Verified DOM rendering proposal

Jason Goodman

April 17, 2017

1 Overview

This project is a verified model of an optimizing browser renderer that creates raster images from DOM trees. In particular, this renderer will optimize for the case of rendering a DOM that is nearly identical to a previously rendered one.

The renderer's structure will be loosely based on this description of Blink's behavior: JavaScript and CSS rules are assumed to have been evaluated and some layout work done; the renderer then performs remaining layout calculations and "paints" multiple layers (e.g. absolute-positioned blocks) as raster graphics which are then composited together.

A naive renderer could perform all layout computations and paint each element recursively by compositing renderings of children. An immediate optimization is to avoid rendering elements—and potentially short circuit layout calculation—beyond the bounds of an element with invisible overflow. This targets the case of a large page with most content off-screen.

Differential optimizations will assume a single subtree has been replaced with a new subtree and make use of the previous iteration's layer paintings. Different changes will then trigger only compositing, repainting some layers and compositing, updating a layer's layout, repainting, and compositing, or completely starting over when conditions for optimization aren't detected.

As a concrete example, changing an absolute-positioned box's coordinates might only trigger a new compositing of existing layers, changing its background might cause its layer to be repainted, changing its dimensions might cause its layer's layout to be recomputed (if it has children), and replacing its children might entirely invalidate the previous rendering.

2 Property verified

The property being verified is that for all DOMs and subtree replacements, the optimizing renderer produces the same bitmap as an unoptimized reference renderer.

3 Capstone theorem

For the simpler case of a non-differential renderer, the correctness theorem is:

```
forall d : dom,
  reference_render d = optimized_render d
```

The differential case is less clean, taking an optional DOM diff (e.g. $([1;3], d')$ to replace the root's first child's third child with d') to apply paired with the renderer's output for the original DOM. This renderer outputs layout calculations and layer renderings in addition to the composited rendering, but correctness only concerns a projection of the composited rendering.

```
forall (d : dom) (diff : dom_diff),
  reference_render (apply_diff d diff) =
    extract (optimized_render d (Some (diff,
      (optimized_render d None))))
```

4 Project schedule

There are three weeks left. My goal is to have non-absolute layouts, layering, and the overflow optimization done the first week. I'll aim to have some differential optimizations working the second week (getting the model right is going to be a pain) and add more as time permits.

5 Division of labor

I will use this mapping from team members to shares of work.

```
Definition division_of_labor person :=
  match person with Jason => Everything end.
```

6 Risks

The biggest risk seems to be getting stuck trying to model and use complicated structures in Coq, but I think I have some idea of how to avoid painful situations.

It also won't possible empirically to analyze the performance of the optimized renderer except to say that it's based on performant browser architectures, since I'm not worrying about the performance of unary computation, tree comparison used in place of hashing, finite maps implemented with chained closures used in place of arrays, and other Coq-related hacks.

7 Future work

Two ambitious extensions would be to verify an actual browser renderer against a reference and to verify a reference renderer against a formalization of the CSS and HTML specifications.