

Side effects due to tree insertion or removal (script, iframe) #808

domfarolino@google.com // dom@chromium.org

[PUBLIC]

What is this?

This document is a log of scrap notes / work that is accumulating as whatwg/dom#808 is tackled. It is related to [DOM Atomic Move Notes & Things](#).

Spec progress

[whatwg/dom#808](#) has been open for several years. dom@ has started replying [[comment](#), [comment](#)] in attempt to answer the following questions:

- Is the spec actually as unclear as OP makes it sound? → seems **answer is "No"**
- Still, Anne & us wish to change the spec to match Chromium/Gecko.
- Does resolving this *actually* help us with the overall goal of designing an atomic move API?
 - Dom thinks the answer is maybe:
 - Even if we clean up all of #808 and get a sane model for insertion/removal steps & timing, we're still going to want to *manually* bypass many of those steps as a part of the atomic move API, since most of the steps that will be run in the new post-#808 model will need to be skipped (i.e., [destroying a child navigable](#) upon "removal")

I found some really old spec work that tries to make DOM/HTML align with Chromium/Gecko: [whatwg/dom#732](#), [whatwg/html#4354](#). We need to figure out what is required to finish it. **Probably figuring out** every case where it's possible to synchronously run script on insertion/removal. For example, when inserting *multiple* elements at the same time — via `append(el1, el2, ...)` — Chromium is consistent about not invoking `'load'` events or running `<script>` *in between* each individual insertion. Are there other cases where Chromium is *not* consistent (and does actually run script in between individual insertions)? We need to know this. [How about 'focus' and 'blur' events, like Mason brings up?](#) Related: <https://github.com/whatwg/dom/issues/808#issuecomment-1920762780>.

New atomic insertion model

The purpose of the table in this section is to enumerate all:

1. **Script-observable** non-script-executing side-effects of insertion
2. **Script-executing** side-effects of insertion

This is important for two reasons:

1. **Spec:** Currently the spec makes no distinction between the two above stages. This table gives us the distinction, so we know which steps to pull out into a special post-insertion steps queue, which runs any script-executing side-effects for each inserted node. This essentially brings the spec into alignment with Chromium's implementation [[cs](#),[cs](#),[cs](#)]
2. **Impl:** So we know which ways Chromium is inconsistent with the desired spec model. There are a few expected deviations, which will result in Chromium bugs or maybe even spec bugs.

Element	Insertion hook	Script-observable side effects during insertion?	Execute script?	Tested?	Browser results
All form-associated elements	DOM insertion steps extension	Yes; form attribute is set	✗	Yes	✓ Chrome ✓ Firefox ✗ Safari
source	HTML element insertion steps	Yes; sets <code><media>.networkState</code>	✗	Yes	✓ Chrome ✓ Firefox ✗ Safari
		Yes (async); triggers img relevant mutation	✗	??	N/A
img	HTML element insertion steps	Yes (async); triggers img relevant mutation	✗	??	N/A
iframe	HTML element insertion steps	Yes; <code>iframe.contentWindow</code> becomes non-null (this & this)	✓	Yes (load event)	✓ Chrome ✗ Firefox (b ; b) 🟡 Safari
option	HTML element	Yes; assigns selectedness ,	✗	??	N/A

	insertion steps	which I think is observable			
details	HTML element insertion steps	Yes; removes the open attribute	✗	??	N/A
frame	HTML element insertion steps	⬆ (same as iframe)	⬆	⬆	⬆
meta (referrer)	inserted into the document	Yes; immediately assigns document's referrer policy	✗	Yes	N/A
meta (theme)	inserted into the document	No; UA suggested color	✗	N/A	N/A
meta (color)	inserted into the document	No; not mandated	✗	N/A	N/A
meta (http-equiv)	inserted into the document	Yes; "default-style" changes styles immediately. (What about "refresh" ?)	✗	Yes	✗ Chrome ✗ Firefox ✗ Safari
object	inserted into the document	Yes; queues a task to run some algo	✗	N/A	N/A
listed form-associated elements	inserted into the document	Yes; arbitrary element insertion can change a form-associated element's `form` attribute	✗	No	N/A
autofocus	inserted into the document	Yes; autofocus candidate could be set next rendering	✗	No	N/A
style	becomes connected	Yes; style is applied immediately, syncly	✗	Yes	(Still figuring this out: c/c/c)
script	becomes connected	Yes; script executes immediately	✓	Yes	Chrome & Firefox generally passes things. But Chrome trips up sometimes. See below.
				Yes	"Outer script doesn't execute before inner

					scripts do" ✗Chrome ✓Firefox ✗Safari
				Yes	Script & iframe side-effects are <i>both</i> post-DOM insertion, in order ✓Chrome ✗Firefox ✓Safari
				Yes	General script and iframe contentWindow insertion order ✓Chrome ✗Firefox ✓Safari
CE reaction	becomes connected	Yes; microtask to invoke connectedCallback	✗	Yes(?)	Unsure; tested elsewhere
link	"dns-prefetch" becomes browsing-context connected	No? IDT dns prefetch is observable	✗	Not by us	N/A
	"manifest" becomes browsing-context connected	No?	✗	Not by us	N/A
	"modulepreload" becomes browsing-context	Yes, the spec fires `load` synchronously if the script is in the module map . This is bad; see this HTML bug .	✓	No?	All browsers do the right thing by ignoring the spec and firing `load` in

	connected				a queued task. HTML bug
	"preconnect" becomes browsing-context connected	No; the <i>processPrefetchResponse</i> algorithm is always called <i>after</i> fetch goes in parallel	✗	Not by us	N/A
	"prefetch" becomes browsing-context connected	⬆ (same as "preconnect")	⬆	⬆	⬆
	"preload" becomes browsing-context connected	⬆	⬆	⬆	⬆
	"stylesheet" becomes browsing-context connected	⬆	⬆	⬆	⬆

Takeaways

Most of the above entries involve DOM insertions whose side effects are script-observable, but do not themselves execute script. This means most things are WAI and can be left alone. The exceptions are:

- **iframes:** [Spec]: We want to move the entire insertion steps to the post-insertion queue. That is, we want to push *child document creation & load event firing* to the post-insertion queue, to only take effect after all DOM mutations
- **script:** [Impl]: Chromium breaks script ordering. See [this test](#) and [this part of the spec](#) that requires this behavior. How hard would this be to fix?
- **meta ?**: [All]: We need to decide whether we want meta "default-style" to apply synchronously during DOM insertion or not. No browser passes [this WPT](#) at all, which is concerning. [Chrome only processes HTTP-EQUIV in the post-insertion](#)

[queue](#). Maybe we can modify the spec to push the application of these steps to the post-insertion queue, which should match all browsers??

- **link:** Optional follow-up: <https://github.com/whatwg/html/issues/10166>.
- **Video:** [all is well!](#)

PRs:

- <https://github.com/whatwg/dom/pull/1261>
- <https://github.com/whatwg/html/pull/10188>
- TODO [DOM]: Add an argument for "is insertion" to "child changed steps"
- TODO [HTML]: Use the argument

New atomic removal model

See [this GitHub Gist](#).

Element	removal hook	Script-observable side effects during insertion?	Execute script?	Tested?	Browser results
focusable elements			✗	✓	✗ Chrome ✓ Firefox ✓ Safari See gist
iframe	HTML element removing steps	Yes, same-origin documents are disposed (`pagehide`)	✓	Yes, yes	✗ Chrome ✗ Firefox ✗ Safari See gist
frame	HTML element removing steps				
select	HTML element removing step				
dialog	HTML element removing steps				
meta	"theme-color"				

	removed from the document				
	all removed from the document				
object	removed from the document				
media	removed from the document				
form-associated elements	removed from the document				
style	becomes disconnected				
custom elements	becomes disconnected				
render-blocking element	becomes disconnected				

TODO(dom): Fill this out (with [this gist too](#)).

Takeaways

- <https://github.com/whatwg/html/issues/3847>
- <https://github.com/whatwg/html/pull/8392> → <https://crbug.com/40888574> → <https://crbug.com/41484175>
-

WPTs

- crrev.com/c/5251828 → [wpt#44308](https://wpt.fyi/#44308)
- Many more WPTs that Anne just made me aware of: [wpt#15264](https://wpt.fyi/#15264); he's encouraged us to repurpose them (i.e., take over the PR I guess)
- Add more focus/blur tests for other applicable elements

Implementation notes

- ☑ ~~TODO(dom): Start getting a grip on how the Chromium implementation achieves the "deferred steps queue" that Anne wants, and see if it can be used as inspiration to finish the lingering spec work. See:~~

- ☑ <https://github.com/whatwg/dom/pull/732>
- ☑ <https://github.com/whatwg/html/pull/4354>

dom@ thinks the desired queue of "post-insertion steps" that we'll maintain in the spec is likely related to the [InsertionNotificationRequest](#) DOM enum, which is used to allow an element to run script synchronously after DOM insertion, by having its `InsertedInto()` return `kInsertionShouldCallDidNotifySubtreeInsertions`.

An interesting example was raised which shows that in Chromium, all script side-effects for insertion happen after **all** node insertions are complete, for any given `append()` call or `DocumentFragment` insertion. This is an important quality for insertion atomicity, and now we'll explore how Blink DOM's implementation achieves this, so we can model the spec off of it.


`ContainerNode::AppendChild()`

`ContainerNode::InsertNodeVector()`

`ContainerNode::NotifyNodeInsertedInternal(..., *post_insertion_notification_targets)`

`ContainerNode::DidInsertNodeVector()`

Drafts

 [Drafts] Side-effects #808