Publicly shared

Author: kenjibaheux@chromium.org, toyoshim@chromium.org

Reload, reloaded

Background

Chrome is apparently performing a lot of unwarranted revalidations. While it might not explain it all, we found out a couple of contributing factors.

As we looked into the code to find more causes, we couldn't help but noticed the multitude of Reloading flavors:

- Normal navigation: regular *Load*
- Same URL navigation: *Load validating main resource*
- Pull to refresh: *Load validating all resources*
- Reload button: *Load validating all resources*
- Shift/Ctrl-click the Reload button (OS specific): *Load bypassing cache*
- F5, ctrl+r/cmd+r (OS specific): *Load validating all resources*
- Shift + [ctrl+r / cmd+r / F5] (OS specific): *Load bypassing cache*
- Third mode in Reload's context menu when DevTools is opened: *clear cache then Load bypassing cache*
- Tab recovery:
  - *Load preferring cache* on Android
  - *Load validating all resources* on Desktop
- Reopen closed tab: *Load preferring cache*

The big picture of loading behaviors

Blink	target	referrer update	form reset	update history	main frame	resource
location.reload	frame	yes	yes	no	validate cache	validate cache
history.go(0)	location.reload					
history.go(n!=0)	History Navigation					
KeyboardEvent	History Navigation					
HTMLAnchorElement	frame	yes	-	yes	protocol	protocol
InspectorPageAgent::reload(ignore cache)	page	no	yes	no	validate/bypass	validate/ignore
Inspector internal reload	-	-	-	-	only from cache	only from cache
Content/Chrome						
Reload	page	no	yes	no	validate cache	validate cache

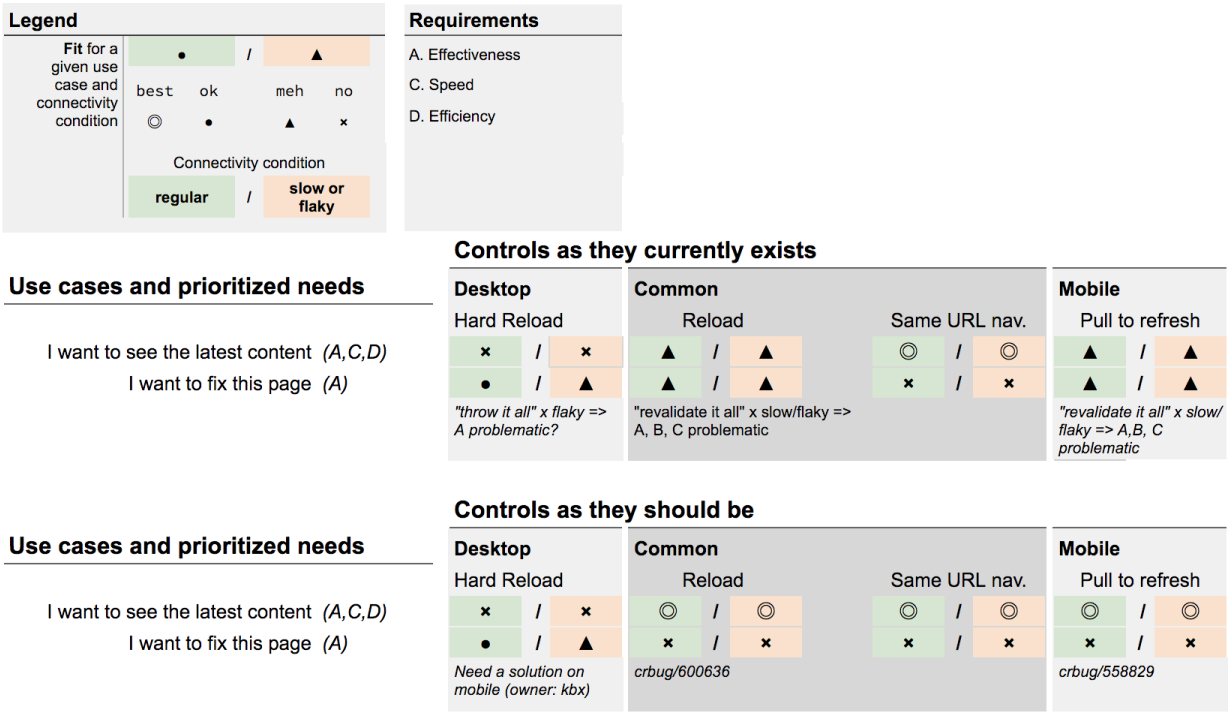
Bypassing-Reload	page	no	yes	no	bypass cache	bypass cache
Context Menu Reload / Ctrl+R	Super-Reload					
Context Menu per-Frame Reload	frame	no	yes	no	validate cache	validate cache
Pull to refresh (expected)	page	no	yes	no	validate cache	protocol cache
History Navigations (GET)	page	no	no	no	preferring cache	preferring cache
History Navigations (POST)	page	no	no	no	only from cache	only from cache
Omnibox - Enter	page	yes	yes	no	validate cache	protocol cache
Tab Restore	History Navigation					
ContentSettings update	Reload					
NetErrorHelper (auto redirect from dinosaur)	Reload					

Use cases, current state, desired state

I believe that there are only 2 user facing use cases:

1. I want to see the latest content
2. I want to fix this page

Here is an overview of what the current state means for these use cases and what I think we should aim for:



Observations about the current state

The “revalidate it all” approach used for pull to refresh and reload is suboptimal for the use case it’s intended for (i.e. “I want to see the latest content”). Especially on slow/flaky connections, we end up revalidating perfectly fine assets. This leads to slow page loads in the best case scenario or failed page loads in the worst case scenario (e.g. timed out, connection drops).

The “throw it all” approach used for Hard reload might be problematic on a flaky connection. It’s unclear how much of an issue this is in practice but a workaround probably consists of hard reloads + reload combos.

Proposal

Keep only 2 user facing reload behavior:

- *Load validating main resource* (i.e. validate the main resource + regular *Load*)
- *Load bypassing cache*

Align system reloads (e.g. tab recovery, reopen closed tab) to one behavior:

- *Load preferring cache*

Unchanged

- Same URL navigation: *Load validating main resource*
- Shift/Ctrl-click the Reload button, Shift + [ctrl+r / cmd+R / F5]: *Load bypassing cache*
- Third mode in Reload’s context menu when DevTools is opened: *cache cleared then Load bypassing cache*
- Reopen closed tab: *Load preferring cache*

Changes

- Pull to refresh: *Load validating main resource*
- Reload button, F5, ctrl/cmd+R: *Load validating main resource*
- Tab recovery: *Load preferring cache* on all platforms
- Introduce UX for *Load bypassing cache* on mobile

Unclear

NetErrorHelper (auto redirect from the Offline Dinosaur): change it from *Load validating all resources* to:

- *Load*?
- *Load validating main resource?*
- *Load bypassing cache?*

ContentSettingsUpdate: change it from *Load validating all resources* to:

- *Load*?
- *Load validating main resource?*
- *Load bypassing cache?*

JS reload()

- current behavior appears to be a Super reload but MDN says that the browser may load from the Cache.
- MDN also mentions a (non-standard?) parameter to specify the behavior:
    - When the parameter is true, it means that the browser must “load from the Server” which I interpret as a hard reload.
    - we currently don’t support the parameter.

Behaviors in other browsers

- Internet Explorer’s behavior
- *Others* TODO

Measuring success

Changing the longstanding behavior of Reload is not without risk. It’s possible that some websites have come to rely on said behavior as part of their intended user-experience. For instance, one could imagine a website with a dynamically-updated image that's short-term cacheable. Or maybe a rotating logo and the user wants to cycle through them. Or perhaps an article with short-term cacheable assets that are frequently updated in the context of live blogging an event.

There are obviously better ways to achieve the same effect (e.g. a live blogging service) but these don’t apply to websites that are still used but not updated.

Metrics

Benefits

We expect to see the following improvements:

- Fast Loading user experience (observable via Page Load Time metrics)

- Data savings (observable via Net metrics)
- Power consumption savings (not sure how much to expect; no metrics?)

User frustration

In the case where the new reload behavior didn't fulfill the user's desire, we expect to see an increase of reload actions, in particular fast-follow reloads. It's probably risky to only measure the number of reload actions because if reload becomes significantly faster, it might be used more lightly than it was before.

Strawman:

- Histogram showing the # of reloads triggered per document
- *Histogram showing the time between subsequent reloads per document*

Underlying impact

In addition, we can measure the underlying impact of forced revalidations, or the lack thereof:

- Histogram showing the outcome of a force revalidation via a classic reload:
    - not updated
    - updated

For picking an adequate risk mitigation option:

- Histogram showing the max-age of assets that were updated as the outcome of a force revalidation via a classic reload.

Caveat: this doesn't tell us if the updated resource mattered to the user, hence the need for a metric that measure user frustration as explained above.

Risk mitigation options

Depending on what we learn from the behavior change outlined in the first part of this document, we might need to make some adjustments to mitigate risk.

Async revalidations

Instead of performing regular validations that hurt the loading user experience, opt for async validations.

Pros:

- a subsequent reload will use fresh assets, therefore fixing any user frustration.
- Improvements on the Loading user experience are maintained

Cons:
- Improvements on data usage (for users and webmasters) are gone.
- Improvements on power consumption are gone.

Key on long max-age

Assume that assets with a relatively long max-age are unlikely to be updated.

Pros:
- simplicity

Cons:
- Improvements to the Loading user experience are largely gone.
- Improvements on data usage (for users and webmasters) are largely gone.
- Improvements on power consumption are largely gone.

UX solutions

Currently, there is no hard reload affordance on Chrome for Android.
Beside the mitigation risk benefits, there are situations for which a hard reload comes handy.

Offer a hard reload UX affordance on all platforms

Pros:
- Fix all the issues
- No compromises on the regular reload

Cons:
- Discoverability issues (esp. on Mobile)

Piggy-back on user frustration to fix things

Change the behavior of a subsequent reload to help the user:
- Reload: as outlined in the document
- First fast-follow subsequent reload: reload with validations
- Second fast-follow subsequent reload: hard reload

Pros:

- Eventually, fix all the issues
- No compromises on the regular reload

Cons:

- Defining “fast-follow” might be tricky: when the page is seemingly stuck loading, users tend to perform a subsequent reload for which a hard reload or even extra validations would be unwelcomed.
- Perhaps, only consider subsequent reloads that happen after the first meaningful paint (=> dependency).
- Complexity => user not in control

<u>Variation</u>

Change the behavior of a subsequent reload to help the user:

- Reload: as outlined in the document
- First subsequent reload: reload with async validations
- Second subsequent reload: as outlined in the document