# DOM Timing proposal draft

alternative naming suggestions welcome :)

## Introduction

The idea is to measure a metric akin to paint times, but for specific DOM elements. This allows creating custom visual performance RUM metrics tailored to a website. Eg. “time-to-logo, time-to-first-paragraph, time-to-banner”.

This new metric would help bridge the gap between synthetic visual metrics and RUM. Currently the only visual metrics available as RUM are disconnected from the content in its semantic form. firstPaint, firstContentfulPaint only worry about the first *anything* on screen, where in practice some elements are more important than others visually.

Elements of high visual importance have a higher weight in users’ performance perception, as demonstrated by research that used eye tracking to prioritize loading the elements that were most looked at. Being able to select DOM elements also frees us from the limitations of the above-the-fold cutoff often used by visual metrics. Viewport sizes vary a great deal between clients, picking one set of viewport dimensions to generate a metric is an arbitrary choice. And with existing APIs like the Intersection Observer, it’s possible to know to what extent an element watched by DOM Timing is above the fold for the current client.

Having a metric that focuses on when content is actually done rendering would be bringing us closer to the point in time that satisfies users’ expectation (the content they’re waiting for is actually displayed). The fact that the browser has the DOM, or that the first pixel of anything has been painted, doesn’t tell us how long actual fine-grained pieces of content took to render after that point. Users aren’t waiting for the browser to know about the DOM, or for the first non-blank pixel to be displayed, they’re waiting for actual content. The closer we can get to users’ needs, the more likely the metric is to be correlated with people’s perception of performance.

The timespan between firstContentfulPaint and something as significant as the top paragraph of a blog article being rendered is something we can’t measure with RUM at the moment. Tricks like inserting inline JS in the text content don’t account for the various kinds of buffering at the browser level that can create significant delays between parsing the DOM and its inline JS, and actually flushing a chunk of the rendered DOM to the viewport.

# Response header

In order to avoid the overhead of always recording timings for all DOM elements, this feature would rely on a response header. When the DOM-Timing response header is present, the browser knows to monitor the painting of this particular DOM selector, or list of selectors.

Example:

```
DOM-Timing: div#p-logo > a.mw-wiki-logo, div.mw-parser-output > p:nth-of-type(1), div#centralNotice > :first-child
```

On English Wikipedia, these selectors would monitor the Wikipedia logo, the first paragraph of a Wikipedia article and any banner that might be dynamically inserted into the page (fundraising/community announcement).

# Measurement

As soon as the browser is made aware of that response header, it will watch for that list of query selectors. As soon as the DOM elements represented by a given selector are fully painted, a monotonic performance timing is recorded on the Performance Timeline to represent that.

It doesn't matter if the element is in the visible viewport of not. The definition of "painted" is that if it is (or would be) above the fold, this is the point in time when the user would be able to see it fully rendered. This is an important distinction from existing APIs that tend to ignore the potential delay between the browser having an intent to paint something and when the user actually sees what they're looking for, i.e. the actual content.

If multiple elements match a given selector, each of them results in a PerformanceEntry, when they are individually painted. This allows the API to work for dynamically inserted DOM elements as well, which might happen after the initial page load.

If an element contains other elements, the timing is when all its child elements and the monitored element itself are fully painted.

# Performance timings

The measurements made by this API would be added to the Performance timeline in a new category: https://www.w3.org/TR/performance-timeline-2/

Each timing would be a PerformanceEntry
https://www.w3.org/TR/performance-timeline-2/#the-performanceentry-interface

Where the “name” is the watched selector, “entryType” is “dom”, “startTime” is the timing and “duration” is 0. “Visible” indicates whether at the time of the event the element was visible or hidden.

It might be worth considering this new type of PerformanceEntry to also include an extra attribute pointing to the actual DOM element that results in the entry, since the selector in the name can represent multiple elements. As it stands, in the case of multiple elements watched with a shared selector, one wouldn’t be able to tell which is which.

*And alternatively, the “duration” of the entry could be the time delta between DOM insertion and the rendering completion.*

Examples:

```
PerformanceEntry {
        entryType: “dom”,
        name: “div#p-logo > a.mw-wiki-logo”,
        startTime: 988.7999999991735,
        duration: 0,
        visible: true
}

PerformanceEntry {
        entryType: “dom”,
        name: “div.mw-parser-output > p:first”,
        startTime: 1024.6589871687987,
        duration: 0,
        visible: true
}

PerformanceEntry {
        entryType: “dom”,
        name: “div#centralNotice > :first-child”,
```

```
        startTime: 3365.9239028842390,
        duration: 0,
        visible: true
}
```

# Open questions

## How expensive would the overhead get?

This is only viable if the browser overhead is limited when monitoring a small set of DOM query selectors.

## How do you define the point in time when an element is fully painted?

There might be challenges with knowing when a DOM element tree is fully rendered. Can the browser easily match a CSS selector to the fact that itself and its children have been painted?

Animated elements. Maybe the answer is as simple as “when the first frame is painted” for that case.

## Are there any security concerns?

This should abide by CORS rules and disallow measuring content one doesn’t “own” in iframes, etc.

Is the ability to measure the delta between DOM insertion and actual rendering introducing a new class of side channel timing attacks? I believe this would be the first browser API that allows one to measure this particular time delta. Maybe the existing fuzzing in the Performance Timeline that was introduced in implementations  is enough to protect against this sort of thing, though.

## Why not an integral metric akin to SpeedIndex?

This would require frequent snapshots of the DOM elements as they’re being rendered, which would probably introduce significant overhead. Visual measurements are also required. This could quickly get complex when dealing with a deep DOM tree.