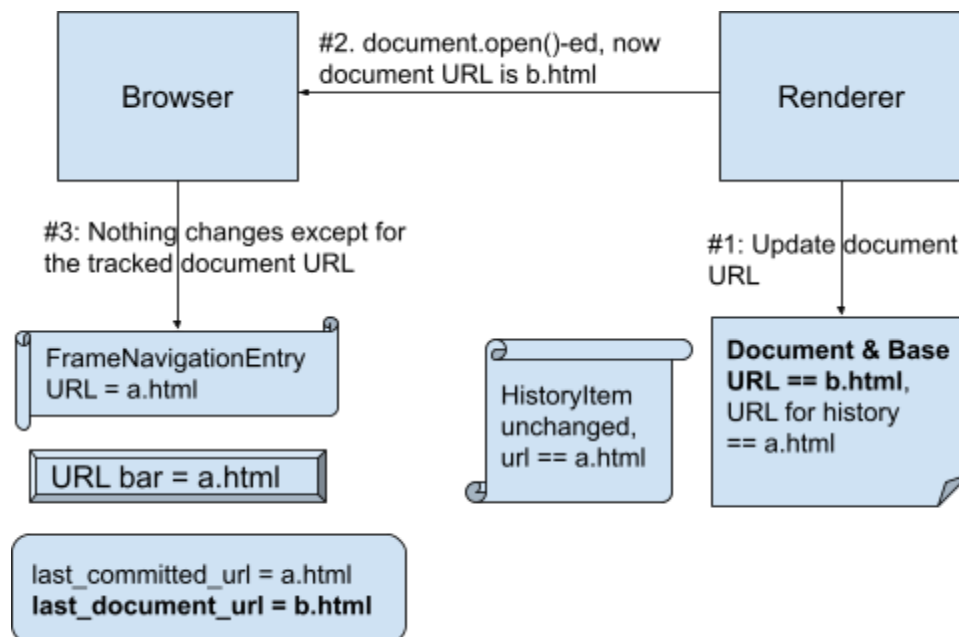


Another interesting quirk is that the URL that is compared for “should the navigation do replacement or not?” is the original URL instead of the final navigation URL.



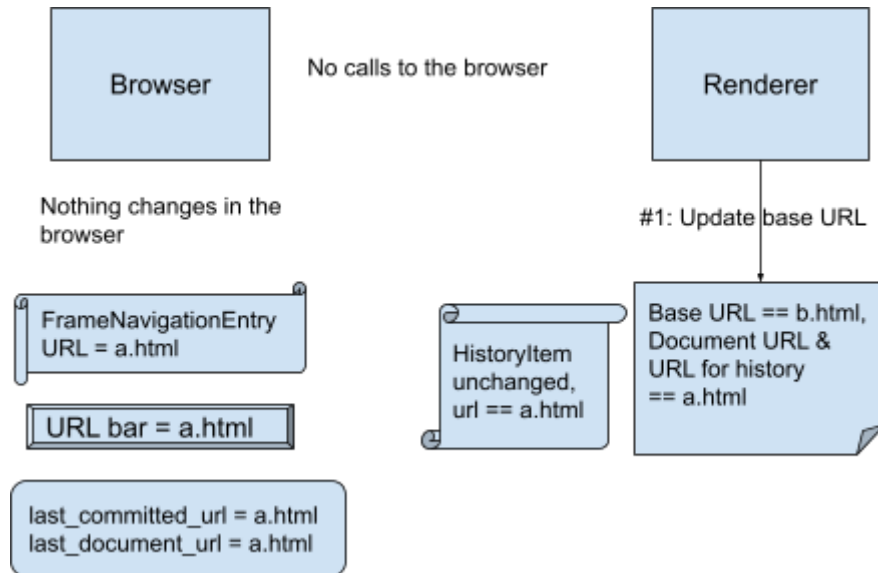
document.open()

(assuming this happens after the “Normal Navigation” case above, with the document.open() updating the document URL to b.html)



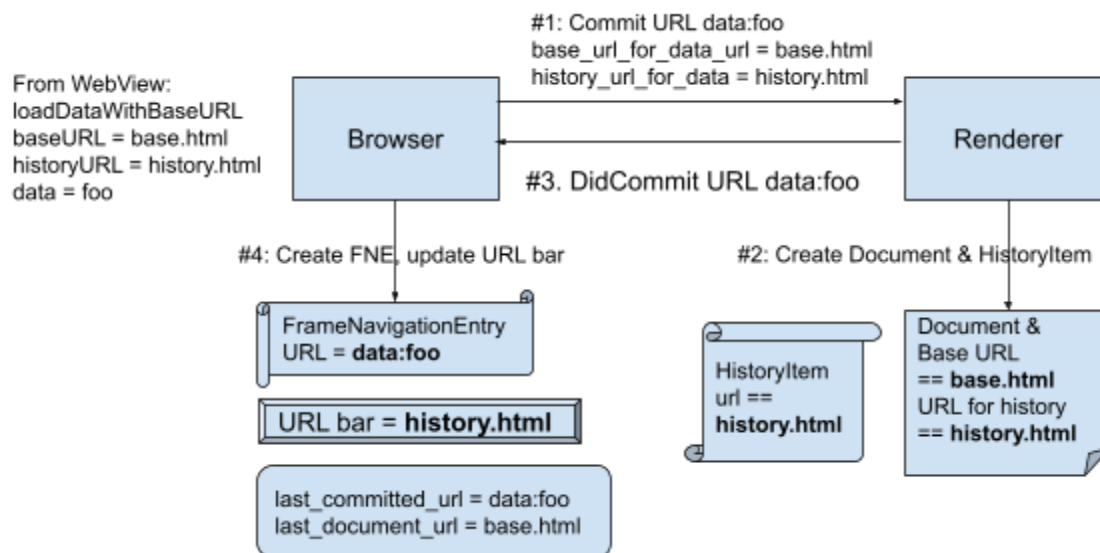
Base URL update

<base> element etc can change the base URL to b.html. This does not change any other URLs and does not notify the browser in any way. **Interesting note:** since the base URL is used for URL resolving, clicking on relative anchors (e.g. #foo) will probably create a cross-document navigation instead of a same-document navigation (if it happens after the navigation illustrated below, it starts a navigation to b.html#foo instead of a.html#foo)



loadDataWithBaseURL cases

loadDataWithBaseURL ([Android WebView](#) & [<webview>](#)) is very [interesting](#). It commits a data: URL, but the document & base URL is set to the supplied "base URL", while the HistoryItem in the renderer uses another URL: the "history URL" (but the FNE/session history entry in the browser uses the data: URL).



Detailed version, with potential problems in bold:

- In the browser, before commit:
 - The data: URL is used as the “navigation URL”. Note that the data URL might contain [empty data](#) (the URL is just the data header) if data_url_as_string is used (this is used for when the data is very long and can’t be sent through IPC as a GURL)
 - The supplied base URL & history URL is set as “base_url_for_data_url” and “virtual_url_for_data_url” for both [Android WebView](#) & [<webview>](#)
 - When creating NavigationEntry for the navigation, the base & history URLs are [saved](#) to the NavigationEntry.
 - When creating NavigationRequest for the navigation, the base URL is set [directly](#) but the history URL is [only set when the base URL is set](#)
 - Note: on history navigations (navigation to pre-existing NavigationEntry), the base URL & virtual URL saved in the NavigationEntry is used.
Potential problem: this is set on subframe navigations too, so subframes in loadDataWithBaseURL navigation use this! Probably this is accidental.
 - Most things in the browser determine if a navigation is a loadDataWithBaseURL navigation or not by [checking if the base URL is empty or not](#)
 - Used for various origin/permission-related stuff, where the base URL is used instead of the data: URL:
[GetOriginForURLLoaderFactoryUnchecked\(\)](#), [Origin Isolation](#), [UpdatePermissionsForNavigation\(\)](#), etc.
 - **Potential problem:** see 1 problem above. We set the base URL on subframe history navigations. The method above doesn’t check if it’s a main frame navigation, so probably some of the current usage is problematic?
 - **Potential problem:** see the renderer section, where the checks in the renderer side are slightly different. Is this intended?
- In the renderer, while committing:
 - The renderer determines if a navigation is a loadDataWithBaseURL navigation with a [slightly different check](#) from the browser (it adds main frame check, and even if the base URL is empty but “data_url_as_string” is used it will go through the special loadDataWithBaseURL path). If the navigation doesn’t pass the check it will commit a normal data: URL navigation (using the data URL as the document/base URL too). Otherwise, the supplied base URL and history URL will be used by the renderer:
 - The “[document URL](#)” will [use](#) the supplied base URL (falling back to the data: URL if it’s empty)
 - The “[unreachable/history URL](#)” will use the [supplied history URL](#) (which might be empty)
 - **Potential problem:** This check is different from the browser side, most notably the “data_url_as_string”. Probably it’s there to ensure that data_url_as_string is processed (the data is used instead of from the empty data: header in the CommonNavigationParams url)

- Another way the renderer determines if a navigation is a loadDataWithBaseURL navigation is through a third check, different from the other two.
 - This checks if the history URL is empty or not and whether the URL is a data: URL or not, but doesn't check for "data_url_as_string" and whether the navigation is on a main frame or not.
 - This check determines whether the data: URL should be saved or not, to later be [returned](#) as the "url" in DidCommitProvisionalLoadParams (see ["Loading" URL](#))
 - **Potential problem:** This check is different from the browser side and the other renderer side check, probably by accident. Some weird cases this introduces: if the history URL is empty, the DidCommit URL will be set to the base/document URL instead of the data: URL. This will never use the history URL but the history URL somehow affects this...
 - **Potential problem:** This "return the data: URL" rule applies as long as the document stays the same as it's saved in [DocumentState](#). This means if a same-document navigation happens the DidCommitProvisionalLoad URL stays the same (the original data: URL).
- The document URL (base URL if it's not empty, falls back to the data: URL otherwise) is saved as the DocumentLoader's [url](#).
 - **Potential problem:** This means fragment navigations initiated in the renderer and pushState/historyState checks will be based on the document/base URL. This is different than browser-initiated fragment navigations which will be based on the data: URL (see browser-after-commit section)
- The unreachable URL (history URL in most cases, falls back to the document URL otherwise) is saved as the DocumentLoader's [unreachable url](#), which is used by [UrlForHistory\(\)](#).
 - As noted in ["Document URLs"](#), navigation decisions should be made based on the document URL, but in chrome it's currently based on the "UrlForHistory()".
 - **Potential problem:** this means two back-to-back loadDataWithBaseURL navigations with the exact same data:, base, and history URLs will not be treated as same-URL, since it compares the second navigation's document URL (base.html) with the first navigation's history URL (history.html).
 - The HistoryItem's url uses UrlForHistory()
 - **Potential problem:** This is different than what the browser uses in FrameNavigationEntry (see the browser-after-commit section)
- If the navigation fails (NavigationThrottle can block this type of navigation) the base URL & history URL will not be used, instead it will go through the normal
 - **Potential problem:** does error page isolation kick in for this case? What origin is used?
- In the browser, after commit:

- The DidCommitProvisionalLoadParams url (in most cases, the data: URL) is used as the URL for the FrameNavigationEntry
 - **Potential problem:** The FrameNavigationEntry URL (most likely set to data: URL) can be different from the HistoryItem URL (most likely set to the supplied history URL, falls back to the base URL).
- The virtual URL will use the NavigationEntry's virtual URL, which is set to the history URL.
 - **Potential problem:** The virtual URL will [fall back to the commit URL](#) (the data:URL). Maybe this should fall back to the base URL instead?
- The last committed URL is set to the DidCommitProvisionalLoadParams url (in most cases, the data: URL). Most decisions in the browser are based on this, unless they explicitly check the BaseURL (and not many do?)
 - **Potential problem:** Same-document navigation behaves differently in the browser vs renderer. Because the last committed URL is the data: URL instead of the document/base URL, browser-initiated fragment navigations to the base URL will be converted to cross-document navigations (because it doesn't pass [this check](#) against the last committed URL). If the navigation is renderer-initiated instead, it will be [compared against the document/base URL](#) and successfully committed as same-document.

Questions

- WDYT about the potential problems above? Which ones are intended vs accidental?
- What are the guarantees given for the base URL & history URLs? Can they be empty/invalid/etc? Do we need to care about these cases?
 - The Android WebView [documentation](#) says these URLs will default to about:blank, but probably it's always possible to provide invalid URLs which will be converted to empty URLs?
 - The <webview> [documentation](#) does not mention any defaults.

Potential next steps?

- File bugs for the unintended cases.
- Improve the documentation on how this works.
 - First attempt: crrev.com/c/2971664
- Align the checks in the browser and renderer
 - crrev.com/c/2971664 aligns the two renderer-side checks.
 - Probably needs more investigation for the browser-side checks, including clarifying what origins should be used in various cases...
- Reconsider if we need to send the history URL to the renderer at all
 - The documentation says the history URL is only intended to be used as “the URL to show the user” so probably the renderer shouldn't make decisions based on it
 - The HistoryItem URL is different from the actual session history entry URL. Probably we should just make it the same.

- Outside of HistoryItem, UrlForHistory() is only used for some same-URL checks, and the for loadDataWithBaseURL this seems wrong (it will check document/base URL against history URL), we should make it compare document URL against document URL instead.
 - According to <https://crbug.com/1088547>, the usage of history URL is pretty low so these changes are probably not too risky?
- Remove data_url_as_string special cases
 - As noted in the “renderer” section above, the renderer check checks for [data_url_as_string](#) while the browser checks don’t. Probably it’s there to ensure that data_url_as_string is processed (the data is used instead of from the empty data: header in the CommonNavigationParams url) and only the loadDataWithBaseURL special path has that processing, so it must go there.
 - From alexmos@: Instead of adding that to the renderer-side check, probably we can immediately set the CommonNavigationParams url to the data_url_as_string URL as soon as it arrives in the renderer (since its only purpose is to circumvent the IPC limitations on GURL). Then we add a check before sending the URL back to the browser in DidCommitProvisionalLoadParams that changes it back to just the data: URL header.
- Fix other bugs identified..