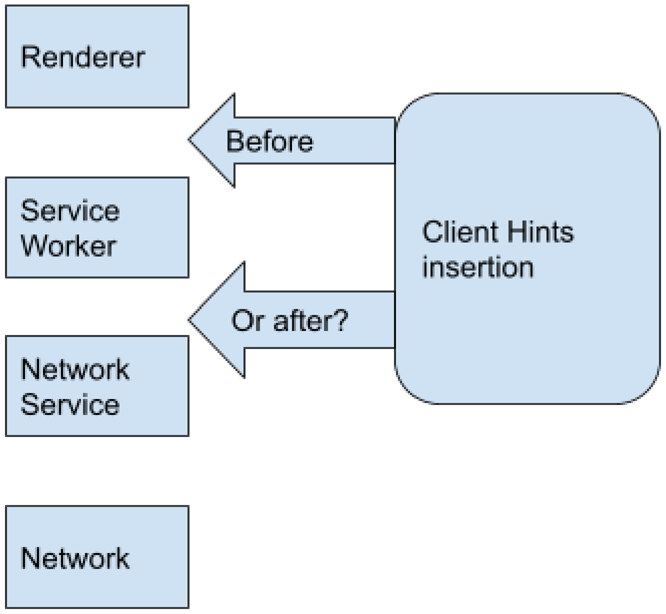
yoavweiss@ - September 2020
Last update: November 3rd 2020

Public document

Introduction

Client Hints and Service Workers seem to have a couple of contradicting requirements.
On the one hand, we want Client Hints to be visible on Requests that Service Workers handle, to enable them to properly cache their responses with the right cache keys, in case those responses `` `Vary` `` on the hints.
At the same time, we want Service Worker generated requests to include Client Hints in their requests.

This raises the question: should we add client hints to requests **before or after** Service Workers?



Beyond that, some of the hints (e.g. `Viewport-Width, DPR, Width`) don't really make sense outside the contexts of a renderer: A user may have multiple screens or multiple open windows, and viewport and DPR may vary on a per-renderer basis. Width is even more special, as it is specific to an image `Request` coming from a renderer.

Current status

Client Hints

Currently, Client Hints are specified to be added before Service Workers, to make sure they are visible to the SW fetch handler.
Because of the way Fetch is specified and layered, adding the hints before the SW also means that the same step is called for SW-generated requests, as they simply call the Fetch processing model again. At the same time, that second call to append client hints to request is somewhat different than the first:

- The request's client is no longer the Document's environment settings object, but the SW's `WorkerGlobalScope` (TODO: verify that this is accurate)
- Permission Policy's Should request be allowed to use feature bails when the request's window is not a `Window`. That means it would return false in the SW-generated requests case.
- It's unclear what hints should be added for renderer-specific features. (e.g. `Viewport` or `DPR`)

User-Agent and Cookie

It also seems that similar content-negotiation flows, based on the `` `User-Agent` `` header as well as `` `Cookie` `` are not currently properly handled in SWs, as those are added to requests **below** the Service Worker.

Service Worker code is expected to handle those out-of-band.
It should be noted that this has not come up as a significant issue so far.

Cache.match()

The cache.match() API automatically implements `` `Vary` `` functionality, enabling the SW to take content negotiation into account. If the SW doesn't see the content-negotiation based values, that part loses its meaning.
That is already the case with `` `User-Agent` `` and `` `Cookie` `` based negotiations, at least per spec. Some implementations vary (pun not intended) in their behavior. The same is true for the quality of the vary implementation.

Problems

- SW-generated requests do not contain hints, making them unreliable
    - We would need to fix Permission Policy and its interaction with workers
- Some hints are hard to define for SW generated requests (e.g. `Viewport` and `DPR` depend on the `Document` that initiated the Request)

- SWs have no way to generate requests that “pretend” to represent certain user conditions.
    - E.g. When sending viewport based requests, SWs cannot fetch resources based on other potential viewports (e.g. different device orientation). That can be harmful for offline readiness.
- SW’s `Vary` implementation may mislead developers and lead to confusion and cache matching issues
- For Client Hints reliability, it’s unclear what layer should handle the internal redirect.

Solution

- Adding Client Hints also after workers
    - The current spec Just Works™ when it comes to addition
    - Implementation
        - Add Client Hints in WorkerFetchContext::PopulateResourceRequest, on top of the current FrameFetchContext location
        - Add Client Hints after navigation requests in worker - needs some research
        - Sync Accept-CH cache for workers when values change
            - Requires spec changes
- Permission Policy for worker-generated requests
    - We need to change the Should request be allowed to use feature algorithm, both in spec and implementation
        - Phase 1
            - When the request.client is a `WorkerGlobalScope`, allow hints for same-origin requests, but not for cross-origin ones
                - That’s only because we have no policy, and that seems like the least bad immediate solution
        - Phase 1.5 ??
            - Find a way to collect and intersect policies from the Service Worker clients and apply that
            - Runs a risk of introducing races, and may require specification efforts, so might be worthwhile to skip this
        - Phase 2
            - Actually implement a Permission Policy for workers that would enable developers to control what hints their SW-generated requests are allowed to send.
- Client Hints Reliability
    - We need to add Fetch handling for Client Hints reliability’s internal redirect, and add another location where we call append client hints to request.
        - We can append the hints based on the pre-redirect `request.client`.
        - If the client is a SW, that means we won’t add Viewport/DPR/etc. That seems fine. If developers want that, they should pass-through those

navigation requests. If that's not sufficient (e.g. for navigation preloads), one can imagine a dedicated API for that purpose.

Considered alternatives

We must solve the issue of SW-generated requests not containing Client Hints. Other problems above can also be nice to solve. Below are a few alternatives:

Content negotiation after SW

- Add CH after SW
    - That would prevent us from requiring CORS exemptions for them and make them consistent with User-Agent and Cookie
    - We'd add the hints based on request.client
        - In the case of SW-generated requests, the client is the SW itself
        - WorkerGlobalScope do have a “client hints set” attached, but won't have any viewport, DPR, etc values
- Annotate Requests so that renderer-dependent values can be passed-through
    - If we're adding the values below SW, we need a way to pass them from the renderer
- Add API to add renderer-dependent values to SW-generated requests
    - If we want SW-generated requests to add renderer-specific values, we need a way for the SW to set those values. Since CH are (ideally) \`Sec-\` prefixed, the only way to set that is for the SW to flip flags on the Request itself.
- Expand \`cache.put()\` and \`cache.match()\` to be able to add various content-negotiation mechanisms, including Client Hints, User-Agent and Cookie
    - That would enable \`Vary\` to Just Work™ when dealing with negotiated content
    - All CH are today available through a JS API (even if not all in a Worker context), so SW code can access them
        - We may need to make all relevant JS APIs available in workers

Open Questions

- SWs get their Accept-CH cache when initialized.
    - Can we ensure they get updates when the user-agent's Accept-CH cache gets updated?
- Alternative considered(?): If all of the info relevant to a SW are available via JS APIs, do they reeeeeally need Client Hints?
    - Ergonomics would be nice (requests look the same from Fetch and SW fetch)
    - Is it worth discussing?

Draft notes

If we add a way for SW to indicate a navigation destination, can CH reliability work below the SW?
How can we update the SW’s Accept-CH cache when Accept-CH changes?