LCP WebPerf Implementation

This document is public
npm@
April 2019

This document contains implementation details for Largest Contentful Paint (LCP) Web Perf API. An explainer for LCP can be found here. As of right now, the precise shape of the API has not been decided so this is a rough draft that tries to avoid going into those details.

We consider the following objective: expose LCP for each frame, leaving room to expose different information depending on whether the content being exposed passes TimingAllowOrigin (TAO) checks.

There are two alternatives from which we could expose: ElementTiming code or UMA/UKM LCP code. We have decided to expose via the latter because it will be simpler.

If we added the hooks via ElementTiming code, we would need to add the following logic to that part of the code:

- Keep track of the largest element seen thus far (not needed right now by ElementTiming). This would be relatively simple.
- Remove from consideration elements that are disconnected from the DOM tree. Currently ElementTiming does not consider this at all, although it will with the addition of the ‘element’ attribute (which needs to return null once an element is disconnected). This part is relatively complex because it means we need to keep track of a set of largest candidates instead of just the largest, in order to update the largest if the largest is disconnected. This logic is already implemented in UMA/UKM LCP code.

Due to this, we consider it simpler to add the hooks via UMA/UKM LCP code.

Adding the hooks via the UMA/UKM LCP code means we need to augment that code as follows:

1. Hook into the LCP WebPerf class from both ImagePaintTimingDetector::UpdateCandidate and TextPaintTimingDetector::UpdateCandidate because the image and text candidates are considered separately. The LCP Web Perf class will keep track of the largest image and largest text. Any time the candidates are updated, the ‘largest’ might be updated. Note that just keeping track of the largest overall is not good enough. For example, if the largest image is disconnected and the new largest image is smaller than the current largest text, then we need to update LCP to be the largest text.
2. Include enough information in the ImageRecord and TextRecord for the hook into LCP WebPerf class to have all the information it needs. Because the shape of the API is

known at the moment, it’s unclear what this information is. However, it would be expected for the following to be the minimum:

  - Size of the LCP
  - Paint timestamp of the LCP
  - Element associated to the LCP
  - Image url if the content is an image (note that this is not recoverable from the Element because it could be a background image, in which case the element does not necessarily provide information to determine which is the URL).
  - Enough information for the TAO check if the content is an image, also not recoverable from the Element.

It seems that most of this is there already. The size and paint timestamp are explicitly there, and the Element can be obtained from the DOMNodeId. The main missing components are the image URL and the information to compute TAO checks for images. The image_url is available in DEBUG mode, but since this is not enough to compute the TAO check I would recommend just adding the pointer to the ImageResourceContent to ImageRecord because that class contains both the URL and the ResourceResponse (needed for the TAO check, see how it is done in ImageElementTiming).