Scheduling JS Timer Execution

alexclarke@chromium.org   Tracking bug: crbug.com/463143

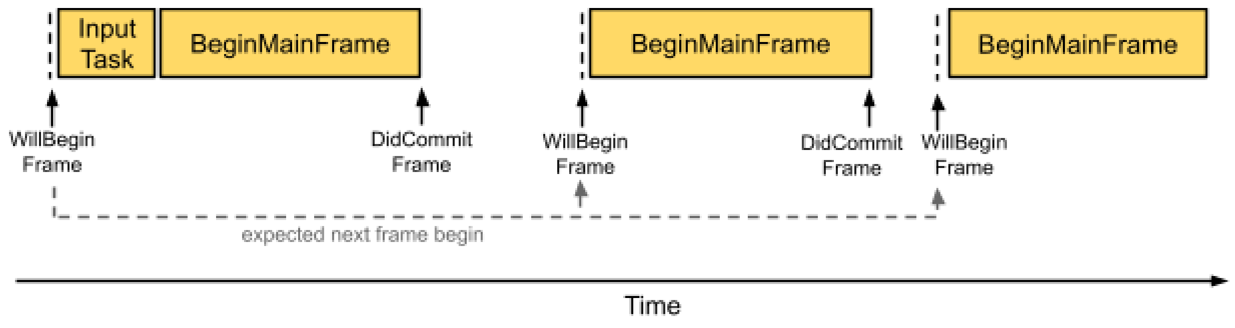
(Public document)

Goal

Reduce Jank, input latency and queueing durations by allowing the scheduler to defer javascript timer execution.

Background

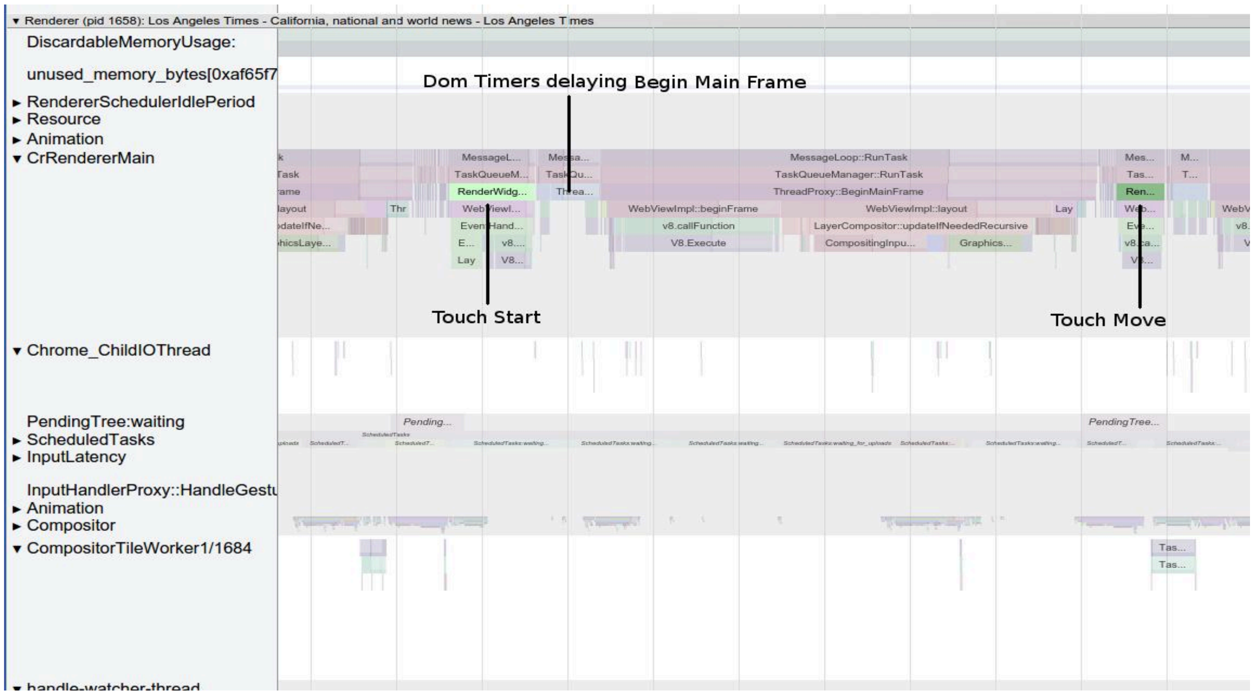
The render thread is often highly congested with a great variety of tasks vying for processor time. Input tasks and (sometimes) compositor tasks are on the critical path for smoothness and if we are optimizing for that (i.e. the scheduler is in COMPOSITOR_PRIORITY) we need to avoid delaying them.



As the diagram above illustrates, input and cc tasks come in at a predictable time and order (thanks to information sent by the compositor in RendererScheduler::WillBeginFrame). Currently interleaved with all this are loading tasks (such as html parsing), shared timer callbacks (mostly DOMTimers) and miscellaneous browser tasks. Most of the browser tasks are quite short and don't typically cause a problem. Historically loading tasks have caused jank, but these days they have their own queue and are run at a lower priority when the scheduler is in COMPOSITOR_PRIORITY.

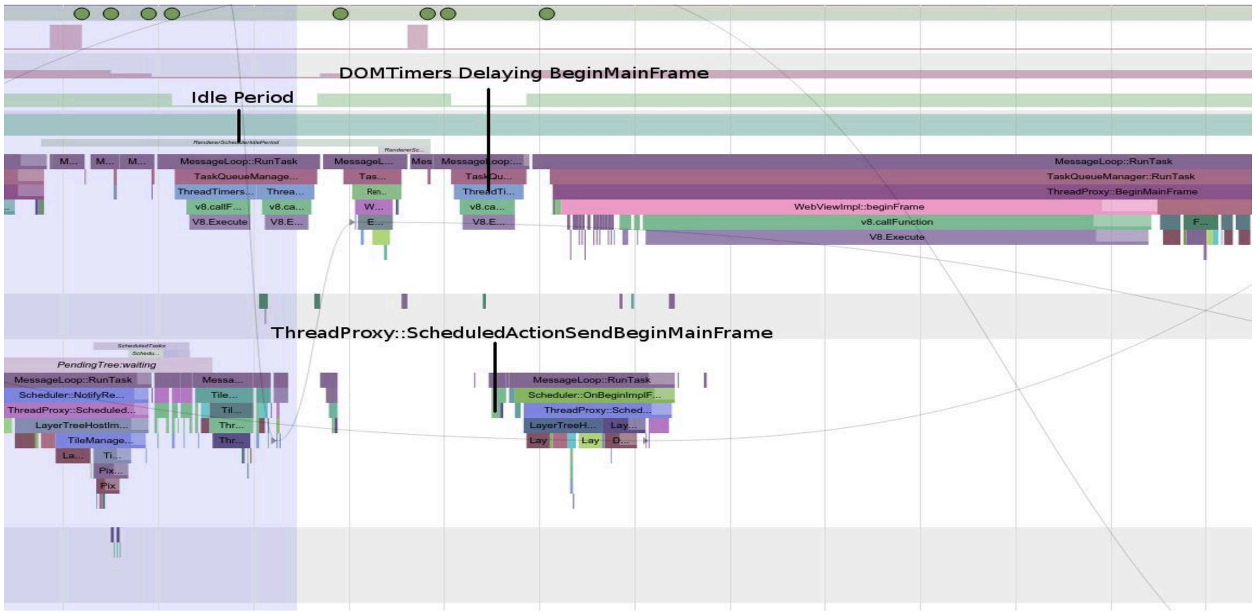
Javascript supports several APIs (setTimeout and setInterval) collectively known as DOMTimers.   While these can block the main thread for an arbitrary length of time, this document is focused on fixing problems cause by tasks running at unfortunate times. The following traces illustrate delays to input handling and BeginMainFrame execution caused by DOMTimers.

1. The user has just started a touch gesture but we execute a DOMTimer shortly before the BeginMainFrame arrives. Since we can't preempt the js execution, processing of the input event in BeginMainFrame is delayed even though the ThreadTimers code yields after running one timer.



Renderer (pid 1658): Los Angeles Times - California, national and world news - Los Angeles Times
DiscardableMemoryUsage:
unused_memory_bytes[0xaf65f7
RendererSchedulerIdlePeriod
Resource
Animation
CrRendererMain
Chrome_ChildIOThread
PendingTree:waiting
ScheduledTasks
InputLatency
InputHandlerProxy::HandleGestu
Animation
Compositor
CompositorTileWorker1/1684
handle-watcher-thread
Dom Timers delaying Begin Main Frame
Touch Start
Touch Move
MessageLoop::RunTask
TaskQueueManager::RunTask
ThreadProxy::BeginMainFrame
WebViewImpl::beginFrame
v8.callFunction
V8.Execute
WebViewImpl::layout
LayerCompositor::updateIfNeededRecursive
CompositingInpu...
Graphics...
RenderWidg...

2. BeginMainFrame can be in the critical path when the there's main thread driven scrolling. Executing DOMTimers shortly before BeginMainFrame is scheduled to occur can delay rendering. (NB the lilac section below shows the vsync)



The blink ThreadTimers poll shouldYieldForHighPriorityWork after each task has been run but by the time that returns true, it's too late. The important task has already been delayed.

The fundamental problem is that Javascript timers are a poor way of sharing the render thread with the browser. Javascript execution can't be preempted and currently javascript has no visibility of browser workloads so its very difficult for javascript to share the thread nicely with the browser.   Better APIs can help, e.g. the requestAnimationFrame callback executes early in the BeginMainFrame task, so it doesn't get in the way of any input tasks.   The proposed requestIdleFrame would execute after the commit, in idle periods (as illustrated above). In addition the requestIdleFrame callback provides a deadline which, if respected, guarantees javascript will have yielded before the next predicted WillBeginFrame.

Design

We will add the ability for the Blink Scheduler to defer DOMTimer execution when a high priority task is anticipated in the near future. This is a potentially risky change, so we will roll it out slowly in a number of phases.

Phase 1: Remove BlinkTimer Heap

Currently Blink Timers execute in one or more monolithic blocks, taking up to 50ms in absence of a high priority task. In addition to DOMTimers, quite a lot of tasks in Blink are timers posted with startOneShot and a delay on zero. As an aside, in the future we would like to refactor all instances of startOneShot into tasks to bring these closer to the chromium style.

Blink Timers have several properties that are slightly unusual from a chromium perspective that we need to preserve:

- They are cancellable
- Their firing interval can be changed in flight (e.g. to align them to fire once every 1s when the tab is in the background)
- They can repeat (in practice most timers are one shot)

We propose to:

1. Refactor handling of delayed tasks in the TaskQueueManager so that it owns delayed tasks posted to it.
    a. On the main thread, this is required because we must ensure pending delayed tasks from blink are deleted before the blink heap goes away during shutdown.
    b. This problem doesn't appear to exist for workers since during shutdown the worker threads go away before the blink heap does.

2. Refactor TimerBase to post cancellable delayed tasks through the scheduler using a new postTimerTask API (or via Platform::currentThread for workers)
    a. Since the timers will no longer execute as a monolithic block, they will interleave better with other work. This may affect loading times (see Risks below).
    b. As a side effect, blink timers posted with a delay of zero will be treated as a non-delayed task and will tend to get executed with less latency than before. NB we are not proposing to remove the 1ms clamp for setTimeout.
    c. The Blink Timer Heap can be deleted as can the SharedTimer support code.
    d. We can't re-use the CancellableTaskFactory because during shutdown in some of the blink unit tests, some delayed tasks posted by Document are left dangling because it's apparently possible for a Document to get deleted without it's destructor running. This is a problem because CancellableTask's destructor does a memory access.
    e. WorkerThreads need a proper solution for IdleTasks. The right way to do this is to add a WorkerScheduler (e.g. see this prototype).
        i. We will introduce a BaseScheduler with support for Normal and Idle tasks. The RenderScheduler will inherit from that, as will the new WorkerScheduler. All the common logic for IdleTasks will be moved to BaseSchedulerImpl which RenderSchedulerImpl and WorkerSchedulerImpl will inherit from (note virtual inheritance is needed to deal with the diamond problem).
        ii. The chromium side of the Workers needs to live in content/child which isn't allowed to include files from content/renderer (the reverse is allowed). This means we need to move quite a lot of Scheduler code either into content/child or possibly base/ for the TaskQueueManager
        iii. The WorkerScheduler will need Long Idle tasks (work for that is done).
3. Add a new TIMER_TASK_QUEUE to the blink scheduler.
    a. On the main thread, TimerBase should post delayed tasks to this new queue.
    b. For worker threads, TimerBase should post delayed tasks to the thread's taskrunner.
4. We would need to add some logic to the Blink Scheduler to support BlinkPlatformImpl::SuspendSharedTimer. Internally the scheduler would support timer suspension via disabling the TIMER_TASK_QUEUE.

Risks:

- The TimerBase refactor may affect loading times since we will probably run loading tasks sooner than before. On some sites this may be a win, on others where DOMTimers create dynamic content this may be a regression. It may also change the number of layouts that occur during page load. We won't know how big this effect is until we try it.
- There's a hard to diagnose UAF bug affecting the existing DOMTimer heap. This refactor may, or may not, help with that.

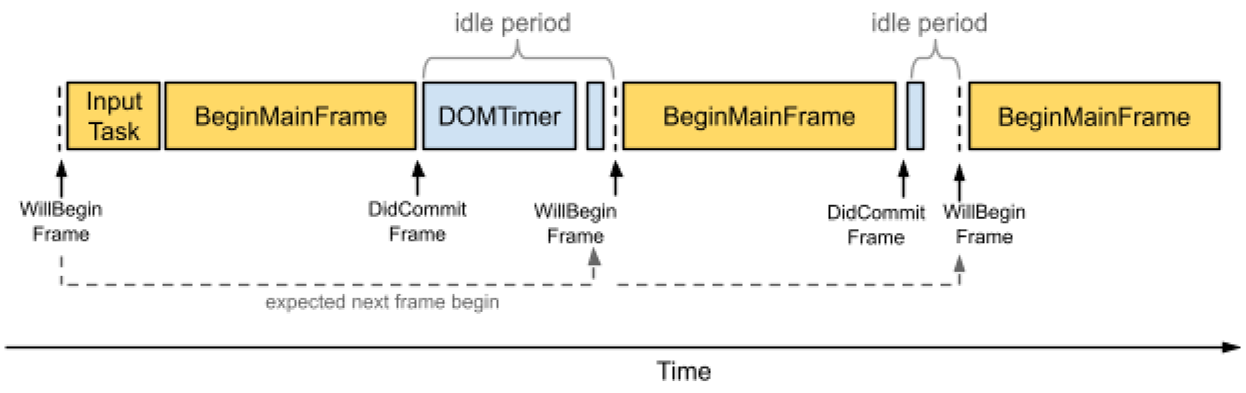
Phase 2: Preventing Touch Start Jank

Once the foundation patches from phase one have stuck, we will simply disable TIMER_TASK_QUEUE when the scheduler is in TOUCHSTART_PRIORITY. This does not affect touch handlers, since these are not executed via timers.

This should be a fairly low risk change that we expect to noticeably improve perceived responsiveness on heavy pages.

Phase 3: Preventing Main Thread Scrolling Jank

When we are using main thread driven scrolling, BeginMainFrame is in the critical path. From the scheduler point of view, this only matters if we are prioritizing smoothness (i.e we are in COMPOSITOR_PRIORITY policy).   To ensure timely execution of BeginMainFrame, we propose to only allow DOMTimer execution during *idle periods*, provided that DOMTimers have not fallen too far behind (see below).



Idle periods are not guaranteed to occur for two reasons:

1. We've blown the budget for this frame and there's no time left
2. We are not generating frames. -- The [long idle times](#) work will address this and is a prerequisite.

Blowing the budget for a single frame, happens now and then (e.g. we had to an expensive layout) and ideally we'd want to try and recover from this situation by not running DOMTimers that frame.

While we want to try and recover from single frame janks, we don't want to totally starve the DOMTimers when in compositor priority. We will add code to the TaskQueueManager to determine how far behind DOMTimer execution is. If it's above a certain threshold (perhaps two frames) then DidCommitFrameToCompositor will unconditionally enable TIMER_TASK_QUEUE and EndIdlePeriod will not disable TIMER_TASK_QUEUE. This will almost certainly result in some jank but correct functioning of the page takes priority over smoothness.

Phase 4: Experiment with limiting DOMTimer execution to idle periods for compositor scrolling too

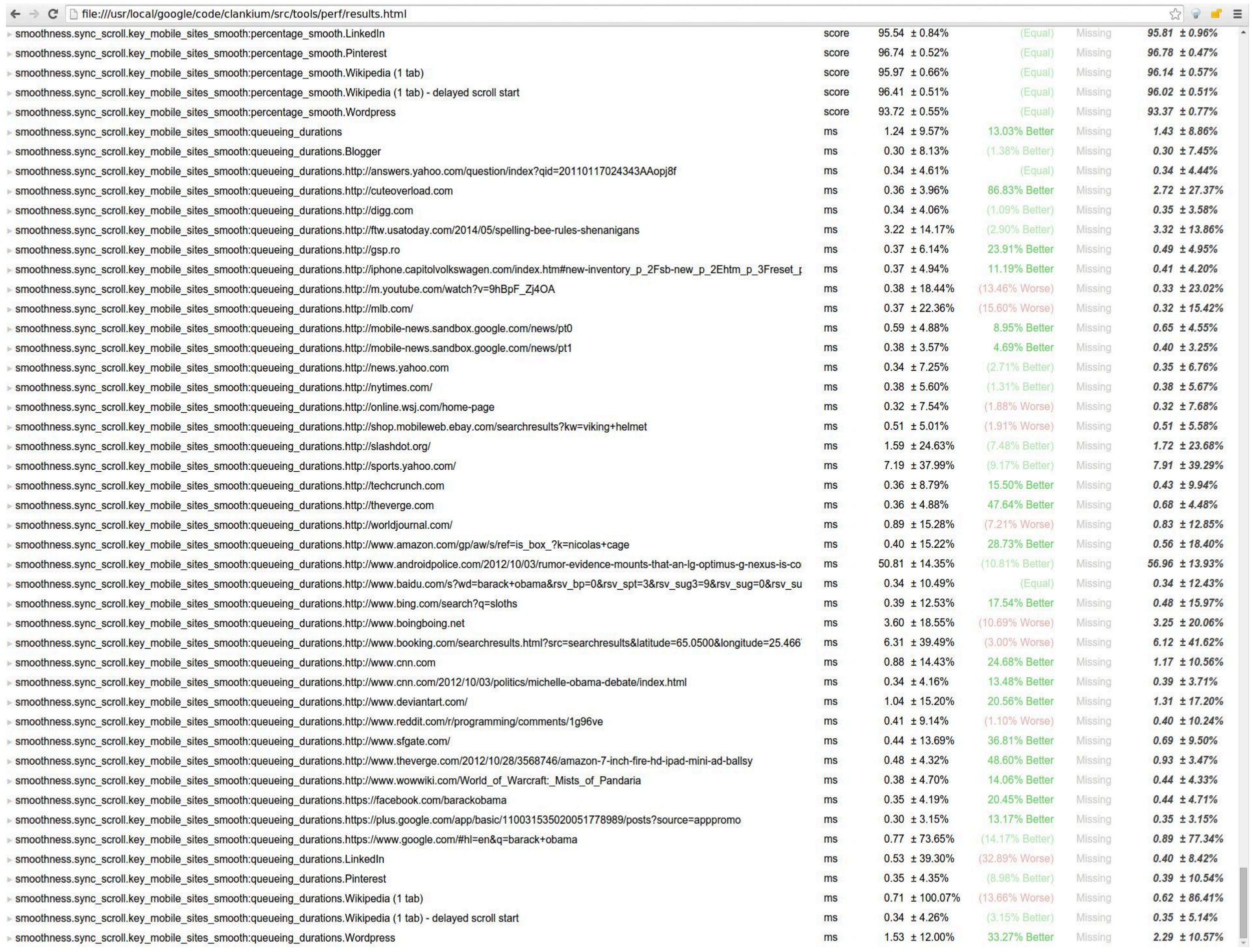
In theory this should reduce scroll hander update latency. Assuming we have a good fallback for a lack of idle time, this ought to be safe.

Phase 5: Simplifying Timer related classes in Blink

Once this has all been done, we have an opportunity to simplify some of the timer logic in blink. The timer alignment code (which is used to round timer fire times up to the nearest second for background tabs) would likely be much simpler if implemented in the scheduler, and we may be able to remove or simplify the DOMTimerCoordinator.

Preliminary Results

Data comparing Baseline with Phase 2 on a Nexus 7v2 with 20 repeats shows an overall 13% reduction in queueing durations for smoothness.sync_scroll.key_mobile_sites_smooth. Bigger reductions are expected for phase 3.



file:///usr/local/google/code/clankium/src/tools/perf/results.html

smoothness.sync_scroll.key_mobile_sites_smooth:percentage_smooth.LinkedIn	score	95.54 ± 0.84%	(Equal)	Missing	*95.81 ± 0.96%*
smoothness.sync_scroll.key_mobile_sites_smooth:percentage_smooth.Pinterest	score	96.74 ± 0.52%	(Equal)	Missing	*96.78 ± 0.47%*
smoothness.sync_scroll.key_mobile_sites_smooth:percentage_smooth.Wikipedia (1 tab)	score	95.97 ± 0.66%	(Equal)	Missing	*96.14 ± 0.57%*
smoothness.sync_scroll.key_mobile_sites_smooth:percentage_smooth.Wikipedia (1 tab) - delayed scroll start	score	96.41 ± 0.51%	(Equal)	Missing	*96.02 ± 0.51%*
smoothness.sync_scroll.key_mobile_sites_smooth:percentage_smooth.Wordpress	score	93.72 ± 0.55%	(Equal)	Missing	*93.37 ± 0.77%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations	ms	1.24 ± 9.57%	13.03% Better	Missing	*1.43 ± 8.86%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.Blogger	ms	0.30 ± 8.13%	(1.38% Better)	Missing	*0.30 ± 7.45%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://answers.yahoo.com/question/index?qid=20110117024343AAopj8f	ms	0.34 ± 4.61%	(Equal)	Missing	*0.34 ± 4.44%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://cuteoverload.com	ms	0.36 ± 3.96%	86.83% Better	Missing	*2.72 ± 27.37%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://digg.com	ms	0.34 ± 4.06%	(1.09% Better)	Missing	*0.35 ± 3.58%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://ftw.usatoday.com/2014/05/spelling-bee-rules-shenanigans	ms	3.22 ± 14.17%	(2.90% Better)	Missing	*3.32 ± 13.86%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://gsp.ro	ms	0.37 ± 6.14%	23.91% Better	Missing	*0.49 ± 4.95%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://iphone.capitolvolkswagen.com/index.htm#new-inventory_p_2Fsb-new_p_2Ehtm_p_3Freset_	ms	0.37 ± 4.94%	11.19% Better	Missing	*0.41 ± 4.20%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://m.youtube.com/watch?v=9hBpF_Zj4OA	ms	0.38 ± 18.44%	(13.46% Worse)	Missing	*0.33 ± 23.02%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://mlb.com/	ms	0.37 ± 22.36%	(15.60% Worse)	Missing	*0.32 ± 15.42%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://mobile-news.sandbox.google.com/news/pt0	ms	0.59 ± 4.88%	8.95% Better	Missing	*0.65 ± 4.55%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://mobile-news.sandbox.google.com/news/pt1	ms	0.38 ± 3.57%	4.69% Better	Missing	*0.40 ± 3.25%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://news.yahoo.com	ms	0.34 ± 7.25%	(2.71% Better)	Missing	*0.35 ± 6.76%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://nytimes.com/	ms	0.38 ± 5.60%	(1.31% Better)	Missing	*0.38 ± 5.67%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://online.wsj.com/home-page	ms	0.32 ± 7.54%	(1.88% Worse)	Missing	*0.32 ± 7.68%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://shop.mobileweb.ebay.com/searchresults?kw=viking+helmet	ms	0.51 ± 5.01%	(1.91% Worse)	Missing	*0.51 ± 5.58%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://slashdot.org/	ms	1.59 ± 24.63%	(7.48% Better)	Missing	*1.72 ± 23.68%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://sports.yahoo.com/	ms	7.19 ± 37.99%	(9.17% Better)	Missing	*7.91 ± 39.29%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://techcrunch.com	ms	0.36 ± 8.79%	15.50% Better	Missing	*0.43 ± 9.94%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://theverge.com	ms	0.36 ± 4.88%	47.64% Better	Missing	*0.68 ± 4.48%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://worldjournal.com/	ms	0.89 ± 15.28%	(7.21% Worse)	Missing	*0.83 ± 12.85%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.amazon.com/gp/aw/s/ref=is_box_?k=nicolas+cage	ms	0.40 ± 15.22%	28.73% Better	Missing	*0.56 ± 18.40%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.androidpolice.com/2012/10/03/rumor-evidence-mounts-that-an-lg-optimus-g-nexus-is-co	ms	50.81 ± 14.35%	(10.81% Better)	Missing	*56.96 ± 13.93%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.baidu.com/s?wd=barack+obama&rsv_bp=0&rsv_spt=3&rsv_sug3=9&rsv_sug=0&rsv_su	ms	0.34 ± 10.49%	(Equal)	Missing	*0.34 ± 12.43%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.bing.com/search?q=sloths	ms	0.39 ± 12.53%	17.54% Better	Missing	*0.48 ± 15.97%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.boingboing.net	ms	3.60 ± 18.55%	(10.69% Worse)	Missing	*3.25 ± 20.06%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.booking.com/searchresults.html?src=searchresults&latitude=65.0500&longitude=25.466	ms	6.31 ± 39.49%	(3.00% Worse)	Missing	*6.12 ± 41.62%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.cnn.com	ms	0.88 ± 14.43%	24.68% Better	Missing	*1.17 ± 10.56%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.cnn.com/2012/10/03/politics/michelle-obama-debate/index.html	ms	0.34 ± 4.16%	13.48% Better	Missing	*0.39 ± 3.71%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.deviantart.com/	ms	1.04 ± 15.20%	20.56% Better	Missing	*1.31 ± 17.20%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.reddit.com/r/programming/comments/1g96ve	ms	0.41 ± 9.14%	(1.10% Worse)	Missing	*0.40 ± 10.24%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.sfgate.com/	ms	0.44 ± 13.69%	36.81% Better	Missing	*0.69 ± 9.50%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.theverge.com/2012/10/28/3568746/amazon-7-inch-fire-hd-ipad-mini-ad-ballsy	ms	0.48 ± 4.32%	48.60% Better	Missing	*0.93 ± 3.47%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.http://www.wowwiki.com/World_of_Warcraft:_Mists_of_Pandaria	ms	0.38 ± 4.70%	14.06% Better	Missing	*0.44 ± 4.33%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.https://facebook.com/barackobama	ms	0.35 ± 4.19%	20.45% Better	Missing	*0.44 ± 4.71%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.https://plus.google.com/app/basic/110031535020051778989/posts?source=apppromo	ms	0.30 ± 3.15%	13.17% Better	Missing	*0.35 ± 3.15%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.https://www.google.com/#hl=en&q=barack+obama	ms	0.77 ± 73.65%	(14.17% Better)	Missing	*0.89 ± 77.34%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.LinkedIn	ms	0.53 ± 39.30%	(32.89% Worse)	Missing	*0.40 ± 8.42%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.Pinterest	ms	0.35 ± 4.35%	(8.98% Better)	Missing	*0.39 ± 10.54%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.Wikipedia (1 tab)	ms	0.71 ± 100.07%	(13.66% Worse)	Missing	*0.62 ± 86.41%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.Wikipedia (1 tab) - delayed scroll start	ms	0.34 ± 4.26%	(3.15% Better)	Missing	*0.35 ± 5.14%*
smoothness.sync_scroll.key_mobile_sites_smooth:queueing_durations.Wordpress	ms	1.53 ± 12.00%	33.27% Better	Missing	*2.29 ± 10.57%*

References

- Blink Scheduler Design Doc
- Blink Idle Task Scheduling
- Cooperative scheduling in Javascript
- Long Idle Tasks
- Tricky UAF bug affecting the Blink TimerHeap