First Input Delay

By Philip Walton

Engineer at Google working on the Web Platform

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 measuring how much users like a site's design is hard to do with web APIs, measuring its speed and responsiveness is not!

The first impression users have of how fast your site loads can be measured with metrics like First Paint (FP) and First Contentful Paint (FCP). But how fast your site can paint pixels to the screen is just part of the story. Equally important is how responsive your site is when users try to interact with those pixels!

To help measure your user's first impression of your site's interactivity and responsiveness, we're introducing a new metric called First Input Delay.

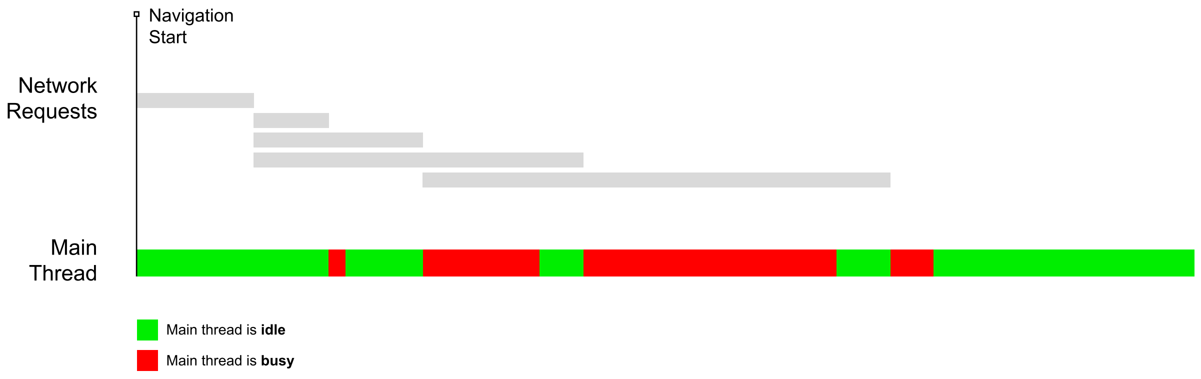
What is first input delay?

First Input Delay (FID) measures the time from when a user first interacts with your site (i.e. 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 respond to that interaction.

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.

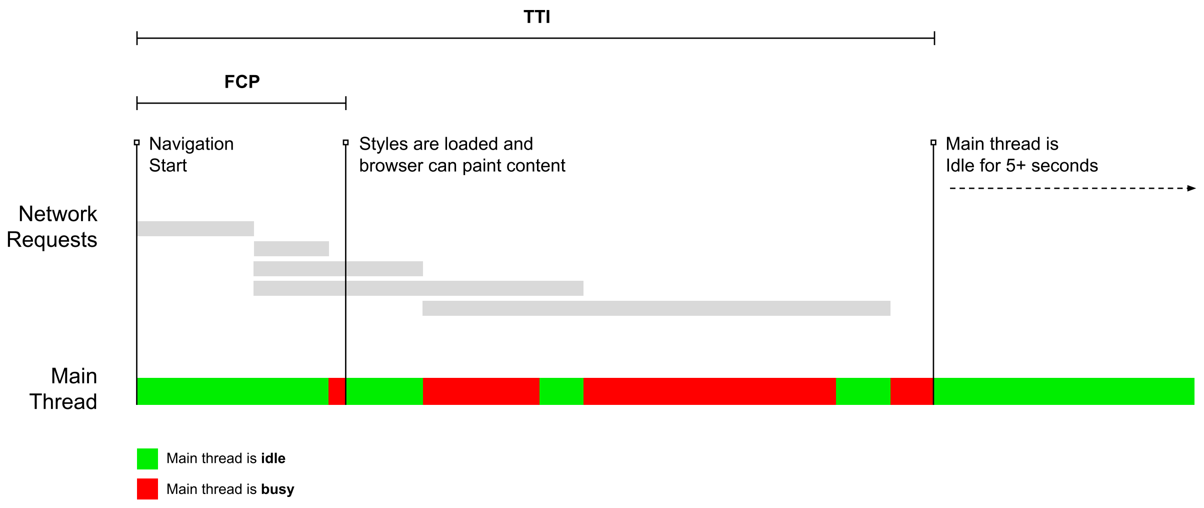
In general, input delay (or input latency) happens because the browser's main thread is busy doing something else, so it can't (yet) respond to the user. One common reason this might happen is the browser is busy parsing and executing a large JavaScript file loaded by your app. While it's doing that, it can't run any event listeners because the JavaScript it's loading might tell it to do something else.

Consider the following timeline of a typical web page load:



The above chart 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 being processed on the main thread. This results in periods where the main thread is momentarily busy (which is indicated by the red color on the chart).

Now let's add two other metrics: First Contentful Paint (FCP), which I mentioned above, and Time to Interactive (TTI), which you've probably seen in tools like Lighthouse or WebPageTest:

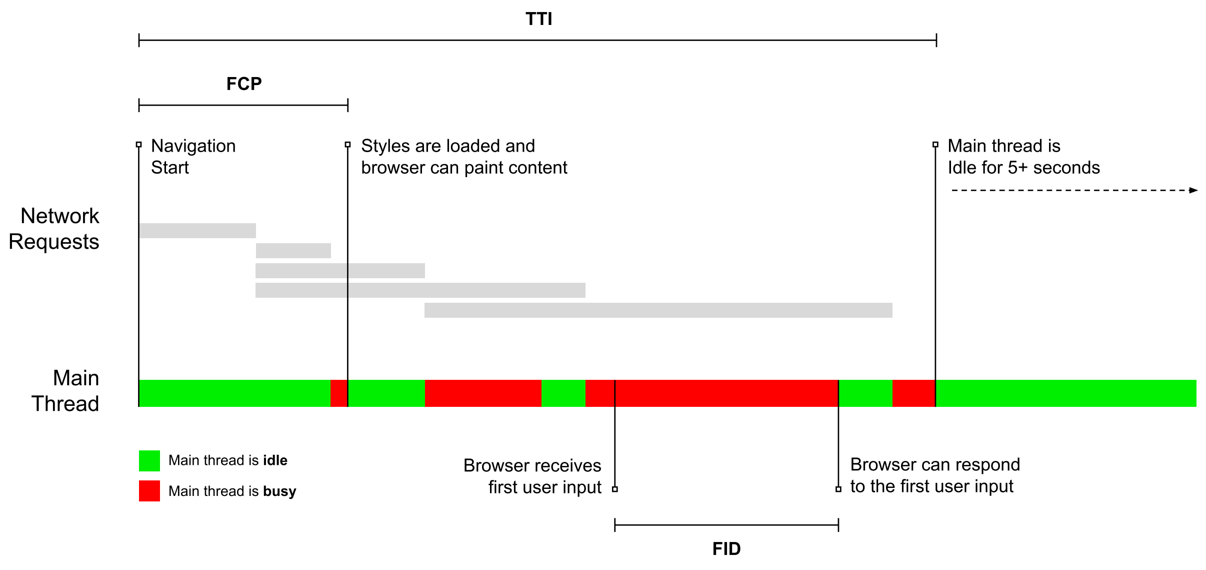


As you can see, FCP measures the time from Navigation Start until the browser paints content to the screen (in this case not until after the stylesheets are downloaded and processed). And TTI measures the time from Navigation Start until the page's resources are loaded and the main thread is idle (for at least 5 seconds).

But you might have noticed that there's a fair amount of time between when content first paints to the screen and when the browser's main thread is consistently idle and thus reliably capable of responding quickly to user input.

If a user tries to interact with the page during that time (e.g. click on a link), there will likely be a delay between when the click happens and when the main thread is able to respond.

Let's add FID to the chart, so you can see what that might look like:


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

Here, the browser receives the input when the main thread is busy, so it has to wait until it's not busy to 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 delays 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.

Why do we need another interactivity metric?

Time to interactive (TTI) is a metric that measures how long it takes your app to load and become capable of quickly responding to user interactions, and First Input Delay (FID) is a metric that measures the delay that users experience when they interact with the page while it's not yet interactive.

So why do we need two metrics that measure similar things? The answer is that both metrics are important, but they're important in different contexts.

TTI is a metric that can be measured without any users present, which means it's ideal for lab environments like Lighthouse or WebPageTest. Unfortunately, lab metrics, by their very nature, cannot measure real user pain.

FID, on the other hand, directly represents user pain—every single FID measurement is an instance of a user having to wait for the browser to respond to an event. And when that wait time is long, users will get frustrated and often leave.

For these reasons we recommend both metrics, but we recommend you measure TTI in lab and you measure FID in the wild, with your analytics tool.

In fact, we plan to do the same with our performance tooling here at Google. Our lab tools like Lighthouse and WebPageTest already report TTI and we're exploring adding FID to our Real User Monitoring (RUM) tools like the Chrome User Experience Report (CrUX).

Also, while these *are* different metrics, our research has found that they correlate well with each other, meaning any work you do to improve your TTI will likely improve your FID as well.

Note: while TTI and FID do correlate well, there can be cases where a site with good TTI scores may have bad FID scores (and vice versa). For example, a site might have a very high TTI, but if most users don't attempt to interact with the site until it's fully loaded, those users won't experience the pain of the high TTI. On the other hand a site might have a relatively low TTI, but if the site looks interactive very early on, users may try to interact with it only to find that nothing happens. This is why we recommend optimizing for both metrics (as well as using both lab and RUM tools).

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:

1. 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.

2. 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.
3. 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?

First Input Delay 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, 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 interactions will be at good times (when the main thread is completely idle).

This means some users will have no FID values, some users will have low FID values, and some users will probably have high FID values.

How you track, report on, and analyze FID will probably be quite a bit different from other metrics you may be used to. The next section explains how best to do this.

Tracking FID in JavaScript

FID can be measured in JavaScript in all browsers today. To make it easy, we've even created a JavaScript library that tracks and calculates it for you: GoogleChromeLabs/first-input-delay.

Refer to the library's README for full usage and installation instructions, but the gist is you:

1. Include a minified snippet of code in the **<head>** of your document that adds the relevant event listeners (this code must be added as early as possible or you may miss events).
2. In your application code, register a callback that will get invoked with the FID value (as well as the event itself) as soon as the first relevant input is detected.

Once you have your FID value, you can send it to whatever analytics tool you use. For example, with Google Analytics, your code might look something like this:

```
// The perfMetrics object is created by the code that goes in <head>.
perfMetrics.onFirstInputDelay(function(delay, evt) {
  ga('send', 'event', {
    eventCategory: 'Perf Metrics',
    eventAction: 'first-input-delay',
    eventLabel: evt.type,
    // Event values must be an integer.
    eventValue: Math.round(delay),
    // Exclude this event from bounce rate calculations.
    nonInteraction: true,
  });
});
```

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. In fact, we recommend specifically focusing on the 99th percentile, as that 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 99th percentile of desktop users, and the FID value you care about most on mobile should be the 99th percentile of mobile users.

Unfortunately, many analytics tools do not support reporting on data at specific quantiles without custom configuration and manual data processing/analysis.

For example, it's possible to report on specific quantiles in Google Analytics, but it takes a little extra work. In my article The Google Analytics Setup I Use on Every Site I Build, I have a section on setting hit-level dimensions. When you do this you unlock the ability to filter and segment by individual event values as well as create distributions, so you can calculate the 99th percentile. If you want to track FID with Google Analytics, I'd definitely recommend this approach.

Who should be tracking FID?

FID is a metric that any site could benefit from tracking, but there are a few types of sites that I think could particularly benefit from knowing what kinds of first input delays their users are actually experiencing:

Server-side rendered (SSR) JavaScript apps

Sites that send a server-rendered version of their page to the client along with a lot of JavaScript that needs to get loaded, parsed, and executed before the page is interactive are particularly susceptible to high FID values.

The reason is because the time between when they *look* interactive and when they actually *are* interactive is often large, especially on low-end devices that take longer to parse and execute JavaScript.

To be clear, it's absolutely possible to build a server-side rendered app that *also* gets interactive quickly. SSR in and of itself is not a bad pattern; the problem occurs when developers optimize for a fast first paint and then ignore interactivity. This is where tracking FID can be particularly eye-opening!

Sites with lots of third-party iframes

Third-party ads and social widgets have a history of not being particularly considerate of their host pages. They tend to run expensive DOM operations on the host page's main thread with no regard for how it will affect the user.

Also, third-party iframes can change their code at any time, so regressions can happen even if your application code doesn't change. Tracking FID on your production site can alert you to problems like this that might not get caught during your release process.

The future

First Input Delay is a brand new metric we're experimenting with on the Chrome team. It's particularly exciting to me because it's the first metric we've introduced that directly corresponds to the pain users experience with real-life interactions on the web.

Going forward we hope to standardize this metric within the W3C WebPerf Working Group, so it can be more easily accessed by asynchronously loaded JavaScript and third-party analytics tools (since right now it requires developers to add synchronous code to the head of their pages).

If you have feedback on the metric or the current implementation, we'd love to hear it! Please file issues or submit pull requests on GitHub.
