Take control of your scroll: customizing pull-to-refresh and overflow effects

By Eric Bidelman

Engineer @ Google working on web tooling: Headless Chrome, Puppeteer, Lighthouse

By Majid Valipour

Software Engineer working on Chromium

By Sandra Sun

Software Engineer working on Chromium

TL;DR

The CSS `overscroll-behavior` property allows developers to override the browser's default overflow scroll behavior when reaching the top/bottom of content. Use cases include disabling the pull-to-refresh feature on mobile, removing overscroll glow and rubberbanding effects, and preventing page content from scrolling when it's beneath a modal/overlay.

`overscroll-behavior` requires Chrome 63+. It's in development or being considered by other browsers. See chromestatus.com for more information.

Background

Scroll boundaries and scroll chaining

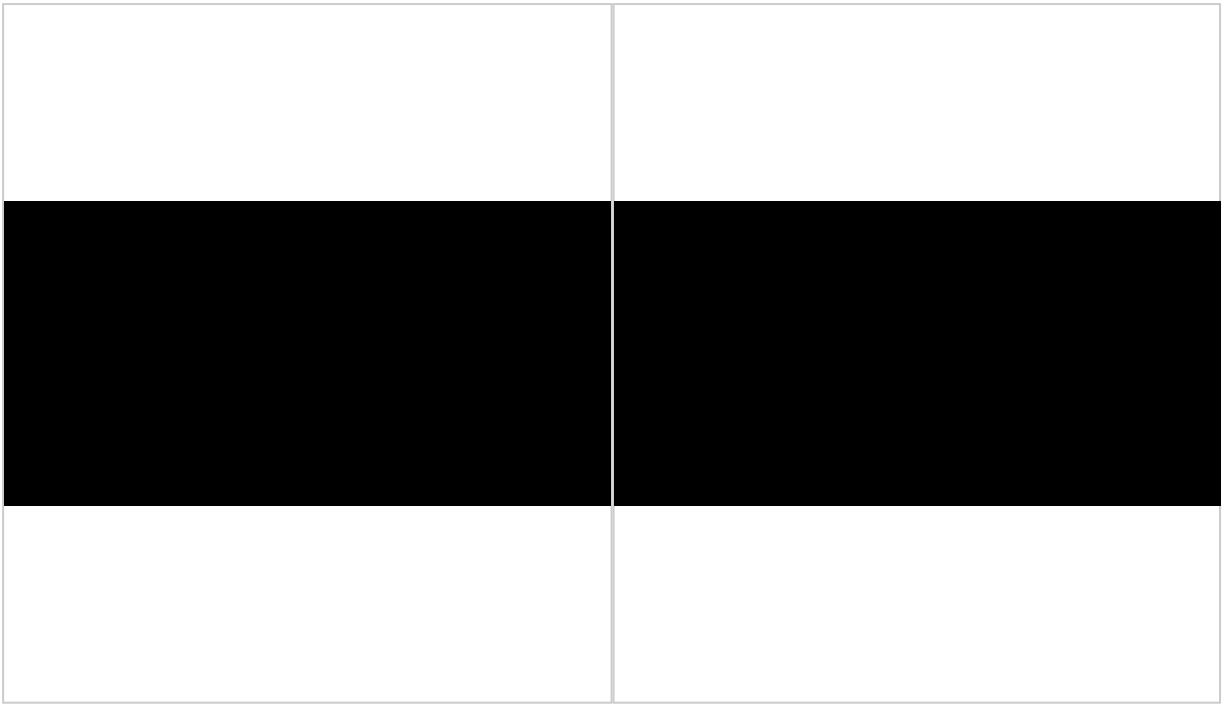
Scroll chaining on Chrome Android.

Scrolling is one of the most fundamental ways to interact with a page, but certain UX patterns can be tricky to deal with because of the browser's quirky default behaviors. As an example, take an app drawer with a large number of items that the user may have to scroll through. When they reach the bottom, the overflow container stops scrolling because there's no more content to consume. In other words, the user reaches a "scroll boundary". But notice what happens if the user continues to scroll. **The content *behind* the drawer starts scrolling**! Scrolling is taken over by the parent container; the main page itself in the example.

Turns out this behavior is called **scroll chaining**; the browser's default behavior when scrolling content. Oftentimes the default is pretty nice, but sometimes it's not desirable or even unexpected. Certain apps may want to provide a different user experience when the user hits a scroll boundary.

The pull-to-refresh effect

Pull-to-refresh is a intuitive gesture popularized by mobile apps such as Facebook and Twitter. Pulling down on a social feed and releasing creates new space for more recent posts to be loaded. In fact, this particular UX has become *so popular* that mobile browsers like Chrome on Android have adopted the same effect. Swiping down at the top of the page refreshes the entire page:



Twitter's custom pull-to-refresh when refreshing a feed in their PWA.

Chrome Android's native pull-to-refresh action refreshes the entire page.

For situations like the Twitter PWA, it might make sense to disable the native pull-to-refresh action. Why? In this app, you probably don't want the user accidentally refreshing the page. There's also the potential to see a double refresh animation! Alternatively, it might be nicer to custom the browser's action, aligning it more closely to the site's branding. The unfortunate part is that this type of customization has been tricky to pull off. Developers end up writing unnecessary JavaScript, add non-passive touch listeners (which block scrolling), or stick the entire page in a 100vw/vh `<div>` (to prevent the page from overflowing). These workarounds have well-documented negative effects on scrolling performance.

We can do better!

Introducing overscroll-behavior

The `overscroll-behavior` property is a new CSS feature that controls the behavior of what happens when you over-scroll a container (including the page itself). You can use it to cancel scroll chaining, disable/customize the pull-to-refresh action, disable rubberbanding effects on iOS (when Safari implements `overscroll-behavior`), and more. The best part is that **using `overscroll-behavior` does not adversely affect page performance** like the hacks mentioned in the intro!

The property takes three possible values:

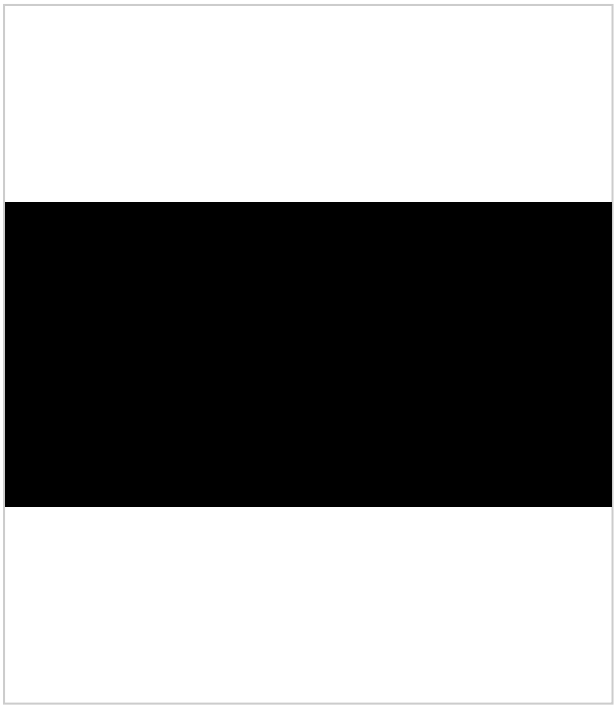
1. **auto** - Default. Scrolls that originate on the element may propagate to ancestor elements.
2. **contain** - prevents scroll chaining. Scrolls do not propagate to ancestors but local effects within the node are shown. For example, the overscroll glow effect on Android or the rubberbanding effect on iOS which notifies the user when they've hit a scroll boundary. **Note**: using `overscroll-behavior: contain` on the `html` element prevents overscroll navigation actions.
3. **none** - same as `contain` but it also prevents overscroll effects within the node itself (e.g. Android overscroll glow or iOS rubberbanding).

Note: `overscroll-behavior` also supports shorthands for `overscroll-behavior-x` and `overscroll-behavior-y` if you only want to define behaviors for a certain axis.

Let's dive into some examples to see how to use `overscroll-behavior`.

Prevent scrolls from escaping a fixed position element

The chatbox scenario



Content beneath the chat window scrolls too :(

Consider a fixed positioned chatbox that sits at the bottom of the page. The intention is that the chatbox is self-contained component and that it scrolls separately from the content

behind it. However, because of scroll chaining, the document starts scrolling as soon as the user hits the last message in the chat history.

For this app, it's more appropriate to have scrolls that originate within the chatbox stay within the chat. We can make that happen by adding `overscroll-behavior: contain` to the element that holds the chat messages:

```
#chat .msgs {
  overflow: auto;
  overscroll-behavior: contain;
  height: 300px;
}
```

Essentially, we're creating a logical separation between the chatbox's scrolling context and the main page. The end result is that the main page stays put when the user reaches the top/bottom of the chat history. Scrolls that start in the chatbox do not propagate out.

The page overlay scenario

Another variation of the "underscroll" scenario is when you see content scrolling behind a **fixed position overlay**. A dead giveaway `overscroll-behavior` is in order! The browser is trying to be helpful but it ends up making the site look buggy.

Example - modal with and without `overscroll-behavior: contain`:

***Before**: page content scrolls beneath overlay.* ***After**: page content doesn't scroll beneath overlay.*

Disabling pull-to-refresh

Turning off the pull-to-refresh action is a single line of CSS. Just prevent scroll chaining on the entire viewport-defining element. In most cases, that's `<html>` or `<body>`:

```
body {
  /* Disables pull-to-refresh but allows overscroll glow effects. */
  overscroll-behavior-y: contain;
}
```

With this simple addition, we fix the double pull-to-refresh animations in the chatbox demo and can instead, implement a custom effect which uses a neater loading animation. The entire inbox also blurs as the inbox refreshes:

Before *After*

Here's a snippet of the full code:

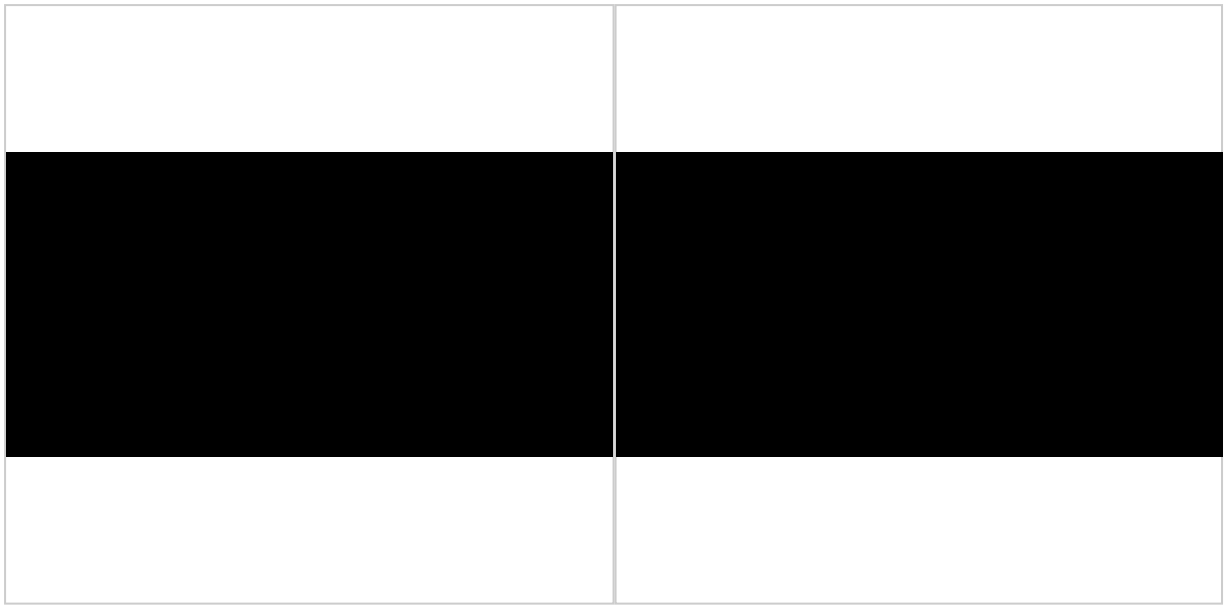
```
<style>
  body.refreshing #inbox {
    filter: blur(1px);
    touch-action: none; /* prevent scrolling */
  }
  body.refreshing .refresher {
    transform: translate3d(0,150%,0) scale(1);
    z-index: 1;
  }
  .refresher {
    --refresh-width: 55px;
    pointer-events: none;
    width: var(--refresh-width);
    height: var(--refresh-width);
    border-radius: 50%;
```

```
    position: absolute;
    transition: all 300ms cubic-bezier(0,0,0.2,1);
    will-change: transform, opacity;
    ...
  }
</style>

<div class="refresher">
  <div class="loading-bar"></div>
  <div class="loading-bar"></div>
  <div class="loading-bar"></div>
  <div class="loading-bar"></div>
</div>

<section id="inbox"><!-- msgs --></section>

<script>
  let _startY;
  const inbox = document.querySelector('#inbox');

  inbox.addEventListener('touchstart', e => {
    _startY = e.touches[0].pageY;
  }, {passive: true});

  inbox.addEventListener('touchmove', e => {
    const y = e.touches[0].pageY;
    // Activate custom pull-to-refresh effects when at the top of the container
    // and user is scrolling up.
    if (document.scrollingElement.scrollTop === 0 && y > _startY &&
        !document.body.classList.contains('refreshing')) {
      // refresh inbox.
    }
  }, {passive: true});
</script>
```

Disabling overscroll glow and rubberbanding effects

To disable the bounce effect when hitting a scroll boundary, use **`overscroll-behavior-y: none`**:

```
body {
  /* Disables pull-to-refresh and overscroll glow effect.
     Still keeps swipe navigations. */
  overscroll-behavior-y: none;
}
```



***Before**: hitting scroll boundary shows a glow.*

***After**: glow disabled.*

Note: This will still preserve left/right swipe navigations. To prevent navigations, you can use `overscroll-behavior-x: none`. However, this is still being implemented in Chrome.

Full demo

Putting it all together, the full chatbox demo, uses `overscroll-behavior` to create a custom pull-to-refresh animation and disable scrolls from escaping the chatbox widget. This provides an optimal user experience that would have been tricky to achieve without CSS `overscroll-behavior`.

View demo | *Source*

Last updated July 2, 2018.