@yoavweiss - September 2020

Public document

This document intends to outline what are the different specification points that would be required in order to properly specify the various pre* primitives required for instant navigation - preload, prefetch, prenavigate and prerender.
Right now, it's not yet fully decided what some of those would look like. This document explores the specification implications of different options.

API shape changes

Preload

No API shape changes planned for preload

Link headers

No API changes planned.

Prefetch

Currently prefetch is allowed to fetch any resource type and bring it into the HTTP cache, where it will live for at least 5 minutes (unless it's a no-store resource). That includes cross-origin resources.
Cache partitioning will make that fetch operation useless, resulting in wasted bandwidth. Therefore, we may want to modify the API shape so that prefetch will only work for same-site resources as the top level origin.
That would mean that many prefetch indications on the web today would become no-ops, and some use-cases won't be enabled (e.g. prefetching a cross-origin resource to be used by a same-site page).
TBD what we actually want to do there.

Prenavigate

Prenavigate is not currently a thing beyond TPAC discussions. The proposal there was somewhat complex, so it might be good to coalesce it with prerender signals.

Uncredentialed Prerender

The old prerender triggered credentialed requests and was problematic in many ways (non-idempotency of links, privacy, etc)

We want a new uncredentialed prerender, requiring the prerendered page to:

- Opt-in to being prerendered without credentials
- Provide hooks (e.g. event handlers) that enable the app to switch from non-credentialed mode (before the user actually navigated) to credentialed mode.

The latter part would require apps to change the way they introduce personalization and many other features relying on user credentials. At the same time, it'd make their HTML more cacheable and potentially makes for a more robust site architecture.

The opt-in above seems relevant for manual prerender, P4, portals and potentially Fenced Frames.

For triggering prerender we have multiple options on the table:

- An implicit "prerender all the things" signal, potentially with URL patterns that are OK to pre-render
- A weak "this is OK to prerender" signal for a specific URL
- A stronger "This needs to be prerendered" signal (similar to link rel=prerender in the past)

If we want to coalesce prenavigate with prerender, we may want to add the flexibility for the browser to terminate prerendering navigations at certain points in time.

TBD - what we actually want to do there

WebBundle loading header

If WebBundle loading moves from `<link>` to `<script>`, we may want headers that can trigger that.
TBD working with partners to decide if and how that'd look like

Potential spec changes

Preload

Preload Cache

- Each Document has a Preload cache it's associated with
- In main fetch before step 5 or in HTTP fetch before step 3, add a check to see if the request is in the preload cache as well as if all the request parameters match the response in the cache. If it is, return the response object from there.
    - There are many devils in the details of "request parameters match the response" and browser compatibility. But what Chromium does may make sense as a starting point.

Link Headers

Processing model

- In HTTP fetch, before step 6, inspect `actualResponse`'s headers, and process any Link headers, and trigger the corresponding processing model depending on the link (for supported ones)
- Redefine the processing of preconnect, preload and prefetch in HTML to not require any element, and call them from Fetch

Prefetch

- Integrate processing with HTML/Fetch, define `Purpose`
    - Modify existing PRs based on what we actually want to do and land them: Fetch, HTML

Uncredentialed Prerender

Referring page opt-in

Depending on what we want, we would need to define the opt-in's shape.
In case the opt-in is implicit and requires the UA to scan the document, we could provide hooks in <a>/<area> processing to do that.
Maybe something like:

- When an <a> element becomes connected, we run a check to see if it is safe to prerender. If not, return
- If the UA thinks that the link is worthwhile to prerender, run Prerender processing model

Referred resource opt-in

Need to define the shape, which then runs Prerender processing model

Prerender processing model

- Run `follow the hyperlink`
    - We'd need to modify `follow the hyperlink` to take in the target as a parameter, and set the input target to be a "new prerendered tab" (or some other new target).
    - We'd also want this algorithm to call `navigate` with a flag indicating that it's a prerendered page

Navigate changes

If `navigate` was called with a “prerendered” flag, modify navigation processing to:

- Make sure the fetch is not credentialed
- Require a referred opt-in (or terminate the navigation)
- Add hooks enabling the UA to stop and resume that navigation at various points (to enable a “prenavigate” effect at the UA’s discretion (e.g. when it considers resources scarce, or signal from developer is weak).
- Ensure the resulting document is a “prerendered document”

TODO: figure out *all* the details

Prerendered document

- Ensure the document has no access to credentials and shared storage
- Integration with session history??
- Disabled permissions?
- Fire an event when the user actually navigates??
- Visibility state

TODO: Fill in even more details

WebBundle loading header

????