# Edge conference New York 2013

Session notes for session 2: Rendering Performance

Moderator: ANDRE BEHRENS

Panel: JONATHAN KLEIN, PAUL LEWIS, ARIYA HIDAYAT, JOSHUA PEEK, ELI FIDLER

Notes scribed by: Ada Rose Edwards

## Introductory Text

‘As slick as native’ is a common boast for HTML5 developers. Web developers are not used to dealing in frame rates and memory consumption, but increasingly they need to. If UX is king, the web can only win when rendering is easier. We need great tools, skilled developers that know how to use them, and improvements in the platform to make the task easier in the first place. This session will address the challenges of rendering interfaces in a performant way in the browser.

- 60fps goal
- Animations and scrolling.
- 16.6ms per paint event. ..* Leads to Jank if >16.6ms
- Paint Events in chrome tells you how long it takes to draw a frame.

## Questions

### Is it possible to automate bottleneck testing?

- Very difficult to automate.
- Put checks in your code to ensure no one is doing anything known to be bad. e.g. A developer puts in request timeout.
- Warnings in the timeline. /!\
- Telemetry python based framework. Which adobe use/develop. (See paul lewis)
- No magic bullets. ..* Device specific and site specific problems.
- Being declarative rather than imperative (Css animations are good)
- No drop shadow, super expensive, even though popular!!!!

### TranslateZ is good now but may it eventually be an antipattern?

- Possibly...
- It's something that is rather not done but it fixes the problem now.
- Profile to see if it becomes a problem.
- Try an abstract it away so that it can be changed later.
- Forcing something to be rendered on a new layer should possibly be declarative not being forced using translateZ. ..* On the other hand it may not be something that devs should have to worry about. ..* Counterpoint: Everyone is doing it then perhaps it should

be a thing. ..* Still need to know that only a few properties can be run on the layer. e.g. border radius cannot be done by gpu. ..* Only Rotate, Translate, Scale, filter are done on the gpu. ..* If on layer only do those.
- Complex projects should have access to this.

### We have good tools in the browser but what can we get from the field i.e. from users.

- It would be awesome to get timeline data and memory data to read user performance data.
- How can we get code around the paints to find bad areas.
- How to differentiate info from just the browser os issues could affect fps but not be measurable by the browser so not viable in the real world today. Only applicable in a lab environment.

### Understanding how the browser work is easy for 1 man teams but for larger teams regressions could be introduced unknowingly. Can the DOM do stuff async and provide feedback to the code. (facebook react).

- There are a lot of cool stuff coming which will be fast e.g. offset width does not need to trigger a full page reload.
- DOM info being discussed.

### Selenium performance testing.

- API needed for paint events.
- Chrome telemetry allows an api.
- Setting up testing so that if your rendering gets slowed by a css you find out.

### How can you monitor 3rd party scripts to see if they are screwing stuff up with regards to render speed?

- Profile your app before/after and view difference.
- Badly written 3rd party code tends to be due to event handlers and other cruft.
- Profile to find these. E.g an onscroll event.

### What can browser vendors do to make it easier to profile sites?

- Because low power devices (e.g. glass) it gets harder.
- More tools are needed.
- Chrome keeps adding more tools to allow you to find more bottlenecks. E.g. Images doing stuff or drill down on paint events.
- Having triggers which can record what page is being janky on what devices. To allow further analysis.
- Does google reduce page rank for slow pages? (Audience laughs)

### What should developers take on in regards to Garbage collection and memory management.

- Avoid gc

- Use static memory pools to avoid object creation/removal.
- Having a gc api would be hairy. ..* Your page is not the only page in the browser and should not have the responsibility and the browser will try to do it when possible.
- What about a gc hint? ..* This would just be another hack.
- gc is an issue for games. If that is an issue for you then it is a hard problem. Memory problems are hard for games anyway deal with it.

### Image decodes and resizes are expensive avoid these as a bottleneck.

- Main thread code to intensive.
- Threads not always the answer. Work smarter gpu work to gpu. ..* this should be handled by the browser.
- try to avoid doing layout wherever possible.

### Highly image based webapp is slow even though it is simple what can be done?

- This is a very difficult problem!!
- Try batching your images. ..* One by one, request animation frame?
- What image types are faster?
- Image coprocessor?

### Scrolling to load images is a massive antipattern on mobile?

- Can the browser defer loading and decoding for when they pan in.
- Web audio for images?

### What have been then biggest wins and wastes of time?

Add property > read dom > goto 1

- That is really bad try to queue reads and rights.
- CSS selector matching is a big waste of time it's super fast in the browser.
- Sticking redraws to layer and doing isolation is super useful.
- Remove design elements. E.g. get rid of drop shadows and carosels. (Again yay for flat!!)

### Image decoding is expensive can this be done in a webworker?

- Paul Lewis wrote a library for it.
- This is a super bad idea!! It can be done but you will shoot yourself in the foot.
- Also will annihilate the battery.

### How to make non-devs understand performance?

- Show it to them on low end devices.
- Demo over cellular network.
- Show that is does impact engagement using facebook data.

### You need to get data from other people with large audiences to get this data. Currently not known data.

- Page load performance is less important than render speed.

### Cannot describe performance intent to the browser. E.g. priority of what should be rendered.

- Paul disagrees with 'the browser should do everything for me' spec compliant code should give spec compliant results. Speed is not in the spec browsers try to make it fast. Such an api would be difficult to make to the point where it can be polyfilled for additional functionality.