Android Webview Compositing Cleanup

Tracking Bug: crbug.com/439275

Problem

The scheduling code for supporting webview is complicated. This is mostly because of the mismatch between the compositor's push model vs webview's pull model.

Background

Since webview is just another view in the Android view hierarchy it has to satisfy certain constraints. The biggest difference between webview and other chromium platforms is that webview doesn't own the underlying surface that it draws to. Webview has to draw when Android asks it to. There are two methods in the View API that are of particular interest: View.onDraw and View.invalidate. The onDraw method is sent to webview and that means that webview should draw into a surface provided by the system. The surface may be backed by a software bitmap or it could be a hardware surface. The invalidate method is used by webview to tell Android that it has new content and should be redrawn.

Apart from that these constraints also apply:

1. After calling invalidate, onDraw might not arrive.
2. Webview might receive onDraw calls even if it didn't call invalidate e.g. a view on top of webview going away might cause webview to get an onDraw.
3. The app containing webview can go into background and webview must conserve power if it's in hardware mode.

Current Implementation

This is also discussed in this doc.

When the Scheduler decides it has a new frame to draw:

1. The Scheduler asks SynchronousCompositorExternalBeginFrameSource for BeginFrames.
2. BrowserViewRenderer calls View.invalidate and sets up a 100ms repeating timer to do a fake draw and call View.invalidate again if the onDraw method doesn't get called in that time.

When webview receives onDraw:

1. In SynchronousCompositorOutputSurface we set certain properties such as viewport size, scroll offset, etc. on LayerTreeHostImpl.
2. SynchronousCompositorExternalBFS sends a BeginFrame message to the Scheduler.
3. The Scheduler performs impl-thread animations and draw and initiates any asynchronous operations such as rAF, PrepareTiles, etc. but does not wait for them. It does this by triggering its deadline synchronously while handling the BeginFrame.

4. At the end of the BeginFrame the Scheduler stops asking SynchronousCompositorExternalBFS for BeginFrames.
5. The Scheduler sets up a repeating timer to drive it's state machine forward while it's not receiving BeginFrames.

Proposed Implementation

The Scheduler should behave like it does on platforms other than webview i.e. it should keep asking for BeginFrames as long as it has work to do.

When the Scheduler decides it has a new frame to draw (or an action that can only be performed with a draw):

1. It calls Invalidate on the OutputSurface.
2. SynchronousCompositorOutputSurface calls View.invalidate.

When webview receives onDraw:

1. In SynchronousCompositorOutputSurface we set certain properties such as viewport size, scroll offset, etc. on LayerTreeHostImpl.
2. SynchronousCompositorOutputSurface sends an OnDraw message to the Scheduler.
3. The Scheduler performs any actions associated with the "draw" phase of a frame. Certain async operations such as PrepareTiles might be initiated but no actions associated with the "animate" phase are performed.

When webview receives a vsync signal:

1. SynchronousCompositorExternalBFS send a BeginFrame message to the Scheduler.
2. The Scheduler performs any impl-thread animations and initiates any asynchronous operations associated with the "animate" phase such as rAF. The Scheduler does not trigger any deadline or perform any actions associated with the "draw" phase.

New API had to be added to OutputSurface and OutputSurfaceClient (along with necessary plumbing to/from the Scheduler):

```
// Support for a pull-model where draws are requested by the output surface.
// OutputSurface::Invalidate is called by the compositor to notify that
// there's new content.
virtual void Invalidate() {}


// This allows the output surface to ask it's client for a draw.
virtual void OnDraw() = 0;
```