# A More Compatible, Smoother Touch

**By** Paul Kinlan
Paul is a Developer Advocate

You and your users want mobile web apps that react and scroll smoothly to the touch. Developing them should be easy but, unfortunately, how mobile web browsers react to touch events during scrolling is left as an implementation detail in the TouchEvent specification. As a result, approaches can be broken down roughly into 4 categories. This situation exposes a fundamental tension between delivering scroll smoothness and maintaining developer control.

## Four different models of touch event processing?

The behavior differences between the browsers break down into four models.

1. **Normal synchronous event processing**

   Touchmove events are sent during scrolling and each scroll update blocks until touchmove handling has completed. This is good as the simplest to understand and the most powerful but bad for scroll performance because it means that each frame during the scroll must block on the main thread.

   Browsers: Android Browser (Android 4.0.4, 4.3), Mobile Safari (when scrolling div)

2. **Asynchronous touchmove processing**

   Touchmove events are sent during scrolling, but scrolling can proceed asynchronously (the touchmove event is ignored after scrolling has begun). This can result in "double handling" of events, for example, continuing to scroll after the website does something with the touchmove and calls preventDefault on the event, telling the browser not to handle it.

   Browsers: Mobile Safari (when scrolling Document), Firefox

3. **Touchmove suppressed while scrolling**

   Touchmove events are not sent after scrolling starts and do not resume until after the touchend event. In this model, it's hard to tell the difference between a stationary touch

and a scroll.

Browsers: Samsung Browser (mousemove events sent)

4. **Touchcancel on scroll start**

   You can't have it both ways – scroll smoothness and developer control – and this model makes clear the trade-off between smooth scrolling and event handling, similar to the semantics of the Pointer Events specification. Some experiences that may need to track the finger, like pull-to-refresh, are not possible.

   Browsers: Chrome Desktop M32+, Chrome Android

## Why Change?

Chrome for Android currently uses Chrome's Old Model: touchcancel on scroll start, which enhances scrolling performance, but leads to developer confusion. In particular, some developers aren't aware of the touchcancel event or how to deal with it, and this has caused some websites to break. More importantly, an entire class of UI scrolling effects and behaviors, such as **pull-to-refresh**, **hidey bars**, and **snap points** are difficult or impossible to implement well.

Rather than adding specifically hardcoded features to support these effects, Chrome aims to concentrate on adding platform primitives that allow developers to implement these effects directly. See A Rational Web Platform for a general exposition of this philosophy.

## Chrome's New Model: The Throttled Async Touchmove Model

Chrome is introducing a new behavior designed to improve compatibility with code written for other browsers when scrolling, as well as enabling other scenarios that depend on getting touchmove events while scrolling. This feature is enabled by default and you can turn it off with the following flag, **chrome://flags#touch-scrolling-mode**.

The new behavior is:

- The **first** touchmove is sent **synchronously** to allow scroll to be **canceled**
- During active scrolling
  - touchmove events are sent **asynchronously**

    - **throttled** to 1 event per **200ms**, or if a CSS **15px** slop region is exceeded
    - **Event.cancelable** is **false**
- Otherwise, touchmove events fire synchronously as normal when active scrolling terminates or isn't possible because the scroll limit has been hit
- A touchend event **always** occurs when the user lifts their finger

You can try this demo in Chrome for Android and toggle the chrome://flags#touch-scrolling-mode flag to see the difference.

## Let us know what you think

The Async Touchmove Model has the potential to improve cross-browser compatibility and enable a new class of touch gesture effects. We're interested in hearing what developers think, and in seeing the creative things you can do with it.

---



*Last updated July 2, 2018.*