# Root Layer Scrolling

`bit.ly/root-layer-scrolling`

*author: Steve Kobes <skobes@chromium.org>*

*last updated: 2017-02-07*

*this document is public*

## Summary

Root layer scrolling is a Blink project to make the root PaintLayer scrollable.  This relieves FrameView of its scrolling responsibilities, unifying frame scrolling with overflow scrolling.

## Background

(Also see bit.ly/blink-scrolling for a general overview of scrolling in Blink.)

### Scrollers

Blink has the concept of a **scrollable area** (or "scroller"), which represents a rectangular region that can be independently scrolled. Scrollers implement the ScrollableArea interface and have a number of responsibilities such as:

- being the source of truth for the scroller's current **scroll offset**
- exposing APIs to trigger scroll operations
- invoking **scroll animations**
- tracking the size of the scrolling **content**
- tracking the size of the scroller's **visible rect**
- tracking the **scroll origin** (initial offset in RTL modes)
- deciding when various kinds of scroll operations are possible
- deciding when scrolling should be accelerated by compositing
- deciding when **scrollbars** should exist
- creating, owning, and destroying Scrollbar objects
- computing the layout (position and size) of scrollbars
- computing the bounds (min and max offset) of the scroller
- performing the side effects of adding or removing scrollbars (such as relayout of the scroller's content)
- performing the side effects of changing the scroll offset, such as
    - repositioning composited layers
    - Issuing paint invalidations
    - queuing "scroll" DOM events
    - making overlay scrollbars visible

### Frames and DOM Nodes

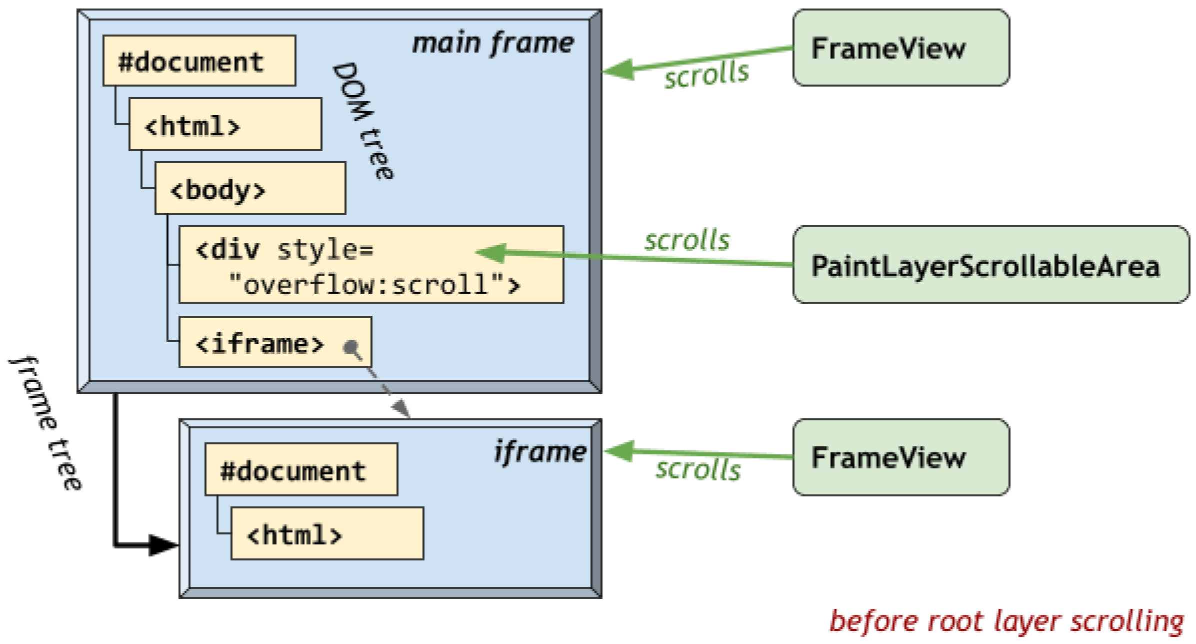
Web content lives in a tree of Frame objects. The root of the frame tree is called the **main frame**. (In a page with no `<iframe>` or `<frameset>` elements, this is the only frame that exists.)

Each frame contains its own DOM tree of Node objects. The root of a DOM tree is a Document.

In general, all frames are scrollable, and some DOM nodes are also scrollable. For example, setting "`overflow: scroll`" on a `<div>` element makes it scrollable. Scrollable

DOM nodes are also called **overflow scrollers**.

Currently, the scroller for a frame is a FrameView, and the scroller for a scrollable DOM node is a PaintLayerScrollableArea (henceforth "**PLSA**").



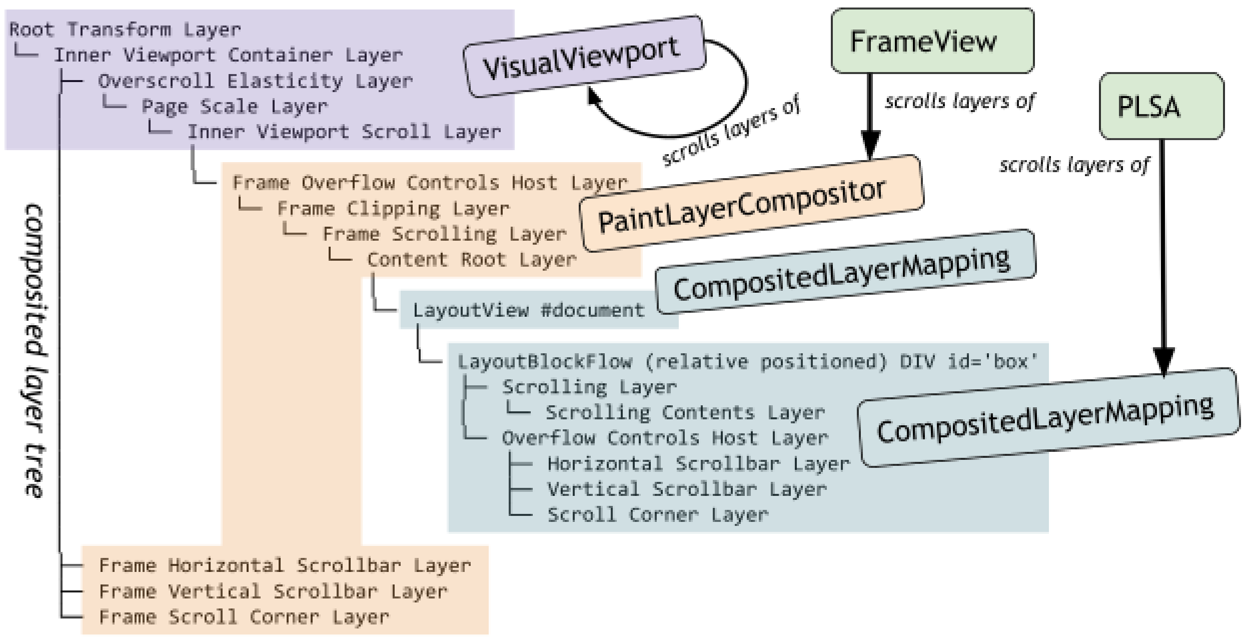
FrameView and PLSA are the only ScrollableArea implementations used in layout, and there is significant code redundancy between them.  Often, scrolling features and bug fixes must be implemented twice—in FrameView for frames, and in PLSA for overflow scrollers.

## Compositing

If a frame or a DOM node is composited, we must create GraphicsLayer objects for the border, the scrolling content, the scrollbars, and the scroll corner.

The composited layers for a scrollable frame are created and managed by PaintLayerCompositor.  The composited layers for a scrollable DOM node are created and managed by CompositedLayerMapping.

Currently, the GraphicsLayer tree for a page with a scrolling `<div>` looks like this:


Root Transform Layer
└ Inner Viewport Container Layer
├ Overscroll Elasticity Layer
└ Page Scale Layer
└ Inner Viewport Scroll Layer
└ Frame Overflow Controls Host Layer
└ Frame Clipping Layer
└ Frame Scrolling Layer
└ Content Root Layer
└ LayoutView #document
└ LayoutBlockFlow (relative positioned) DIV id='box'
├ Scrolling Layer
└ Scrolling Contents Layer
└ Overflow Controls Host Layer
├ Horizontal Scrollbar Layer
├ Vertical Scrollbar Layer
└ Scroll Corner Layer
├ Frame Horizontal Scrollbar Layer
├ Frame Vertical Scrollbar Layer
└ Frame Scroll Corner Layer
composited layer tree
VisualViewport
scrolls layers of
FrameView
scrolls layers of
PLSA
scrolls layers of
PaintLayerCompositor
CompositedLayerMapping
CompositedLayerMapping

There is redundancy between PaintLayerCompositor and CompositedLayerMapping, mirroring the redundancy between FrameView and PLSA.

## Design

### What is root layer scrolling?

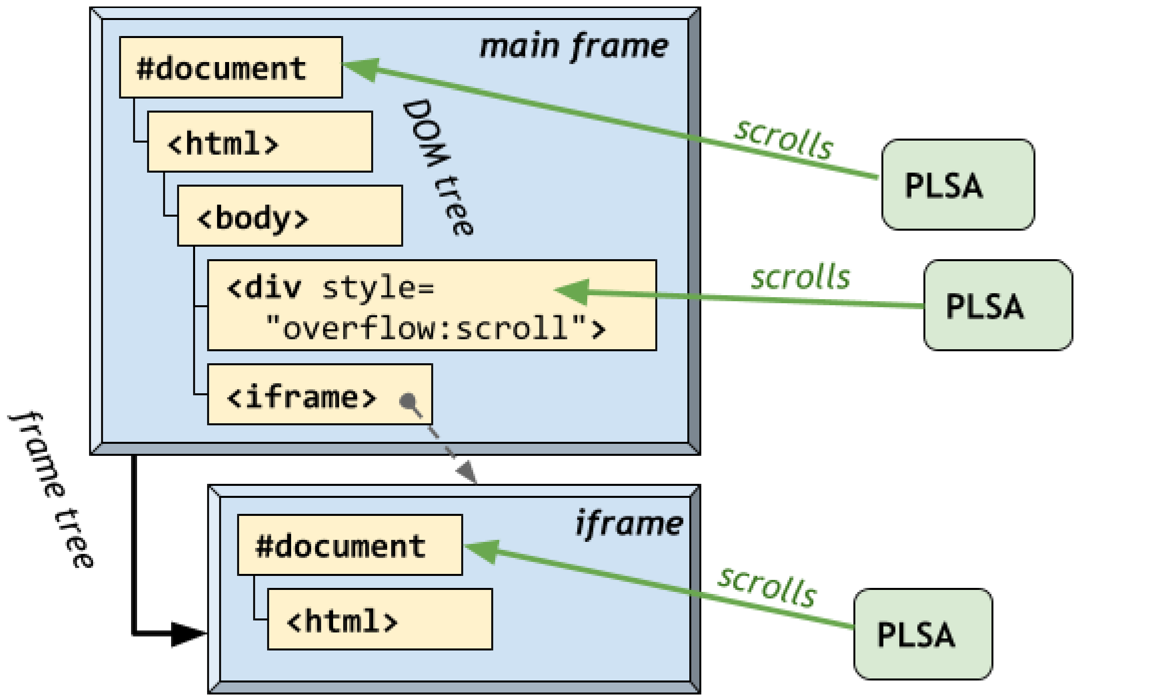
In a sentence: ***Root layer scrolling makes the document scrollable.*** Recall that the document is a node in the DOM tree. Making it scrollable has the following implications:

- The document will clip its own overflow, and scroll it with its own PLSA.

  (Every LayoutBox already constructs a PLSA, but it goes unused for boxes that don't clip their overflow - see crbug.com/467721.)

- If the document is composited, its CompositedLayerMapping will create GraphicsLayers for composited scrolling. (The main frame's document is always composited.)

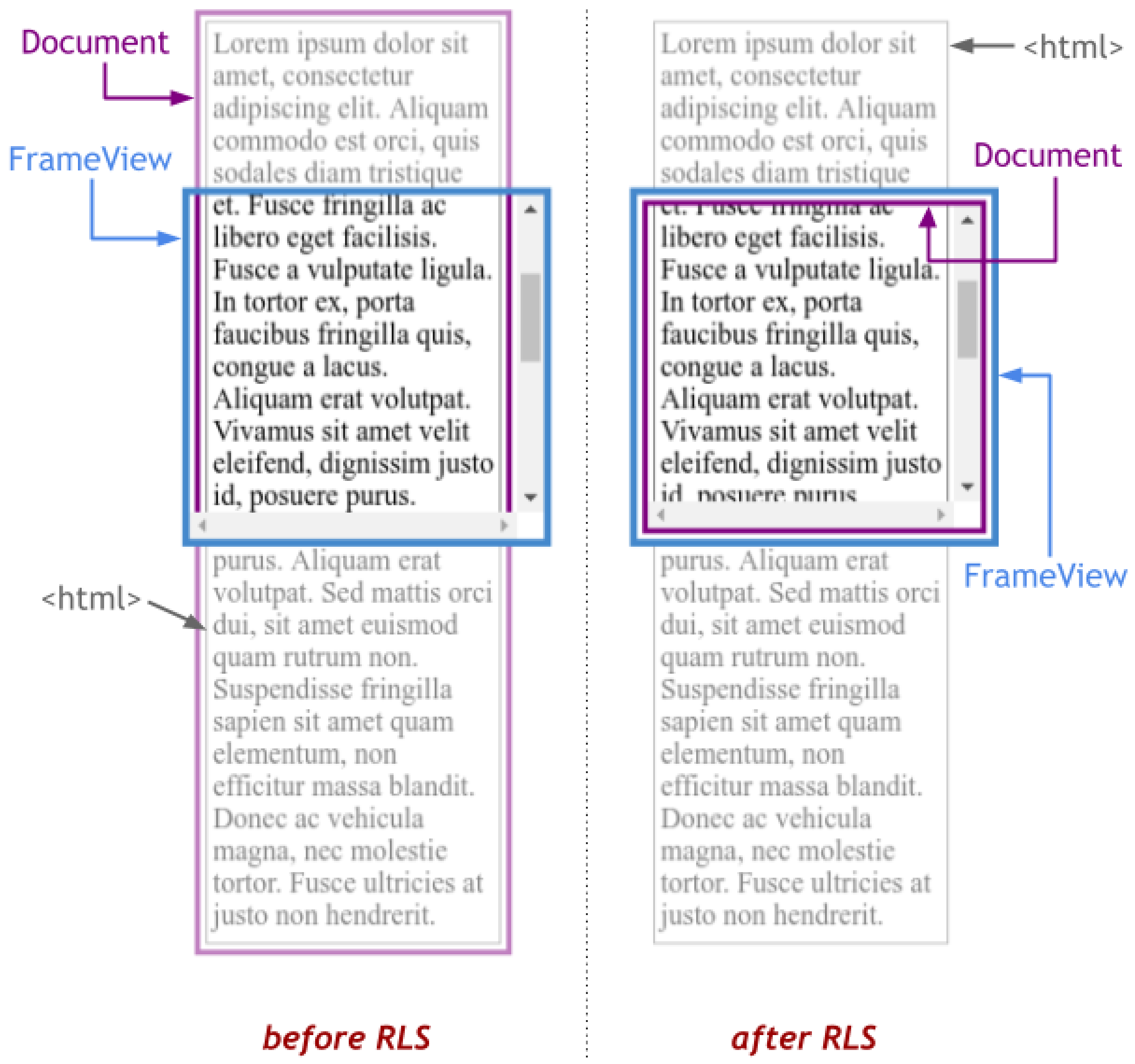
In other words, the document now summons all of Blink's existing machinery for scrollable DOM nodes.



*after root layer scrolling*

### What happens to FrameView?

Recall that the document lives inside a frame, and logically belongs to the "scrolling content" of the FrameView. However, scrollers clip overflow, so a document that is a scroller will contribute no overflow to the frame that contains it.



Thus, when root layer scrolling is enabled, the FrameView cannot scroll. (We have added an ASSERT in [FrameView::updateScrollOffset](FrameView::updateScrollOffset) to enforce this.)

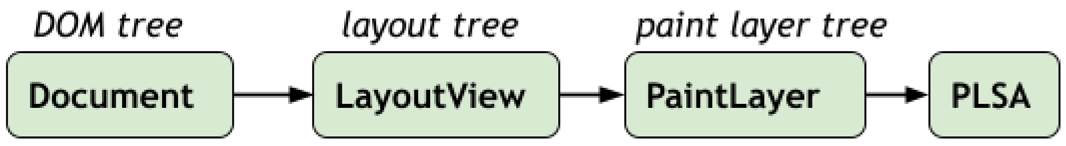
Because it cannot scroll, all of the scrolling code in FrameView, and all of the scrolling layer management code in PaintLayerCompositor, becomes dead with root layer scrolling. After launch:

- FrameView's scrolling code can be removed.
- **FrameView will no longer inherit from ScrollableArea.**
- The GraphicsLayers owned by PaintLayerCompositor will go away. The document's CompositedLayerMapping will be connected directly to the visual viewport layers.

(FrameView will continue to exist, but only for responsibilities unrelated to scrolling.)

### Aside: PaintLayer

The preceding discussion elides some indirection between the document and the PLSA. Ownership of the PLSA is actually through the document's counterparts in the layout tree and the PaintLayer tree:



The name of the feature, "root layer scrolling", refers to the root PaintLayer, which owns the document's PLSA.

However, in the long term we would like to eliminate PaintLayer by refactoring it into more specific components.  In that world, PLSA might be owned directly by the layout tree (and renamed).

### Viewports

To support pinch zoom, VisualViewport injects itself as another scrollable area above the main frame in the GraphicsLayer tree (see diagram under Compositing, and Viewports and Coordinates for general background on the dual viewport model).

Root layer scrolling does not affect the visual viewport, but it changes the identity of the **layout viewport**.  The layout viewport is currently the main frame's FrameView, but after root layer scrolling, the layout viewport will be the PLSA of the main frame's document.

Two root layer scrolling–aware convenience methods are provided on FrameView:

- FrameView::layoutViewportScrollableArea, when called on the main frame, will return the layout viewport.

- FrameView::getScrollableArea, when called on the main frame, will return the RootFrameViewport, which proxies scroll operations into the visual and layout viewports.

When called on an iframe, the two methods are identical, and return the appropriate scrollable area for the iframe depending on whether root layer scrolling is enabled.

> **NOTE:**  The rules for viewport sizing create a complication for root layer scrolling. The layout viewport is sized to accommodate the minimum-scale view, which may

be larger than the CSS initial containing block.  This means (unlike other overflow scrollers) the document's overflow clip rect is different from its box bounds for CSS positioning.  The method override LayoutView::overflowClipRect reflects this wrinkle. (See crbug.com/492871 for more details.)

## Benefits

Root layer scrolling brings the following benefits:

- Eliminating redundancy between FrameView and PLSA, which independently implement the responsibilities of a scroller.

- Eliminating redundancy between PaintLayerCompositor and CompositedLayerMapping, which independently manage GraphicsLayers for composited scrolling.

- Making the Blink scrolling architecture cleaner and simpler to understand.

- Bridging the gap in quality and functionality between frame scrolling and overflow scrolling.  Root layer scrolling has uncovered bugs and limitations of overflow scrolling, particularly around composited overflow scrollers.

Examples of overflow-scrolling bugs and limitations uncovered by root layer scrolling include:

- crbug.com/443283 (compositing)
- crbug.com/464612 (incorrect paint invalidation)
- crbug.com/499083 (hit testing abs-pos content)
- crbug.com/535051 (coordinated scrollbars)

## Philosophy

### Behavior NOP

Root layer scrolling aims to be a pure refactoring.  That is, there should be no change in the observable behavior of the browser or the web platform, and no regressions in performance or stability.

### Frames aren't special

When root layer scrolling uncovers a difference between frame scrolling and overflow scrolling, we should ask if there is a good reason for the divergence, or if it is a bug or a historical accident. For example:

- In an overflow scroller, left vs. right hand scrollbar placement depends on the scroller's content (RTL vs. LTR). But in crbug.com/249860 it is argued that the main frame's scrollbar placement should depend only on the browser UI language. This is a legitimate reason to "special-case" the scrolling of the root layer.

- Coordinated scrollbars (crbug.com/535051) were enabled only for the main frame. But this was just because no one bothered to make them work on overflow scrollers. It is better to give overflow scrollers coordinated scrollbars, than to blindly copy the existing behavior for root layer scrolling.

### Don't cross the streams

We should be careful to preserve the identities of FrameView and the root layer's PLSA as two distinct scrollers (the latter nested in the former) during the transition. It may be tempting to make FrameView methods delegate to PLSA or vice versa. This would be bad—creating confusion, inconsistency, and compromising our guarantees of post-launch code removal in FrameView.

(This is less important for FrameView code that is not part of its ScrollableArea implementation, such as the convenience methods described in Viewports.)

## Related Work

Root layer scrolling is of significant interest to the Slimming Paint project, which changes the way compositing is done in Blink.

Root layer scrolling has significant overlap with Non-Document Root Scrollers, which is a proposed web platform feature to make arbitrary scrollable elements act as the layout viewport (triggering behaviors such as url-bar hiding and pull-to-refresh).

## Testing

The bulk of the implementation work for root layer scrolling lies in tracing regressions to discover and fix code that is coupled to the old architecture or making assumptions that are no longer valid with root layer scrolling.

Therefore, we are making some investments in infrastructure to track regressions in unit tests and layout tests that are correlated with root layer scrolling.

For example, we have enhanced the layout test runner to support flag-specific expectations and baselines with `--additional-driver-flag`, and we plan to spin up a bot that continuously runs the entire suite of tests with `--root-layer-scrolling` on trunk.

This scales better than forcing the main waterfall bots to test root layer scrolling through virtual test suites.

## Status

The implementation of root layer scrolling is tracked in crbug.com/417782 and its dependent bugs.  It is currently behind a Blink REF (`RootLayerScrolling`) connected to a command-line flag `--root-layer-scrolls`.

The layout team has a Q1 2017 OKR to get all unit tests passing with root layer scrolling (crbug.com/490942).