Cleaning up LifecycleObservers

haraken@chromium.org
2016 May 6
STATUS: PUBLIC

Motivation

Currently Blink has a number of lifecycle-related observers:

- DocumentLifecycleObserver
- DocumentVisibilityObserver
- DOMWindowLifecycleObserver
- LocalFrameLifecycleObserver
- PageLifecycleObserver
- ContextLifecycleObserver
- DOMWindowProperty
- ActiveDOMObject

Some of these observers do similar things. Some of these observers do unexpected things (than things you expect from the method names). The behaviors are not documented. Consequently, the observers are wrongly used in many places in the Blink code base. **We should clean up the observers and add a good documentation.**

This document lists up existing issues around the observers and proposes some clean-up. Once the clean-up is done, the world will be simplified with the following observers:

- **ContextLifecycleObserver**
  - Use this when you want to observe ExecutionContext's lifecycle. In common cases you should just use this.
- **DOMWindowLifecycleObserver**
  - Use this when you want to observe DOMWindow's lifecycle.
- **PageVisibilityObserver**
  - Use this when you want to observe page visibility changes.
- **SuspendableObject**
  - Use this when you want to define behavior when the page is suspended or resumed.

Action items

Remove DocumentLifecycleObserver

DocumentLifecycleObserver should be replaced with ContextLifecycleObserver because they do similar things. dcheng@ already fixed it.

Remove DocumentVisibilityObserver

DocumentVisibilityObserver should be replaced with PageLifecycleObserver because they do similar things. dcheng@ already fixed it.

Rename PageLifecycleObserver to PageVisibilityObserver

Make it clear that PageVisibilityObserver is a thing to observe page visibility changes. Also we should remove PageVisibilityObserver::didCommitLoad. The method is used by NavigatorVibration and ScreenWakeLock but I think they can be replaced with ContextLifecycleObserver::contextDestroyed. As a result, PageVisibilityObserver should provide only PageVisibilityObserver::pageVisibilityChanged.

Remove LocalFrameLifecycleObserver

Many call sites are assuming that LocalFrameLifecycleObserver::willDetachFrameHost is called when the page navigates, but the assumption is wrong. The LocalFrame is reused across navigations, so LocalFrameLifecycleObserver::willDetachFrameHost is not called when you expect it being called. Alternately, you should use ContextLifecycleObserver::contextDestroyed. I think all LocalFrameLifecycleObservers can be replaced with ContextLifecycleObservers.

Question: Is there any LocalFrameLifecycleObserver that cannot be replaced with ContextLifecycleObserver?

Remove DOMWindowLifecycleObserver

DOMWindowLifecycleObserver is used to hook addEventListeners/removeEventListeners on a window object. However, if you want to hook addEventListeners/removeEventListeners on some object, you should instead override EventTarget::addEventListenerInternal/removeEventListenerInternal. EventTarget::addEventListenerInternal/removeEventListenerInternal is a general way to observe event listener attachments/detachments on any EventTarget object whereas DOMWindowLifecycleObserver is a specific way to do that on only window objects. We should

remove DOMWindowLifecycleObserver. (At the very least, it should be renamed to WindowEventListenersObserver.)

Replace DOMWindowProperty with LocalDOMWindowLifecycleObserver

DOMWindowProperty is mainly used to observe when LocalDOMWindow::reset is called; i.e., it is used to observe the lifecycle of LocalDOMWindow. We should introduce LocalDOMWindowLifecycleObserver and replace DOMWindowProperty with it.

Some classes that currently inherit from DOMWindowProperty are overriding DOMWindowProperty::willDestroyGlobalObjectInFrame and DOMWindowProperty::willDetachGlobalObjectFromFrame, but I think we can just replace them with ContextLifecycleObserver::contextDestroyed.

ContextLifecycleObserver::contextDestroyed must always be called

We're assuming that ContextLifecycleObserver::contextDestroyed is always called, but it's not true. It is called for a normal Document attached to a frame, but it's not called for Documents created by DOMImplementation::createDocument, temporary Documents used by XSLT etc. As a result, contextDestroyed may not be called. This is problematic because contextDestroyed sometimes does important clean-up.

To address the issue, we need to guarantee that Document::detach is called for all Documents. I'm not sure how easy it is to find an explicit point where we can call Document::detach for Documents created by DOMImplementation::createDocument, but we need to fix it.

ContextLifecycleObserver::m_lifecycleContext should be cleared when contextDestroyed() is called

Currently ContextLifecycleObserver::m_lifecycleContext is not cleared when contextDestroyed is called. It is not cleared until the next GC is triggered. This means that ContextLifecycleObservers can access the lifecycle context even after the context is destroyed (and before the context is GCed). Some call sites are using lifecycleContext() as a way to check whether contextDestroyed has already been called or not, but the check is not working as expected due to the bug.

We should clear the lifecycle context promptly when contextDestroyed() is called, because the context should not be valid after it's destroyed (i.e., after Document::detach is called). However,

when I tried to make the change, it broke a bunch of tests since some call sites are assuming the existing (wrong) behavior. We need to fix it.

Make most ActiveDOMObjects ContextLifecycleObservers

ActiveDOMObjects are objects that implement a special behavior when the page is suspended or resumed (e.g., when DevTools enters/leaves a break point, when a user clicks a "print" button). If you don't need a special behavior for the suspend/resume, you should just use ContextLifecycleObserver. I think that most ActiveDOMObjects should be converted to ContextLifecycleObservers.

Rename ActiveDOMObject to SuspendableObject

To make it clear that it is a thing to define a special behavior for the suspend/resume, rename it to SuspendableObject.

Question: Currently some objects implement ActiveDOMObject::suspend/resume just to suspend/resume timers (example). This is not nice because timers are everywhere and it's weird to implement ActiveDOMObject::suspend/resume for all the timers. In fact, most existing timers are not correctly handled. Instead of asking each object that has timers to override ActiveDOMObject::suspend/resume, would it be possible to forcibly suspend/resume all timers in Timer.h? Or would it be problematic because it will also stop necessary timers that must be executed while DevTools are in a break point? (I don't know.)

Add LifecycleObservers.md

We should add a good documentation :)