bfcache: Test API for eviction

hajimehoshi@chromium.org
Last Updated: 2019-07-16
Status: Public

CL for this proposal

tl;dr

We propose these action items:

- Add DOMMessageQueue::RenderFrameDeleted to make EvalJsOptions return when the frame is gone
- Guarantee that EvalJs (and JavaScriptExecuteRequestForTests) is called on a non-freezable task runner
- Add world_id parameter to JavaScriptExecuteRequestForTests and avoid using JavaScriptExecuteRequestInIsolatedWorld for tests

By these items, we can test BFcache eviction correctly.

Background

Back-forward cache (a.k.a BFcache) aims for faster back-forward navigation by freezing the old page and reusing it when the user goes back to the page.

A BFcached page is supposed not to mutate its state due to privacy concerns. For example, JavaScript must not be executed on the BFcached frozen page. In most cases, JavaScript is executed on per-frame task runners, and we can freeze these task runners. However, there are still a few cases when JavaScript is executed on per-thread task runners. As per-thread task runners are used by multiple pages, we cannot freeze these task runners for one frozen page. Then we need to 'evict' the page when JavaScript execution is detected.

To test the eviction, we need an API to execute JavaScript with some conditions. For example, JavaScript must be executed on a non-freezable task runner in the test, or JavaScript would never be executed. We would also like to specify isolated world IDs for extension tests. We investigated the current APIs to execute JavaScript and these attributes.

Current APIs

	Wait for results?	Can specify an isolated world?	Is the task runner frozen on BFcache?	Who is called on renderer side?
content:: ExecJS	Yes	Yes	either	RenderFrameImpl:: JavaScriptExecuteRequestForTests / JavaScriptExecuteRequestInIsolatedWorld
content:: ExecuteScript	Yes	No	either	(same as above)
content:: ExecuteScriptWithoutUserGesture	Yes	No	either	(same as above)
content:: ExecuteScriptAsync	No	No	either	RenderFrameImpl:: JavaScriptExecuteRequestForTests
RenderFrameHost:: ExecuteJavaScript	No (callback)	No	no?	RenderFrameImpl:: JavaScriptExecuteRequest
RenderFrameHost:: ExecuteJavaScriptInIsolatedWorld	No (callback)	Yes	yes?	RenderFrameImpl:: JavaScriptExecuteRequestInIsolatedWorld
RenderFrameHost:: ExecuteJavaScriptForTests	No (callback)	No	no?	RenderFrameImpl:: JavaScriptExecuteRequestForTests
RenderFrameHost:: ExecuteJavaScriptWithUserGestureForTests	No	No	no?	RenderFrameImpl:: JavaScriptExecuteRequestForTests

For task runners, there is no explicit guarantee. This is the current implementations and they can be changed in the future. It looks like JavaScriptExecuteRequestInIsolatedWorld uses PausableScriptExecutor using a per-frame freezable task runner, while the other functions don't (kInternalIPC, which is not frozen on BFCache).

In the context of BFcache eviction, these properties are required:

- **Want to wait for execution result?**: No. After eviction, the frame no longer exists. If the API waits for the result of JavaScript execution, this would never return.

- **Want to specify an isolated world?**: Yes. Eviction should work on all the isolated worlds (e.g., for extensions), and we want to test it.
- **Want to specify a task runner?:** Yes. The task for JavaScript execution must be on non-freezable task runner for tests. We also want an explicit guarantee of task runners.

Unfortunately, there is no existing API that satisfied all the above requirements.

Proposal

We propose these action items:

- **Add DOMMessageQueue::RenderFrameDeleted to make EvalJsOptions return when the frame is gone**: Now EvalJs blocks forever when the frame is gone. We can fix this by observing RenderFrameDeleted at DOMMessageQueue and abort the run loop inside.
- **Guarantee that JavaScriptExecuteRequestForTests is called on a non-freezable task runner:** JavaScriptExecuteRequestForTests is now called on a non-freezable task runner, and we can guarantee (kInternalIPC). Let's add comments.
- **Add world_id parameter to JavaScriptExecuteRequestForTests and avoid JavaScriptExecuteRequestInIsolatedWorld for tests:** We don't have to use JavaScriptExecuteRequestInIsolatedWorld that uses a per-frame (freezable) task runner. Let's replace the usage in EvalJs.

These should be safe in terms of backward compatibility.