Interaction to Next Paint (INP) becomes a Core Web Vital on March 12. Start making your websites more responsive to user input today. Learn how. (https://web.dev/blog/inp-cwv-march-12?utm_source=web.dev&utm_medium=banner&utm_campaign=inp-cwv)

First Input Delay (FID)

Philip Walton
(https://twitter.com/philwalton) (https://github.com/philipwalton) (https://philipwalton.com/)

Browser Support				
	76	79	89	x

Source (https://developer.mozilla.org/docs/Web/API/PerformanceEventTiming)

Note: First Input Delay (FID) is the stable (/articles/vitals#lifecycle) Core Web Vital metric for measuring load responsiveness because it quantifies the experience users feel when trying to interact with unresponsive pages —a low FID helps ensure that the page is usable. FID will be replaced by Interaction to Next Paint (INP) (/blog/inp-cwv-march-12) as a Core Web Vital on March 12, 2024.

We all know how important it is to make a good first impression. It's important when meeting new people, and it's also important when building experiences on the web.

On the web, a good first impression can make the difference between someone becoming a loyal user or them leaving and never coming back. The question is, what makes for a good impression, and how do you measure what kind of impression you're likely making on your users?

On the web, first impressions can take a lot of different forms—we have first impressions of a site's design and visual appeal as well as first impressions of its speed and responsiveness.

While it is hard to measure how much users like a site's design with web APIs, measuring its

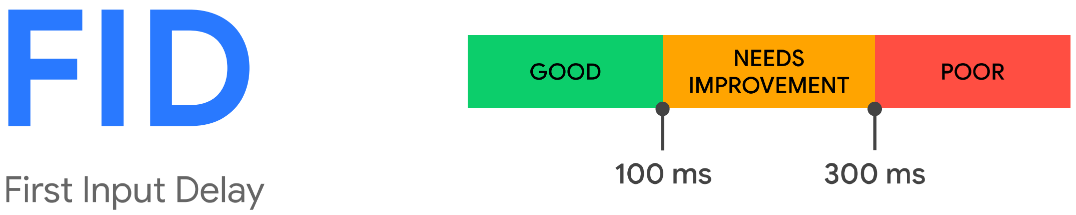
The First Input Delay (FID) metric helps measure your user's first impression of your site's interactivity and responsiveness.

What is FID?

FID measures the time from when a user first interacts with a page (that is, when they click a link, tap on a button, or use a custom, JavaScript-powered control) to the time when the browser is actually able to begin processing event handlers in response to that interaction.

What is a good FID score?

To provide a good user experience, sites should strive to have a First Input Delay of **100 milliseconds** or less. To ensure you're hitting this target for most of your users, a good threshold to measure is the **75th percentile** of page loads, segmented across mobile and desktop devices.



Note: To learn more about the research and methodology behind this recommendation, see: Defining the Core Web Vitals metrics thresholds (/articles/defining-core-web-vitals-thresholds)

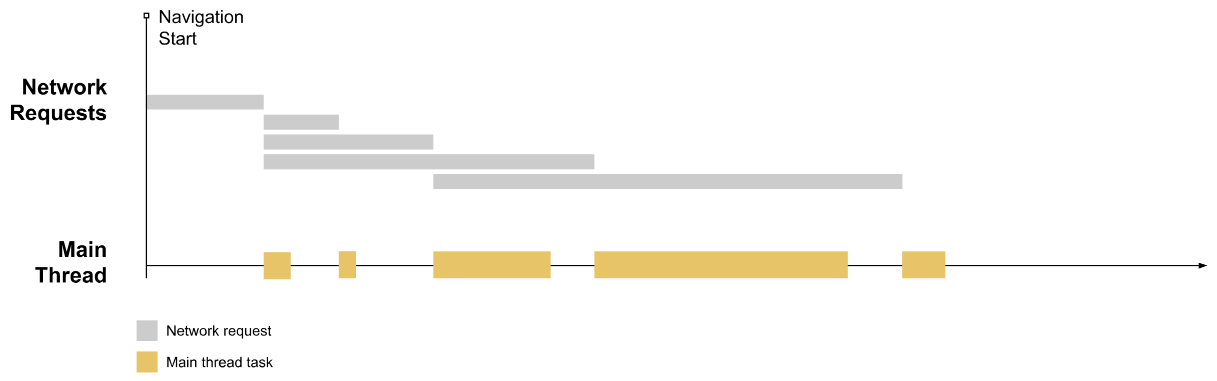
FID in detail

As developers who write code that responds to events, we often assume our code is going to be run immediately—as soon as the event happens. But as users, we've all frequently experienced the opposite—we've loaded a web page on our phone, tried to interact with it, and then been frustrated when nothing happened.

In general, input delay (a.k.a. input latency) happens because the browser's main thread is busy

FID only measures the "delay" in event processing. It does not measure the event processing time itself nor the time it takes the browser to update the UI after running event handlers. While this time does affect the user experience, including it as part of FID would incentivize developers to respond to events asynchronously—which would improve the metric but likely make the experience worse. See why only consider the input delay (#why_only_consider_the_input_delay) below for more details.

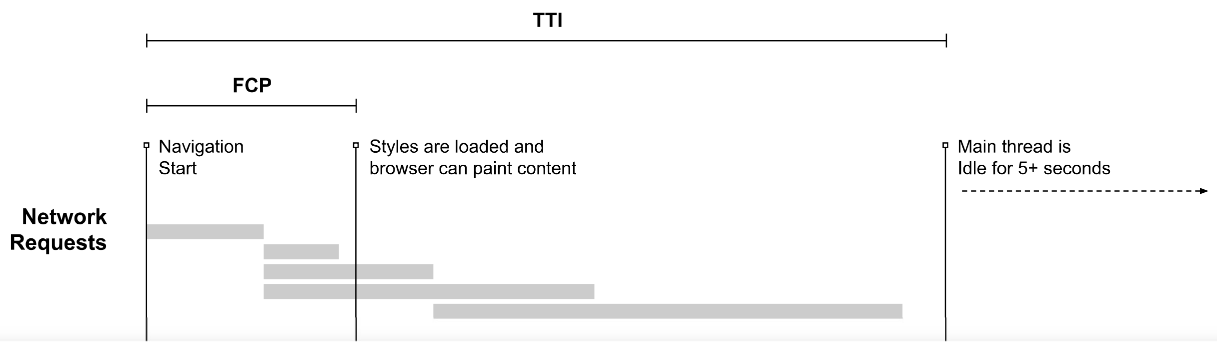
Consider the following timeline of a typical web page load:



The above visualization shows a page that's making a couple of network requests for resources (most likely CSS and JS files), and—after those resources are finished downloading—they're processed on the main thread.

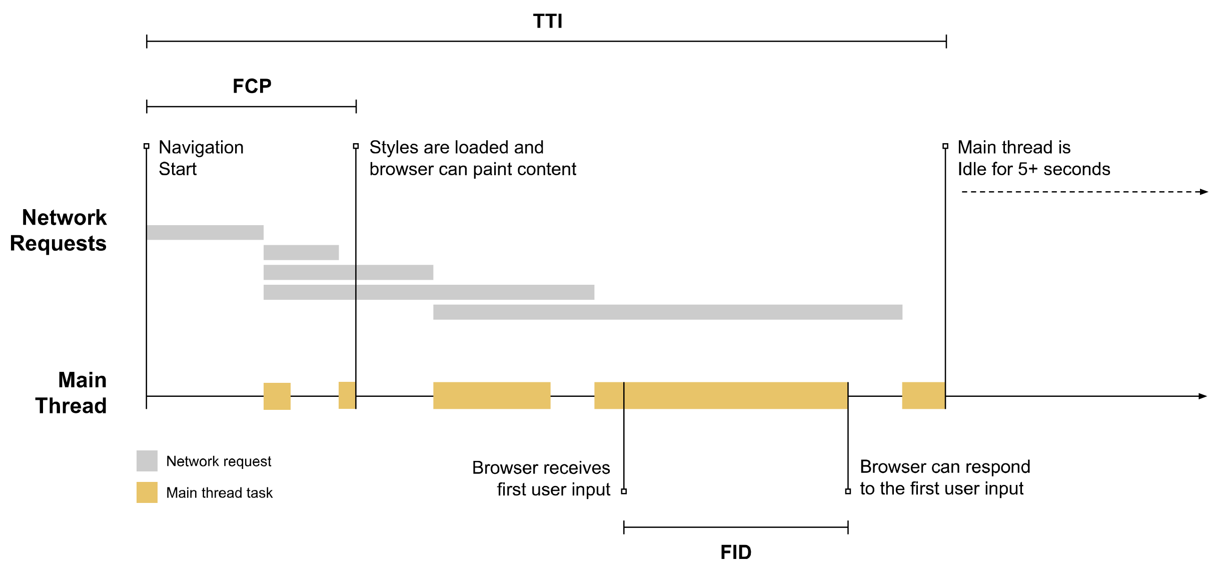
This results in periods where the main thread is momentarily busy, which is indicated by the beige-colored task (https://html.spec.whatwg.org/multipage/webappapis.html#concept-task) blocks.

Long first input delays typically occur between First Contentful Paint (FCP) (/articles/fcp) and Time to Interactive (TTI) (/articles/tti) because the page has rendered some of its content but isn't yet reliably interactive. To illustrate how this can happen, FCP and TTI have been added to the timeline:


TTI
FCP
Navigation Start
Styles are loaded and browser can paint content
Main thread is Idle for 5+ seconds
Network Requests

You may have noticed that there's a fair amount of time (including three long tasks (/articles/custom-metrics#long-tasks-api)) between FCP and TTI, if a user tries to interact with the page during that time (for example, by clicking on a link), there will be a delay between when the click is received and when the main thread is able to respond.

Consider what would happen if a user tried to interact with the page near the beginning of the longest task:


TTI
FCP
Navigation Start
Styles are loaded and browser can paint content
Main thread is Idle for 5+ seconds
Network Requests
Main Thread
Network request
Main thread task
Browser receives first user input
Browser can respond to the first user input
FID

Because the input occurs while the browser is in the middle of running a task, it has to wait until the task completes before it can respond to the input. The time it must wait is the FID value for this user on this page.

Note: In this example the user just happened to interact with the page at the beginning of the main thread's most busy period. If the user had interacted with the page just a moment earlier (during the idle period) the browser could have responded right away. This variance in input delay underscores the importance of looking at the distribution of FID values when reporting on the metric. You can read more about this in the section below on analyzing and reporting on FID data.

What if an interaction doesn't have an event listener?

FID measures the delta between when an input event is received and when the main thread is next idle. This means FID is measured **even in cases where an event listener has not been registered.** The reason is because many user interactions do not require an event listener but *do* require the

- Select dropdowns (`<select>`)
- links (`<a>`)

Why only consider the first input?

While a delay from any input can lead to a bad user experience, we primarily recommend measuring the first input delay for a few reasons:

- The first input delay will be the user's first impression of your site's responsiveness, and first impressions are critical in shaping our overall impression of a site's quality and reliability.
- The biggest interactivity issues we see on the web today occur during page load. Therefore, we believe initially focusing on improving site's first user interaction will have the greatest impact on improving the overall interactivity of the web.
- The recommended solutions for how sites should fix high first input delays (code splitting, loading less JavaScript upfront, etc.) are not necessarily the same solutions for fixing slow input delays after page load. By separating out these metrics we'll be able to provide more specific performance guidelines to web developers.

What counts as a first input?

FID is a metric that measures a page's responsiveness during load. As such, it only focuses on input events from discrete actions like clicks, taps, and key presses.

Other interactions, like scrolling and zooming, are continuous actions and have completely different performance constraints (also, browsers are often able to hide their latency by running them on a separate thread).

To put this another way, FID focuses on the R (responsiveness) in the RAIL performance model (/articles/rail), whereas scrolling and zooming are more related to A (animation), and their performance qualities should be evaluated separately.

What if a user never interacts with your site?

Not all users will interact with your site every time they visit. And not all interactions are relevant to FID (as mentioned in the previous section). In addition, some user's first interactions will be at bad times (when the main thread is busy for an extended period of time), and some user's first you may be used to. The next section explains how best to do this.

Why only consider the input delay?

As mentioned above, FID only measures the "delay" in event processing. It does not measure the event processing time itself nor the time it takes the browser to update the UI after running event handlers.

Even though this time is important to the user and *does* affect the experience, it's not included in this metric because doing so could incentivize developers to add workarounds that actually make the experience worse—that is, they could wrap their event handler logic in an asynchronous callback (via `setTimeout()` or `requestAnimationFrame()`) in order to separate it from the task associated with the event. The result would be an improvement in the metric score but a slower response as perceived by the user.

However, while FID only measure the "delay" portion of event latency, developers who want to track more of the event lifecycle can do so using the Event Timing API (https://wicg.github.io/event-timing/). See the guide on custom metrics (/articles/custom-metrics#event_timing_api) for more details.

How to measure FID

FID is a metric that can only be measured in the field (/articles/user-centric-performance-metrics#in_the_field), as it requires a real user to interact with your page. You can measure FID with the following tools.

Note: FID requires a real user and thus cannot be measured in the lab. However, the Total Blocking Time (TBT) (/articles/tbt) metric is lab-measurable, correlates well with FID in the field, and also captures issues that affect interactivity. Optimizations that improve TBT in the lab should also improve FID for your users.

Field tools

- Chrome User Experience Report (https://developer.chrome.com/docs/crux/)
- PageSpeed Insights (https://pagespeed.web.dev/)
- Search Console (Core Web Vitals report)

To measure FID in JavaScript, you can use the Event Timing API (https://wicg.github.io/event-timing). The following example shows how to create a **`PerformanceObserver`** (https://developer.mozilla.org/docs/Web/API/PerformanceObserver) that listens for **`first-input`** (https://wicg.github.io/event-timing/#sec-performance-event-timing) entries and logs them to the console:

```
new PerformanceObserver((entryList) => {
  for (const entry of entryList.getEntries()) {
    const delay = entry.processingStart - entry.startTime;
    console.log('FID candidate:', delay, entry);
  }
}).observe({type: 'first-input', buffered: true});
```

Warning: This code shows how to log **`first-input`** entries to the console and calculate their delay. However, measuring FID in JavaScript is more complicated. See below for details:

In the above example, the **`first-input`** entry's delay value is measured by taking the delta between the entry's **`startTime`** and **`processingStart`** timestamps. In most cases this will be the FID value; however, not all **`first-input`** entries are valid for measuring FID.

The following section lists the differences between what the API reports and how the metric is calculated.

Differences between the metric and the API

- The API will dispatch **`first-input`** entries for pages loaded in a background tab but those pages should be ignored when calculating FID.
- The API will also dispatch **`first-input`** entries if the page was backgrounded prior to the first input occurring, but those pages should also be ignored when calculating FID (inputs are only considered if the page was in the foreground the entire time).
- The API does not report **`first-input`** entries when the page is restored from the back/ forward cache (/articles/bfcache#impact_on_core_web_vitals), but FID should be measured in these cases since users experience them as distinct page visits.
- The API does not report inputs that occur within iframes but the metric does as they are part of the user experience of the page. This can show as a difference between CrUX and RUM

differences for you (where possible—note the iframe issue is not covered):

```
import {onFID} from 'web-vitals';

// Measure and log FID as soon as it's available.
onFID(console.log);
```

You can refer to the source code for `onFID()` (https://github.com/GoogleChrome/web-vitals/blob/main/src/onFID.ts) for a complete example of how to measure FID in JavaScript.

Note: In some cases (such as cross-origin iframes) it's not possible to measure FID in JavaScript. See the limitations (https://github.com/GoogleChrome/web-vitals#limitations) section of the `web-vitals` library for details.

Analyzing and reporting on FID data

Due to the expected variance in FID values, it's critical that when reporting on FID you look at the distribution of values and focus on the higher percentiles.

While choice of percentile (/articles/defining-core-web-vitals-thresholds#choice_of_percentile) for all Core Web Vitals thresholds is the 75th, for FID in particular we still strongly recommend looking at the 95th–99th percentiles, as those will correspond to the particularly bad first experiences users are having with your site. And it will show you the areas that need the most improvement.

This is true even if you segment your reports by device category or type. For example, if you run separate reports for desktop and mobile, the FID value you care most about on desktop should be the 95th–99th percentile of desktop users, and the FID value you care about most on mobile should be the 95th–99th percentile of mobile users.

How to improve FID

A full guide on optimizing FID (/articles/optimize-fid) is available to guide you through techniques to improve this metric.

To help you manage this, all changes to either the implementation or definition of these metrics will be surfaced in this Changelog (http://bit.ly/chrome-speed-metrics-changelog).

If you have feedback for these metrics, you can provide it in the web-vitals-feedback Google group (https://groups.google.com/g/web-vitals-feedback).

Last updated 2023-05-10 UTC.