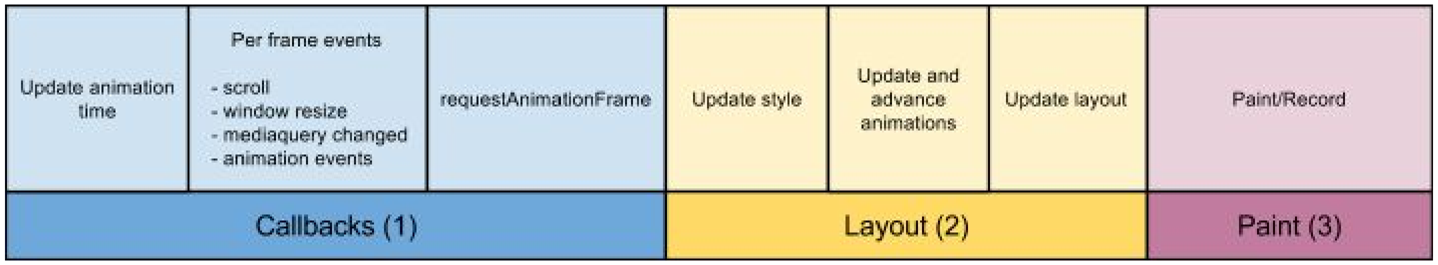
Rendering Processing Model

Introduction

Today browsers use inconsistent sequences of processing steps for rendering new content on the screen. This document proposes a stricter processing model that forms a pipeline. This processing model is easier for authors to reason about and provides consistency guarantees between various web platform features.

Pipeline stages



Exactly once per frame, the browser should run the pipeline from start to finish (i.e. left to right).

Any stage may be skipped if there's no current work to do, though earlier stages should be capable of causing dirtiness in the later stages of the pipeline. For example the *requestAnimationFrame* callbacks in the Callbacks (1) stage can modify style which will then cause the Layout (2) stage to run even if no style or layout was dirty at the start of the pipeline. Similarly sub-steps in each stage can create work for later sub-steps. For example calling *requestAnimationFrame* from inside a scroll event handler should be processed in the same frame.

No additional work should be processed between the stages of the pipeline, for example *setTimeout* and *setInterval* should never be processed between the *requestAnimationFrame* callbacks running in the Callbacks (1) stage and the Paint (3) stage. Similarly input events such as touch and mouse should never be dispatched during the processing of the pipeline.

Each stage of the pipeline processes the entire frame in tree order for for all same origin iframes ensuring that given two same domain iframes all *scroll* events and *requestAnimationFrame* callbacks have been processed in the parent frame before continuing to process the child frames.

Note that the style, layout and animations parts of the pipeline can also run outside of a frame. For example getComputedStyle will force the "update style" sub-step of the Layout (2) stage and offsetTop will force the entire Layout (2) stage, but neither one will cause the whole pipeline to run.

Author assurances

Most browsers do something close to this and we are trying to standardize on this actual pipeline since we believe it to be the most developer and performance friendly by making it clear when certain events and callbacks fire. This lets developers reason about their apps and write more interoperable code. Assurances provided by the pipeline include:

- DOM/Style changes made in timer callbacks never become visible to the user until after the next requestAnimationFrame callback (if any) has run.
- Paint related events fire only once per frame (e.g. scroll events).
- There's a 1:1 mapping between requestAnimationFrame callbacks and paints. You can't have a frame that paints without requestAnimationFrame firing first. **The exception here is threaded (async) scrolling and animations.**

Changes required to existing browsers

This section lists some known issues that would be considered bugs if we adopted this model. There are surely more we don't know about. Please add any others you know of.

In parentheses after each line are the browser that have the issue in question.

- Processing the entire frame tree for same origin iframes in each stage instead of setting independent timers for each one. *(Safari, Edge?, Firefox?)*
- setTimeout/setInterval/input events can't interject between requestAnimationFrame and painting. *(Safari, Firefox, Edge?)*
- scroll and window-resize events would fire once right before requestAnimationFrame. *(Safari, Firefox, Edge?)*
- requestAnimationFrame throttling would throttle the whole pipeline, not just the callbacks. *(Chrome, Safari, ~~Firefox~~)*
- Can't execute script during style/layout computation. *(Chrome, Safari, Firefox?, Edge?)*
- …?

Open questions

- Should changes in a requestAnimationFrame or layout that cause a scroll cause you to loop back through the pipeline and fire scroll events again?
- Should input events fire along with the scroll/resize/etc events? That would force them to be frame-aligned, which is what the majority of pages want, but a small percentage (e.g. games) may want touch events to not be frame-aligned.
- Should the order of scroll, window resize, mediaquery and animation events be specified?
- What about media playback?
- What about canvas, 2d and webgl?
-