an overview of

SCROLLING IN BLINK

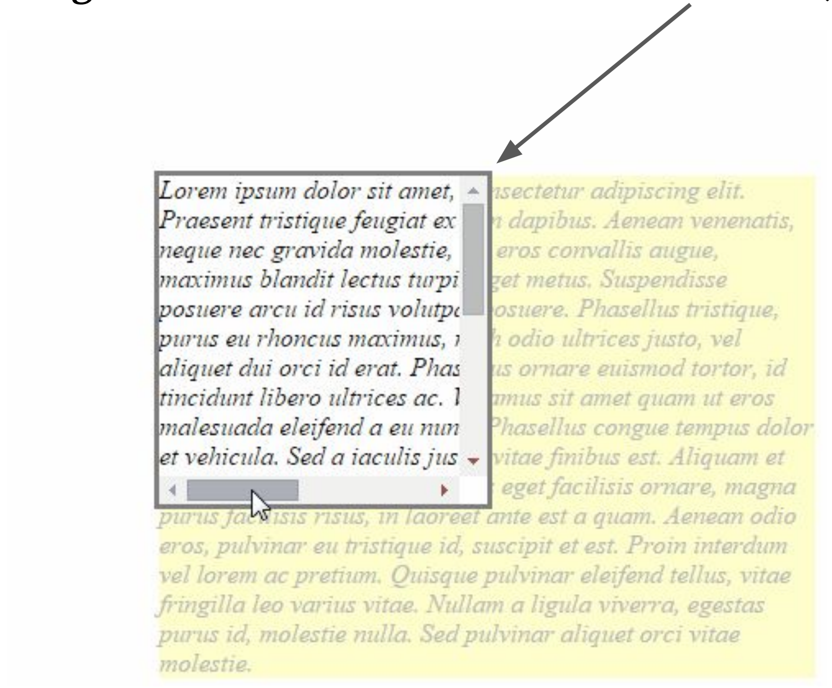
Steve Kobes
skobes@chromium.org

Nov 2015

bit.ly/blink-scrolling

WHAT IS SCROLLING?

Scrolling moves **content** inside a **scrollable area** (or "scroller").



Lorem ipsum dolor sit amet, nsectetur adipiscing elit.
Praesent tristique feugiat ex n dapibus. Aenean venenatis,
neque nec gravida molestie, eros convallis augue,
maximus blandit lectus turpi get metus. Suspendisse
posuere arcu id risus volutp osuere. Phasellus tristique,
purus eu rhoncus maximus, h odio ultrices justo, vel
aliquet dui orci id erat. Phas us ornare euismod tortor, id
tincidunt libero ultrices ac. amus sit amet quam ut eros
malesuada eleifend a eu nun Phasellus congue tempus dolor
et vehicula. Sed a iaculis jus vitae finibus est. Aliquam et
eget facilisis ornare, magna
purus facilisis risus, in laoreet ante est a quam. Aenean odio
eros, pulvinar eu tristique id, suscipit et est. Proin interdum
vel lorem ac pretium. Quisque pulvinar eleifend tellus, vitae
fringilla leo varius vitae. Nullam a ligula viverra, egestas
purus id, molestie nulla. Sed pulvinar aliquet orci vitae
molestie.

Scrolling requires **overflow**.

The overflow is **clipped**.

Scrolling updates a **scroll offset**.

Most scrollable areas have **scrollbars** (or "overflow controls").

WHERE DO SCROLLABLE AREAS COME FROM?

Every **frame** has a scrollable area for its document.

`<iframe>`

`<frameset>`

[CSS 2.1]

default

```
overflow: visible
```

```
overflow: hidden
overflow: scroll
overflow: auto
```

scrollable area

Every **element** has "`overflow`" CSS style.

`<div style="overflow: scroll">`

Some **form controls** are scrollable.

`<input>` `<textarea>` `<select>`

Pinch zoom creates a special scrollable area.

WHAT CAUSES SCROLLING?

The **user** can scroll.

- *mouse wheel*
- *trackpad gesture*
- *touch screen drag*
- *scrollbar interaction*
- *keyboard up/down/...*

The **webpage** can scroll itself with JS.

```
window.scrollTo(...);
window.scrollBy(...);

element.scrollLeft = ...;
element.scrollTop = ...;
element.scrollIntoView();
```

Layout can scroll (by affecting min/max offset).

Navigation scrolls in two ways:

- *to a #hash fragment*
- *to restore a position from session history*

```
<a name="foo">Anchor</a>
```

Scrolling can be a **side effect** of a user action.

- *selection dragging*
- *focus (Tab) movement*
- *finding text (Ctrl-F)*

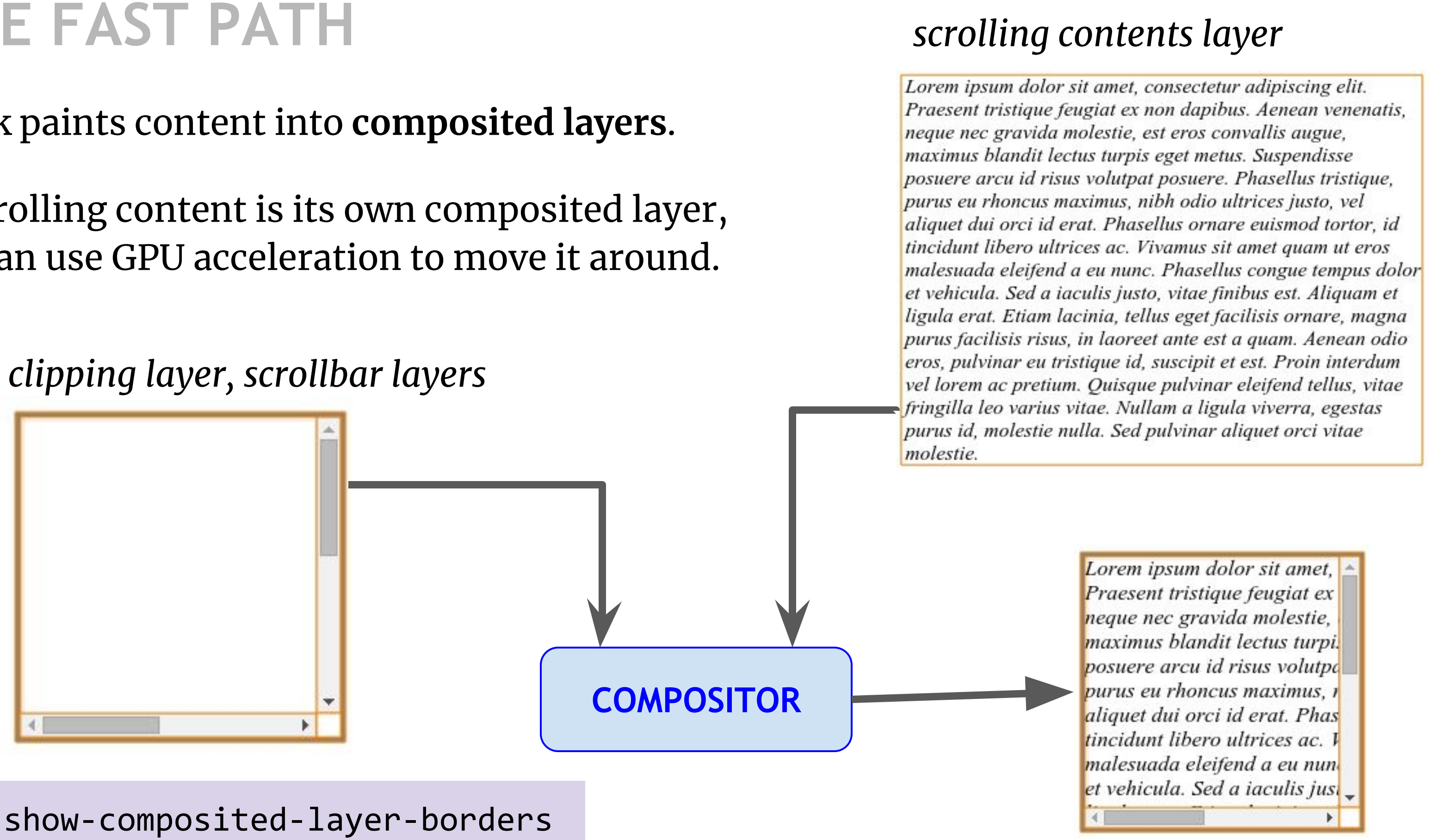
A **<WebView> embedder** can scroll.

```
android.webkit.WebView.scrollTo(x, y)
```

THE FAST PATH

Blink paints content into **composited layers**.

If scrolling content is its own composited layer, we can use GPU acceleration to move it around.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent tristique feugiat ex non dapibus. Aenean venenatis, neque nec gravida molestie, est eros convallis augue, maximus blandit lectus turpis eget metus. Suspendisse posuere arcu id risus volutpat posuere. Phasellus tristique, purus eu rhoncus maximus, nibh odio ultrices justo, vel aliquet dui orci id erat. Phasellus ornare euismod tortor, id tincidunt libero ultrices ac. Vivamus sit amet quam ut eros malesuada eleifend a eu nunc. Phasellus congue tempus dolor et vehicula. Sed a iaculis justo, vitae finibus est. Aliquam et ligula erat. Etiam lacinia, tellus eget facilisis ornare, magna purus facilisis risus, in laoreet ante est a quam. Aenean odio eros, pulvinar eu tristique id, suscipit et est. Proin interdum vel lorem ac pretium. Quisque pulvinar eleifend tellus, vitae fringilla leo varius vitae. Nullam a ligula viverra, egestas purus id, molestie nulla. Sed pulvinar aliquet orci vitae molestie.

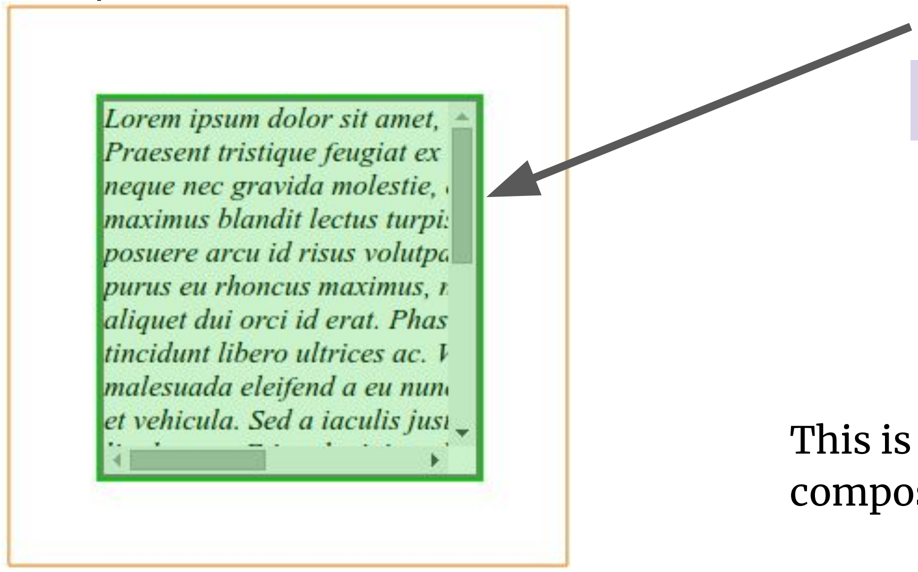
*Lorem ipsum dolor sit amet,
Praesent tristique feugiat ex
neque nec gravida molestie,
maximus blandit lectus turpi
posuere arcu id risus volutp
purus eu rhoncus maximus,
aliquet dui orci id erat. Phas
tincidunt libero ultrices ac.
malesuada eleifend a eu nun
et vehicula. Sed a iaculis jus*

```
--show-composited-layer-borders
```

THE SLOW PATH

If scrolling content is **not** in its own composited layer, a region of the parent layer must be repainted when the scroll offset changes.

composited container



Lorem ipsum dolor sit amet,
Praesent tristique feugiat ex
neque nec gravida molestie,
maximus blandit lectus turpi
posuere arcu id risus volutpa
purus eu rhoncus maximus, r
aliquet dui orci id erat. Phas
tincidunt libero ultrices ac. V
malesuada eleifend a eu nun
et vehicula. Sed a iaculis just

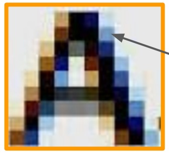
paint invalidation

```
--show-paint-rects
```

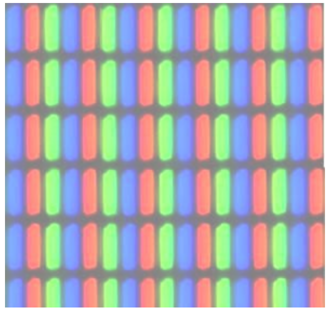
This is more work than a composited scroll.

LCD TEXT

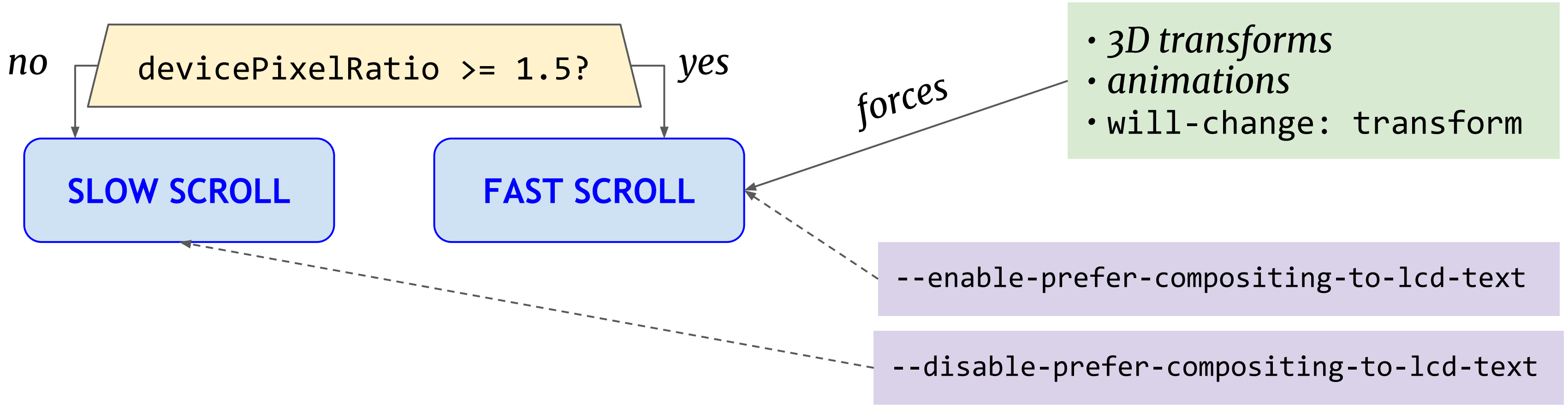
Compositing a **transparent** scroller affects **subpixel antialiasing**.



Edge pixels depend on background color.

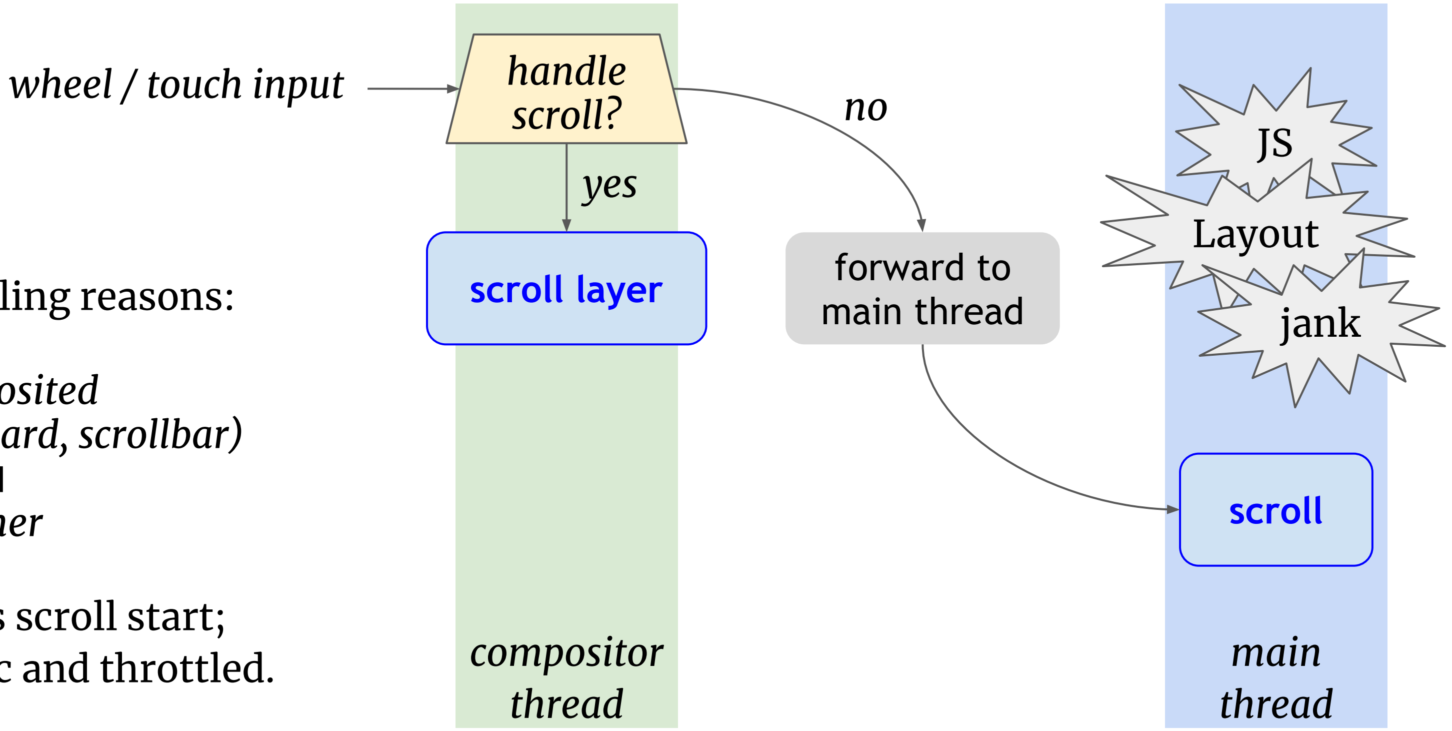


So, we only composite `overflow:scroll` on high-DPI devices.



THREADED SCROLLING

The **compositor thread** can handle some types of scroll input.



Main thread scrolling reasons:

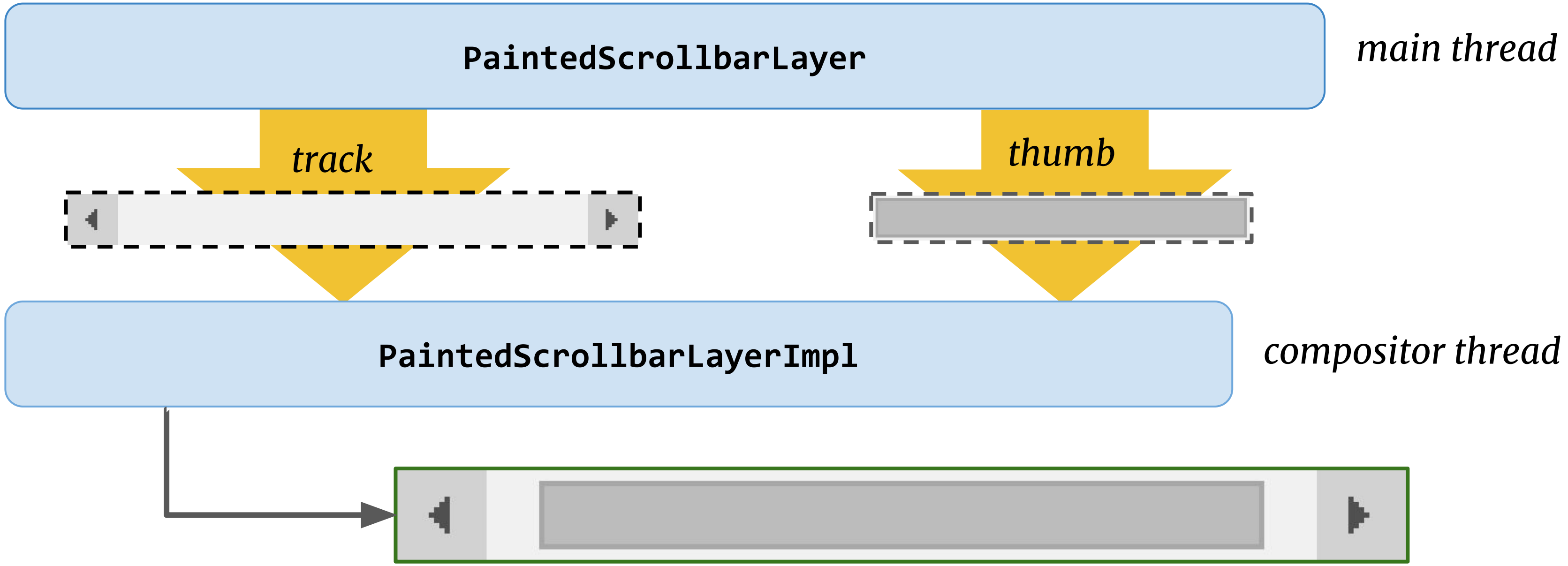
- *scroller not composited*
- *input type (keyboard, scrollbar)*
- `position: fixed`
- `wheel` *event listener*

`touchstart` blocks scroll start;
`touchmove` is async and throttled.

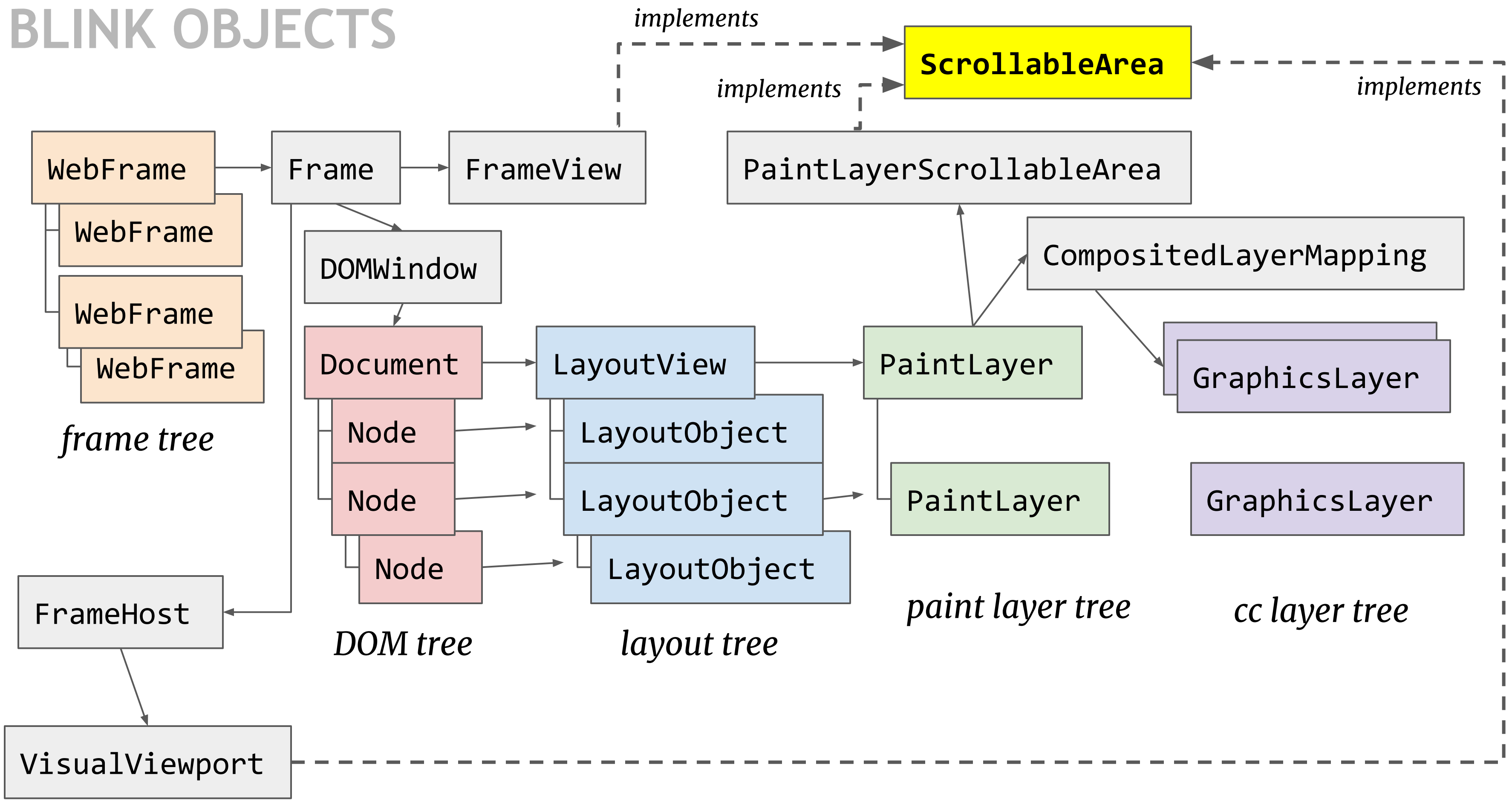
COORDINATED SCROLLBARS

When a scroller is composited, so are its scrollbars.

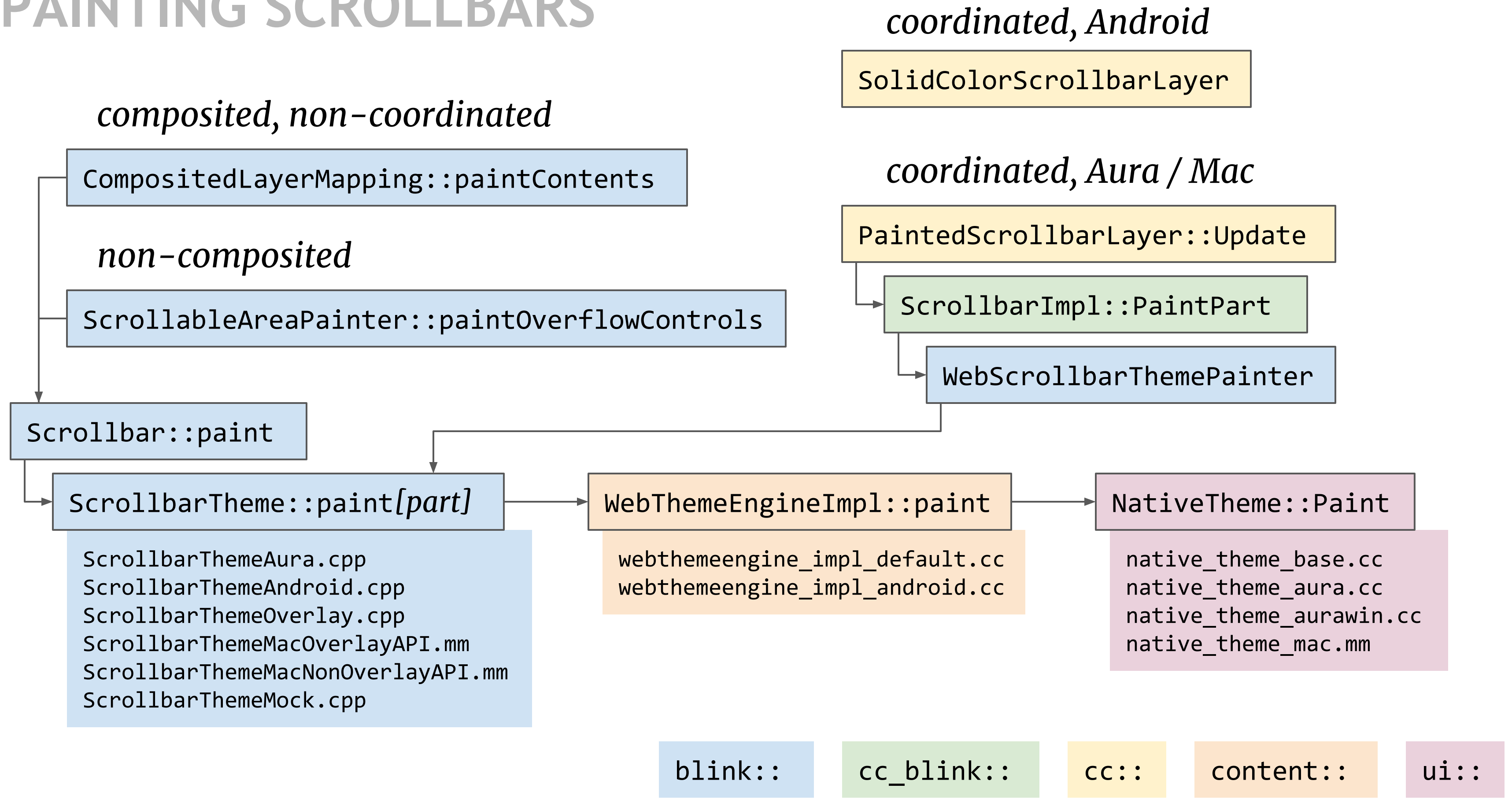
A **coordinated scrollbar** can be updated on the compositor thread.



BLINK OBJECTS



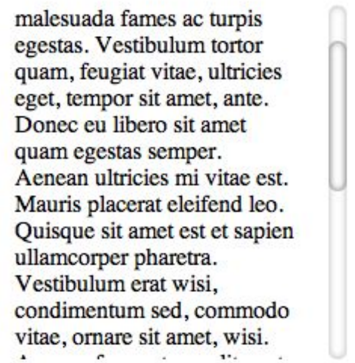
PAINTING SCROLLBARS


composited, non-coordinated
CompositedLayerMapping::paintContents
non-composited
ScrollableAreaPainter::paintOverflowControls
Scrollbar::paint
ScrollbarTheme::paint[part]
ScrollbarThemeAura.cpp
ScrollbarThemeAndroid.cpp
ScrollbarThemeOverlay.cpp
ScrollbarThemeMacOverlayAPI.mm
ScrollbarThemeMacNonOverlayAPI.mm
ScrollbarThemeMock.cpp
coordinated, Android
SolidColorScrollbarLayer
coordinated, Aura / Mac
PaintedScrollbarLayer::Update
ScrollbarImpl::PaintPart
WebScrollbarThemePainter
WebThemeEngineImpl::paint
webthemeengine_impl_default.cc
webthemeengine_impl_android.cc
NativeTheme::Paint
native_theme_base.cc
native_theme_aura.cc
native_theme_aurawin.cc
native_theme_mac.mm
blink::
cc_blink::
cc::
content::
ui::

CUSTOM SCROLLBARS

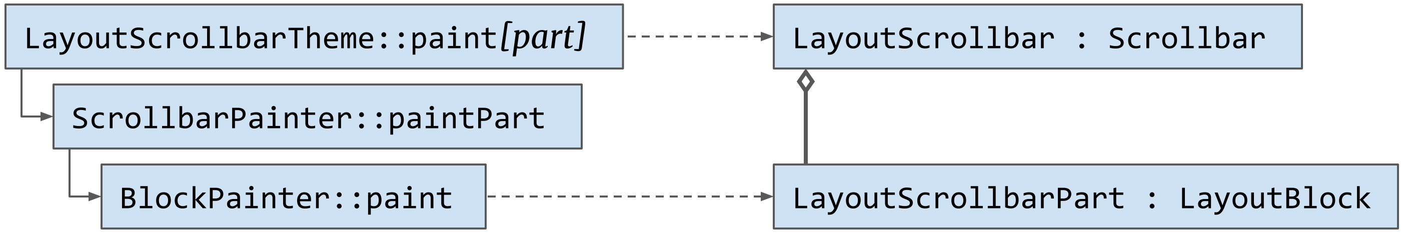
Scrollbars can be styled in CSS.

```
::-webkit-scrollbar { ... }
::-webkit-scrollbar-track { ... }
::-webkit-scrollbar-thumb { ... }
```



malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi.

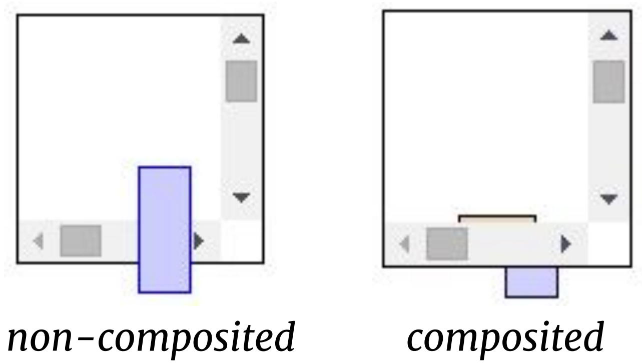
Each piece is a `LayoutBlock`.


LayoutScrollbarTheme::paint[part]
ScrollbarPainter::paintPart
BlockPainter::paint
LayoutScrollbar : Scrollbar
LayoutScrollbarPart : LayoutBlock

- *not standardized*
- *propagates into iframes*
- *state changes can relayout*

PAINT ORDER

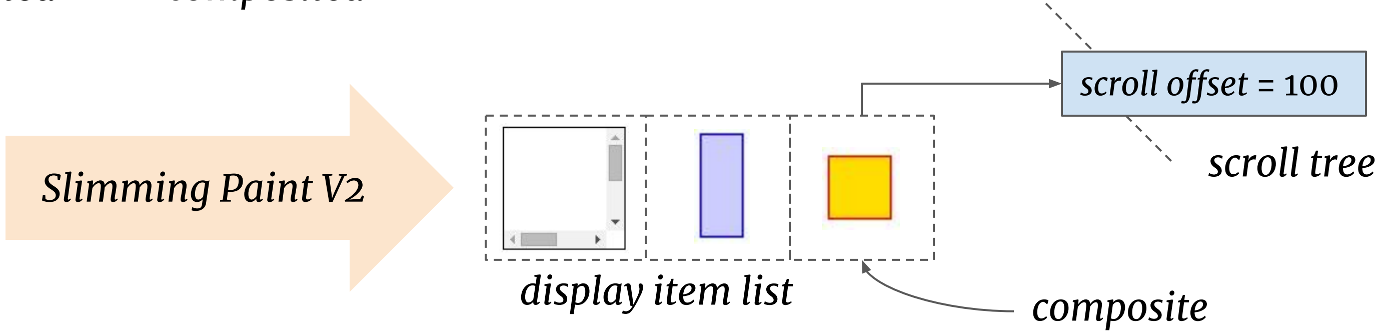
If a scroller is not a **stacking context**, outside content can be interleaved in z-order.



This scroller paints differently when composited.

One example of the **fundamental compositing bug**.

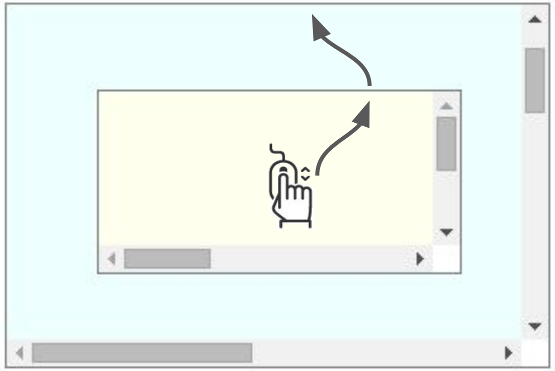
The **slimming paint** project will fix it by making compositing decisions more granular.



CHAINING

Scrollers can nest.

Scroll chaining walks up the **containing block** chain until the delta is consumed.



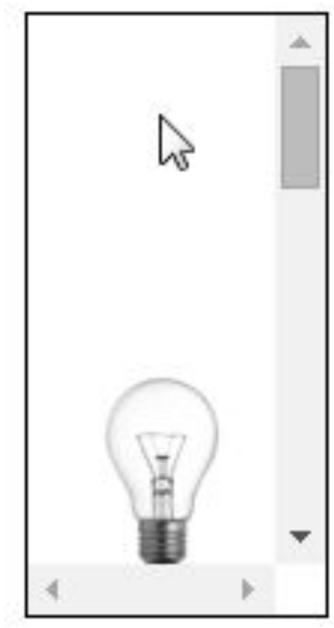
*(Separate from **event bubbling**, which walks up the DOM to invoke listeners.)*

Recent change: **latch** to a single scroller during a touch gesture (chain only at start).

Coming soon: latching for touchpad scrolls.

HOVERING

Elements may enter or exit the **hover state** after a scroll.

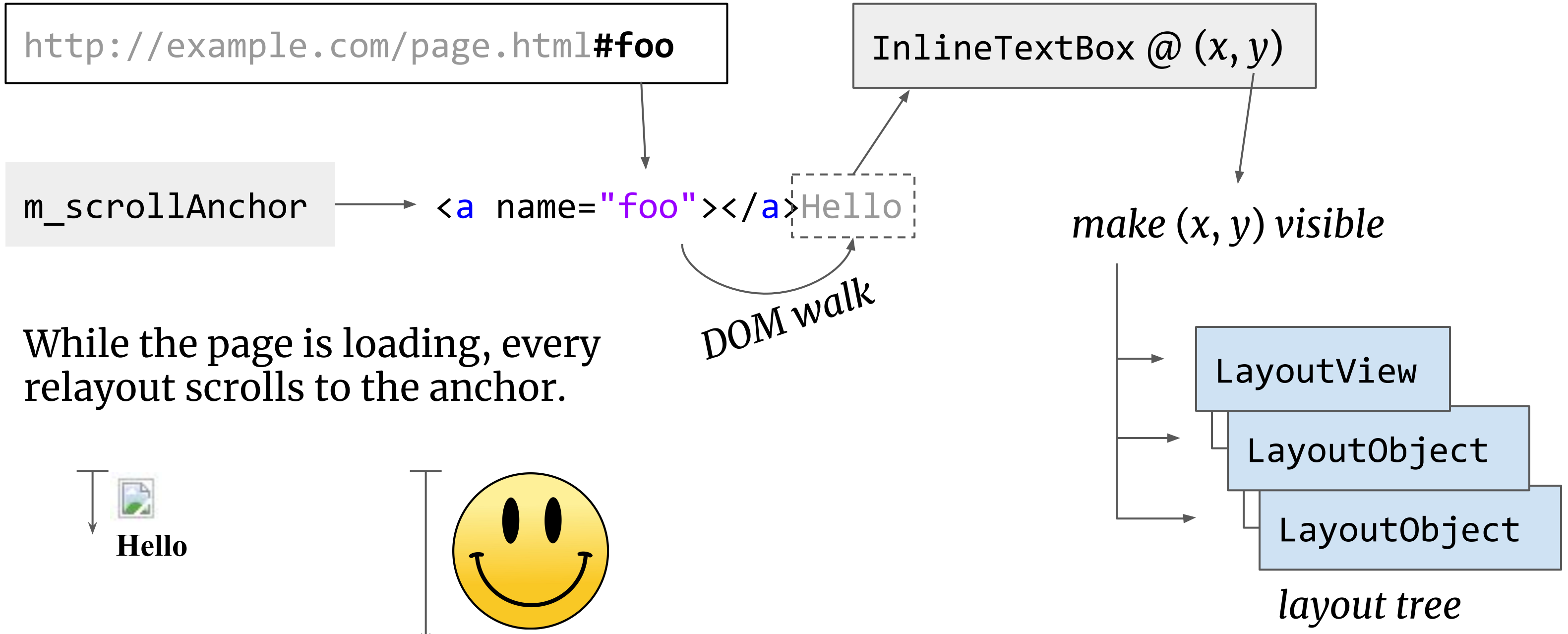


Recent change: defer hover effects and `mousemove` until no scrolls have occurred for 100 ms.

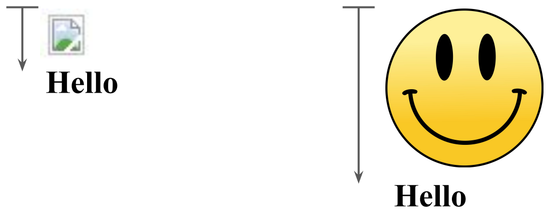
Greatly reduces repaint storms.

ANCHORS

The loader tells the `FrameView` about the **hash fragment**.

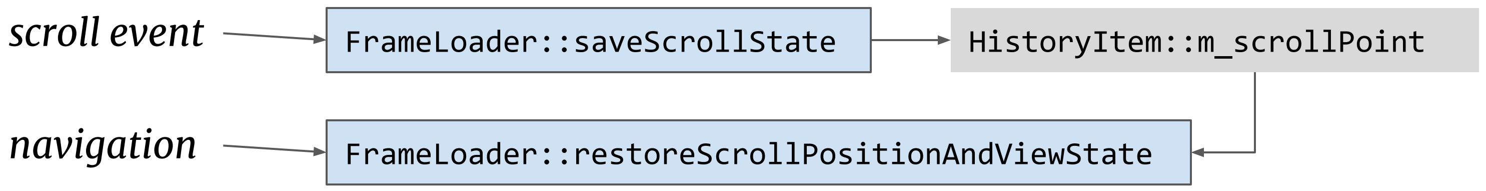


While the page is loading, every relayout scrolls to the anchor.



HISTORY

Each **history item** stores a scroll position along with the URL.


scroll event
FrameLoader::saveScrollState
HistoryItem::m_scrollPoint
navigation
FrameLoader::restoreScrollPositionAndViewState

The page can disable this through the **history API**.

```
history.scrollRestoration = 'manual';
```

History takes precedence over anchors.

RTL

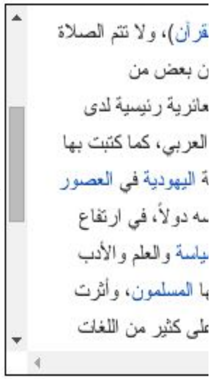
(Coordinate system does not flip.)

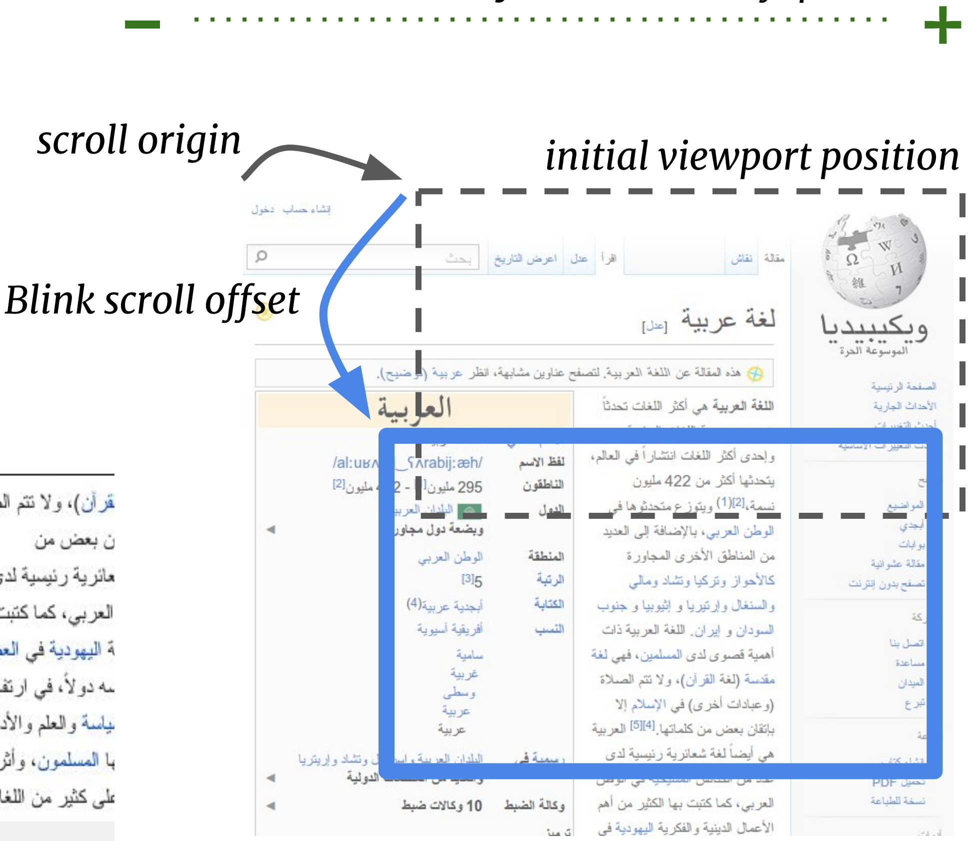
A scroller with `direction: rtl` begins in the rightmost position.

The Blink scroll offset has $x \leq 0$, and is relative to the **scroll origin**.

The `cc::Layer` scroll offset incorporates the scroll origin (as does `element.scrollLeft`).

Elements (but not frames) get left-hand scrollbars.

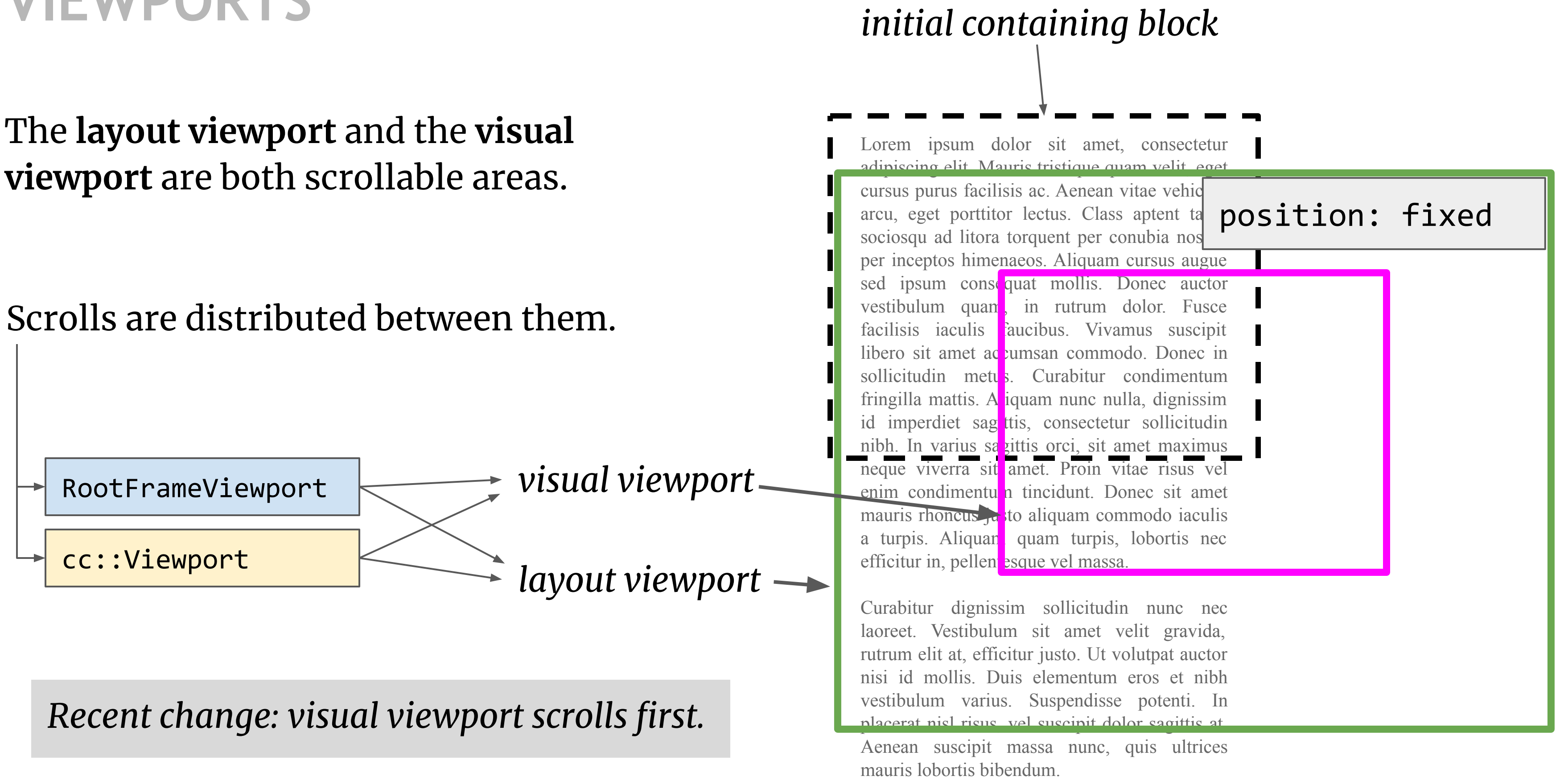




VIEWPORTS

The **layout viewport** and the **visual viewport** are both scrollable areas.

Scrolls are distributed between them.



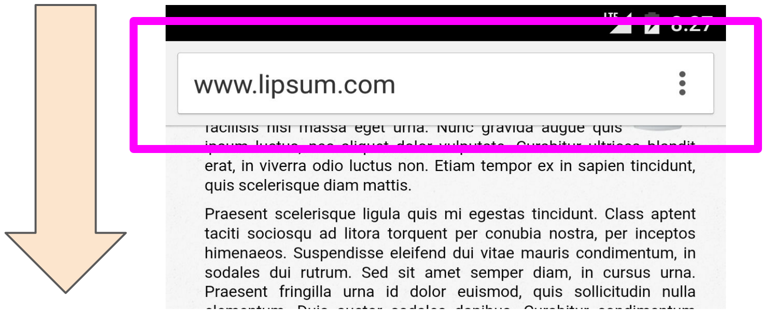
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris tristique quam velit, eget cursus purus facilisis ac. Aenean vitae vehic arcu, eget porttitor lectus. Class aptent ta sociosqu ad litora torquent per conubia nos per inceptos himenaeos. Aliquam cursus augue sed ipsum consequat mollis. Donec auctor vestibulum quam, in rutrum dolor. Fusce facilisis iaculis faucibus. Vivamus suscipit libero sit amet accumsan commodo. Donec in sollicitudin metus. Curabitur condimentum fringilla mattis. Aliquam nunc nulla, dignissim id imperdiet sagittis, consectetur sollicitudin nibh. In varius sagittis orci, sit amet maximus neque viverra sit amet. Proin vitae risus vel enim condimentum tincidunt. Donec sit amet mauris rhoncus justo aliquam commodo iaculis a turpis. Aliquam quam turpis, lobortis nec efficitur in, pellentesque vel massa.

Curabitur dignissim sollicitudin nunc nec laoreet. Vestibulum sit amet velit gravida, rutrum elit at, efficitur justo. Ut volutpat auctor nisi id mollis. Duis elementum eros et nibh vestibulum varius. Suspendisse potenti. In placerat nisl risus, vel suscipit dolor sagittis at. Aenean suscipit massa nunc, quis ultrices mauris lobortis bibendum.

Recent change: visual viewport scrolls first.

TOP CONTROLS

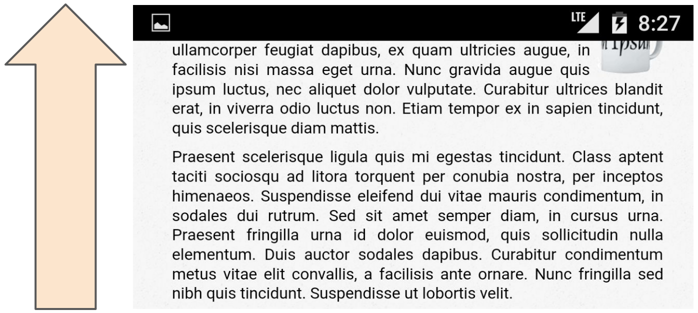
On Android, the **top controls** move in and out of view.


www.lipsum.com

facilisis nisi massa eget urna. Nunc gravida augue quis ipsum luctus, nec aliquet dolor vulputate. Curabitur ultrices blandit erat, in viverra odio luctus non. Etiam tempor ex in sapien tincidunt, quis scelerisque diam mattis.

Praesent scelerisque ligula quis mi egestas tincidunt. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse eleifend dui vitae mauris condimentum, in sodales dui rutrum. Sed sit amet semper diam, in cursus urna. Praesent fringilla urna id dolor euismod, quis sollicitudin nulla

scrolling down



ullamcorper feugiat dapibus, ex quam ultricies augue, in facilisis nisi massa eget urna. Nunc gravida augue quis ipsum luctus, nec aliquet dolor vulputate. Curabitur ultrices blandit erat, in viverra odio luctus non. Etiam tempor ex in sapien tincidunt, quis scelerisque diam mattis.

Praesent scelerisque ligula quis mi egestas tincidunt. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Suspendisse eleifend dui vitae mauris condimentum, in sodales dui rutrum. Sed sit amet semper diam, in cursus urna. Praesent fringilla urna id dolor euismod, quis sollicitudin nulla elementum. Duis auctor sodales dapibus. Curabitur condimentum metus vitae elit convallis, a facilisis ante ornare. Nunc fringilla sed nibh quis tincidunt. Suspendisse ut lobortis velit.

scrolling up

Currently, this resizes the initial containing block, causing relayout.

Proposal: always size the ICB as if top controls are shown.

PASSIVE EVENT LISTENERS

Touch / wheel listeners interfere with threaded scrolling, because the event is **cancelable**.

```
doc.addEventListener("touchstart", function(e) {
  e.preventDefault();
});
```

don't scroll!

Proposal: let listeners be **passive**.

can't cancel

```
doc.addEventListener("touchstart", fn, {passive: true});
```

(in progress)

(`scroll-blocks-on` *is gone)*

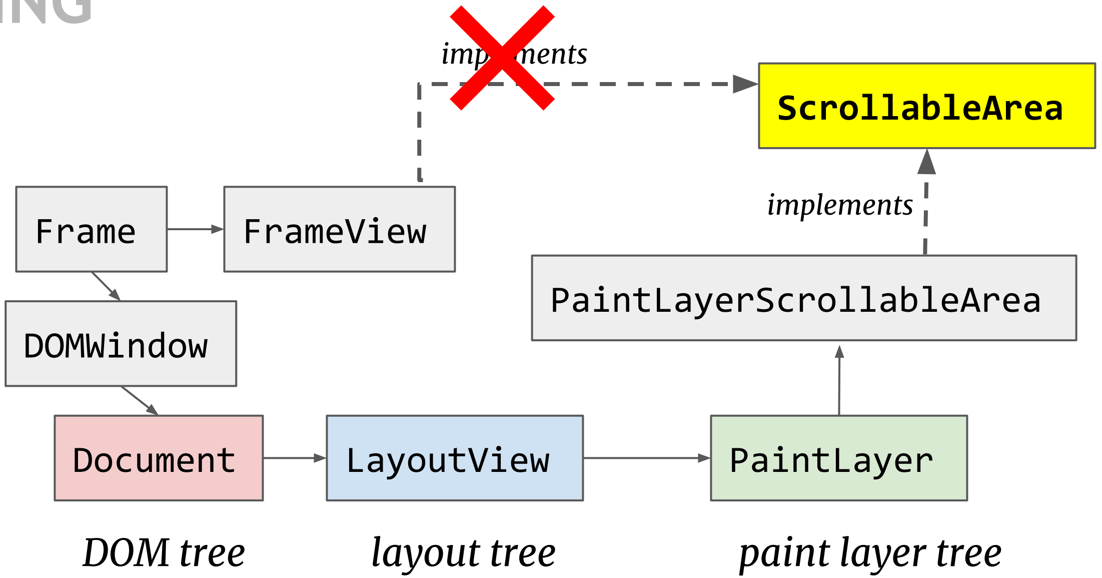
ROOT LAYER SCROLLING

Recall: every PaintLayer has a **ScrollableArea.**

Document has a PaintLayer (the root of the tree).

Why can't the root PaintLayer scroll the document?

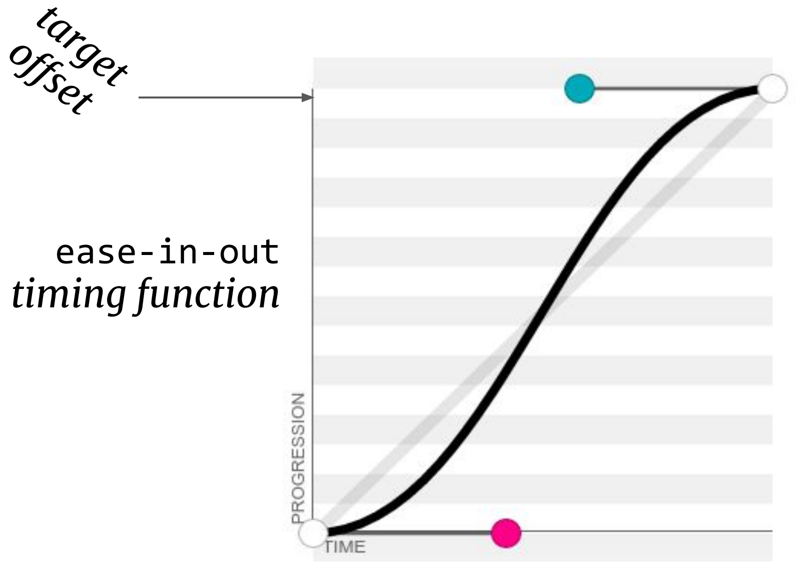
```
--root-layer-scrolls
```



- *eliminates* `FrameView` *scrolling code*
- *document scrolling no longer "special"*
- *reveals bugs in composited* `overflow:scroll`

(in progress)

SMOOTH SCROLLING



(in progress)

Smooth scrolling animates scroll position smoothly up to a target.

input-driven: `--enable-smooth-scrolling`

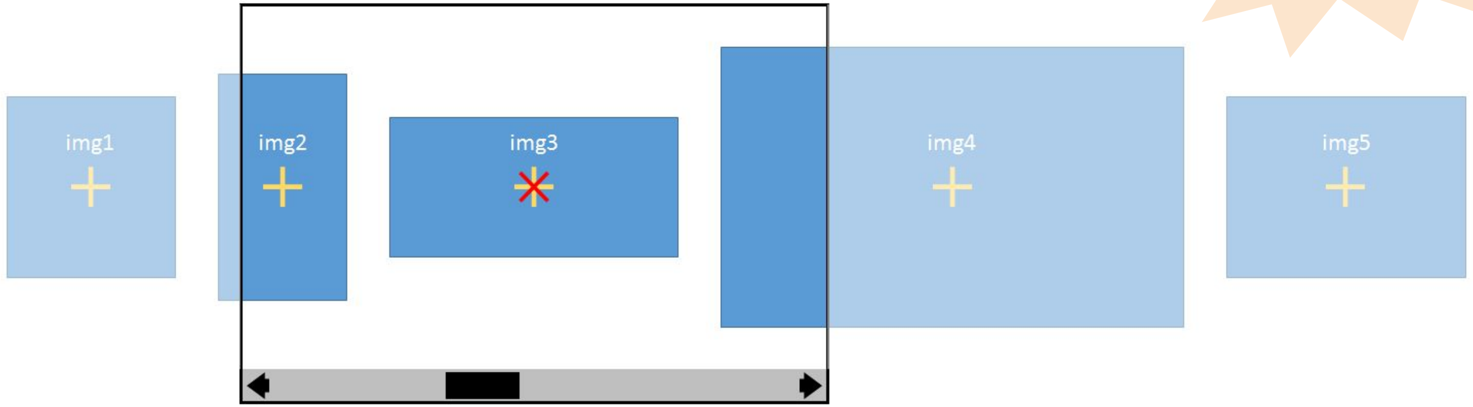
script-driven:

```
element.scroll({
  left: "10",
  top: "10",
  behavior: "smooth"
});
```

`--enable-experimental-web-platform-features`

SNAP POINTS

(in progress)



```
.gallery {
    scroll-snap-destination: 50% 50%;
    scroll-snap-type: mandatory;
}
```

snap to element center

"hard" snap

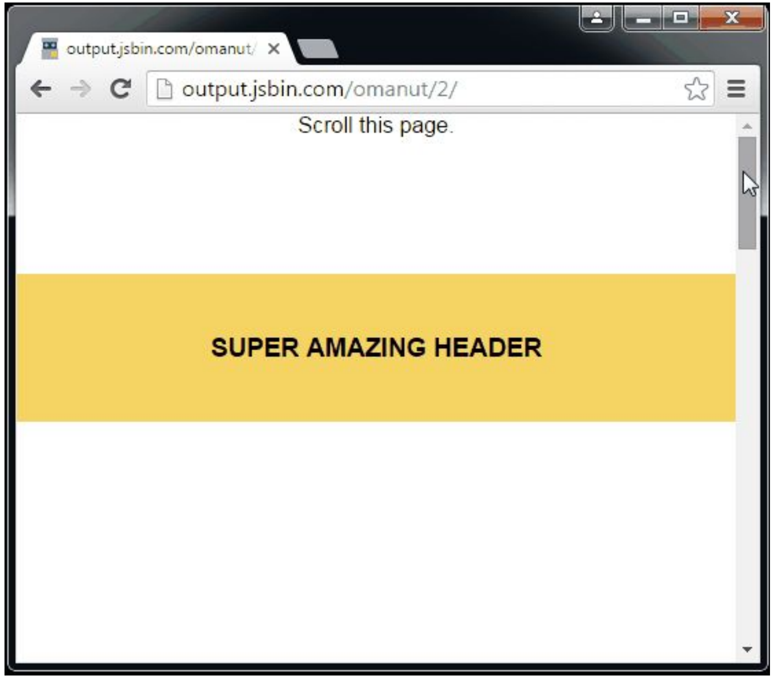
```
--enable-blink-features=CSSScrollSnapPoints
```

POSITION STICKY

(in progress)

```
.header {
    position: sticky;
    top: 0;
}
```

stick to the top


output.jsbin.com/omanut/2/

Scroll this page.

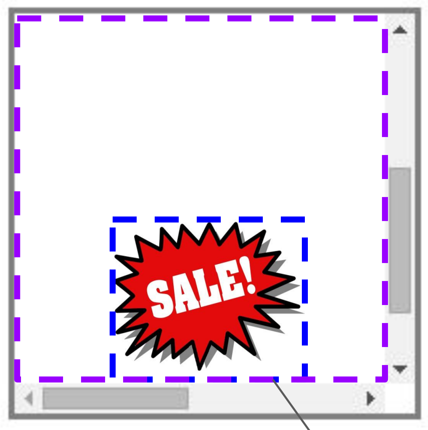
SUPER AMAZING HEADER

INTERSECTION OBSERVER

(in progress)

Invoke a callback when an element scrolls into view.

```
new IntersectionObserver(function(changes) {...}, {}).observe(element);
```



current techniques use expensive polling

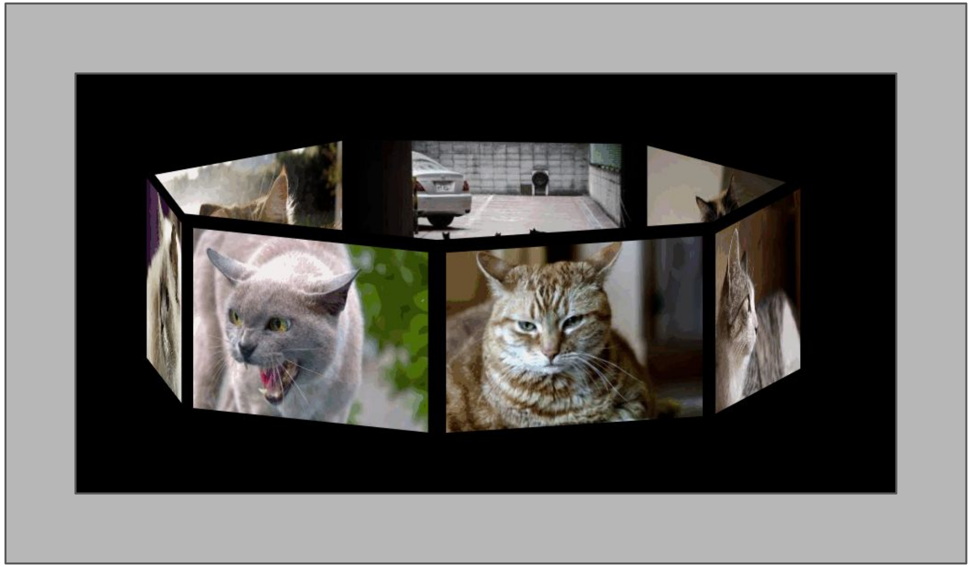
record ad impression

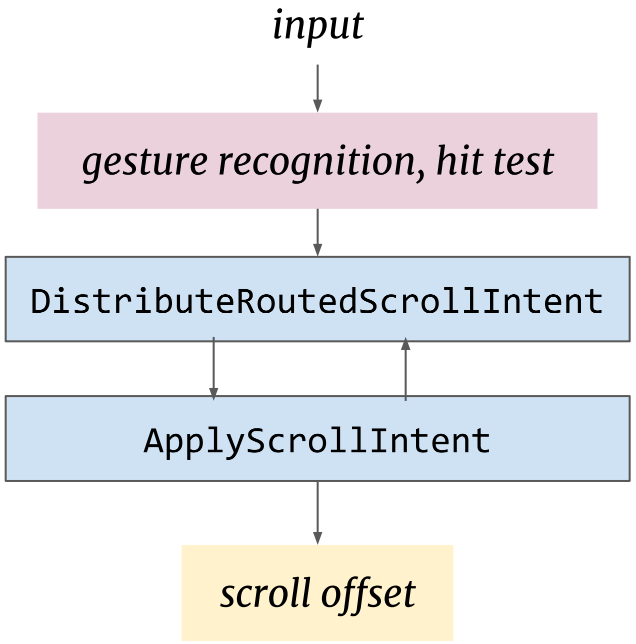
SCROLL CUSTOMIZATION

(in progress)

Proposal: primitives for rich scroll effects in JS.

- *arbitrary composition with native scrollers*
- *use with* ***CompositorWorker*** *for threaded scrolls*





The End

special thanks to technical reviewers:

rbyers, tdresser, bokan, majidvp, chrishtr

bit.ly/blink-scrolling