(Visibility: public. Initial version created by: Henri Sivonen)

rel=preload in the HTML Parser and DOM

Assumption: The completion or failure of a preload fetch fires a load or error event on the link DOM node associated with the preload as well as the on the e.g. img element that uses the resource. (The rel=preload spec refers to the HTML standard in a way that results in events firing on the link element.)

> Yes. The events on the actual element should be unaffected. The events on the <link rel=preload> element should behave as described: onload on success, onerror on any kind of load error (this is easy to decide)

Assumption: Preloads need to be able to be signaled as having a higher networking priority, since Chrome gives them a higher priority.

> Yes. That is the basic idea, to let all the respective preloaders know this is link/preload initiated speculative load and possibly give it a higher priority. We may not necessarily mirror what Chrome does but keep in mind that web pages are optimized for Chrome and we should not at least hurt performance with our approach.

Requirement: When HTML source shows both a rel=preload and <img src> or similar pointing to the same URL, don't start two distinct network fetches.

> Yes. This is ensured by the existing coalescing logic for speculative loads in the HTML5 parser code, here.

Consequence: To avoid starting distinct network fetches for HTML-source-appearing rel=preload and <img src>, rel=preload has to go into the speculative load queue after which the <img src> is considered a duplicate.

> As said above, we already have code that ensures this.

Consequence: We need a way to associate rel=preload-originating speculative load queue items with the eventually-created DOM link element nodes.

> Yes! This was a work in progress, but later we (Dragana) found it was not good enough for an edge case when and actual element is in the markup before the related

> link/preload element (a web page bug, but we have to handle it correctly - deliver events, not duplicate requests)

Observation: It's no good to give start tag tokens an identifier that carries over to the node insertion time, because document.write() can invalidate the tokens.

Solution: Flag speculative load queue items as rel=preload-originating. (Propagate this information to Necko for priority setting.) When the tree op executor creates a link element node whose preload type and URL match a preload fetch, associate that fetch with the link element node. If the fetch has already finished, post a runnable to fire the load/error event as soon as the event loop spins.

There probably needs to be distinction between parser-initiated rel=preload fetches and DOM scripting-initiated rel=preload fetches. In the latter case, the DOM node associated with the fetch exists from the start. Only parser-inserted link DOM nodes need to find an association with parser-initiated rel=preload fetches.

Question: What should happen if there are two <link rel=preload>s with the same attributes? What events, if any, fire on the second one?

> These are good questions. The specification doesn't say anything. I believe we should not start a new load if the attributes are identical. I believe we should fire events on both/all. No idea what the current implementation does, but I believe it's too broken to do things right, anyway.