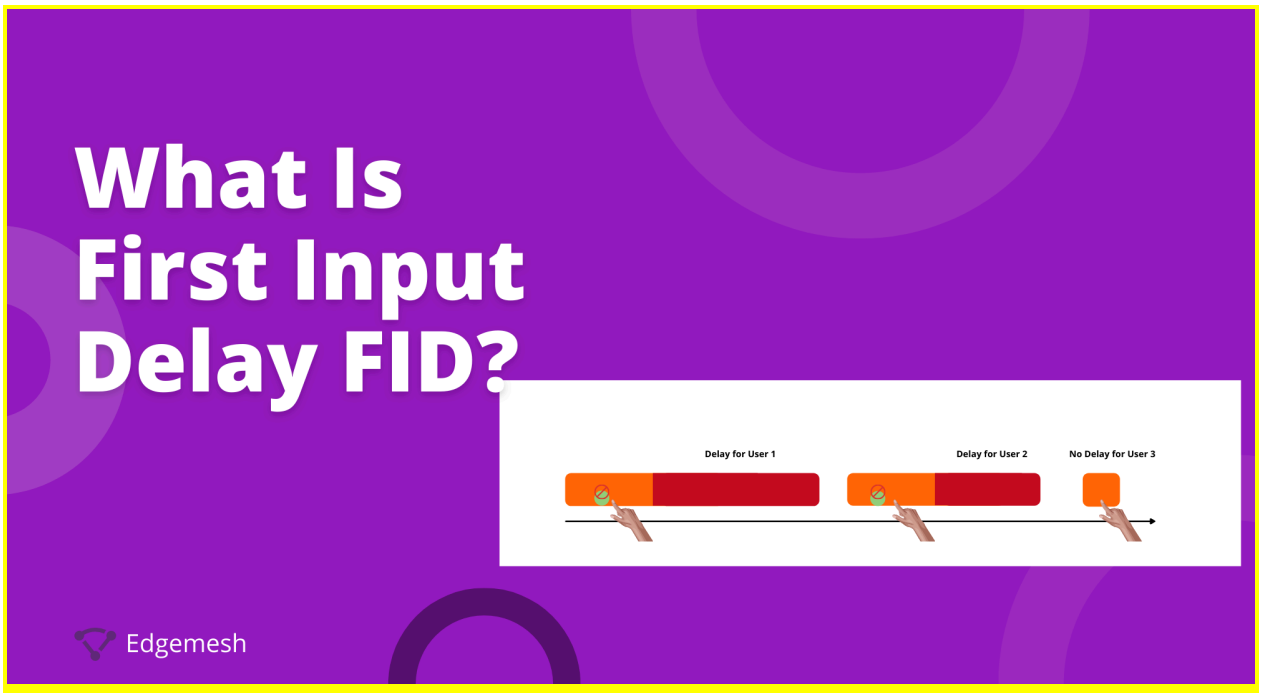
What Is First Input Delay FID)?

The Complete Guide to Google's Core Web Vitals



Have you ever visited a website and when you clicked on a link or tapped a button, it didn't respond fast enough?

Or at the point of your first interaction, the whole page reloads?

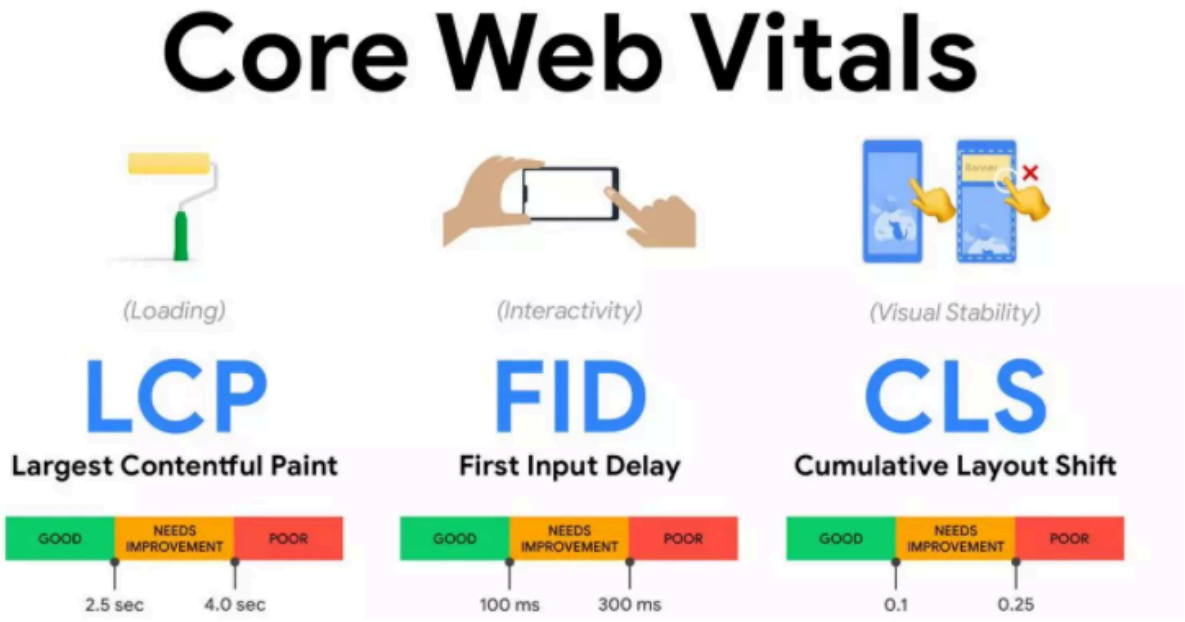
Did the website break? Is it frozen?

And several other questions running through your mind as you're left puzzled about what went wrong.

Most likely, you'd leave the website because it ruined your experience.

This delay while interacting with a website is called the **First Input Delay (FID)**.

The FID is one of the three *(Largest Contentful Paint (LCP) and Cumulative Layout Shift (CLS))* Core Web Vitals introduced by Google as metrics for website performance based on user interaction.



Previously, these metrics focused on user experience but starting mid-June 2021, they've stretched further to influence the rankings of websites.

This means, if your website is not providing the best experience to users, your rankings on Google SERP will be affected.

In this article, we'll cover all you need to know about FID, and how it can be improved to make your website more interactive.

Contents

What is First Input Delay (FID)?

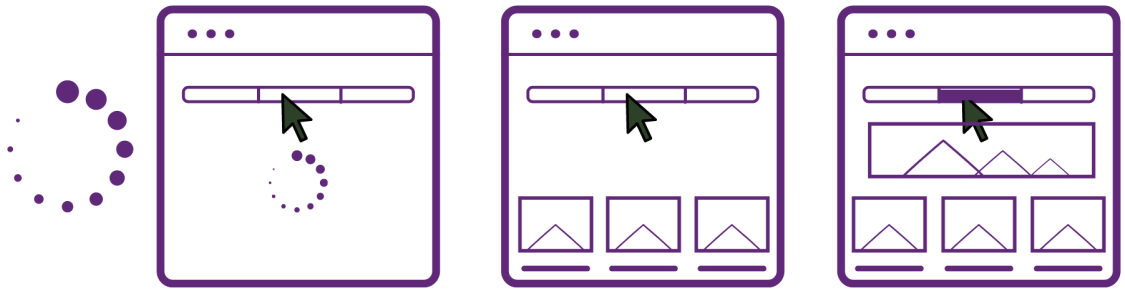
First Input Delay (FID), is a Core Web Vital user-centric metric that measures the time from when a user first interacts with a website (*i.e., tap on a button, or click on a link*) to the time when the browser is able to respond to that interaction.

In simpler terms, FID measures the time it takes for the browser to respond to your first interaction on a website — and it's measured in milliseconds.

***Side Note**:*

- *Scrolling and Zooming are not included in this metric.*
- *FID only measures the first interaction.*

A quick example of how FID works:



The webpage Start loading

The user **taps the sidebar** to access other website content.

The browser disregards the initial user interaction and is still **busy loading a large element**

Once the **large element is loaded.** This is only when the browser begins processing the **interaction with the sidebar**

Edgemesh

The primary concept of FID in the Core Web Vitals' big 3 is based on "*first impressions.*" How your website responds to a first-time user's interaction is the baseline for how it's perceived in terms of performance.

The big question is:

Why is FID only measured after the first input?

To put it simply, an interaction with a web page is what causes FID. If there's no interaction on the web page, there won't be any FID to measure. This is the main reason FID can't be measured on every page of a website.

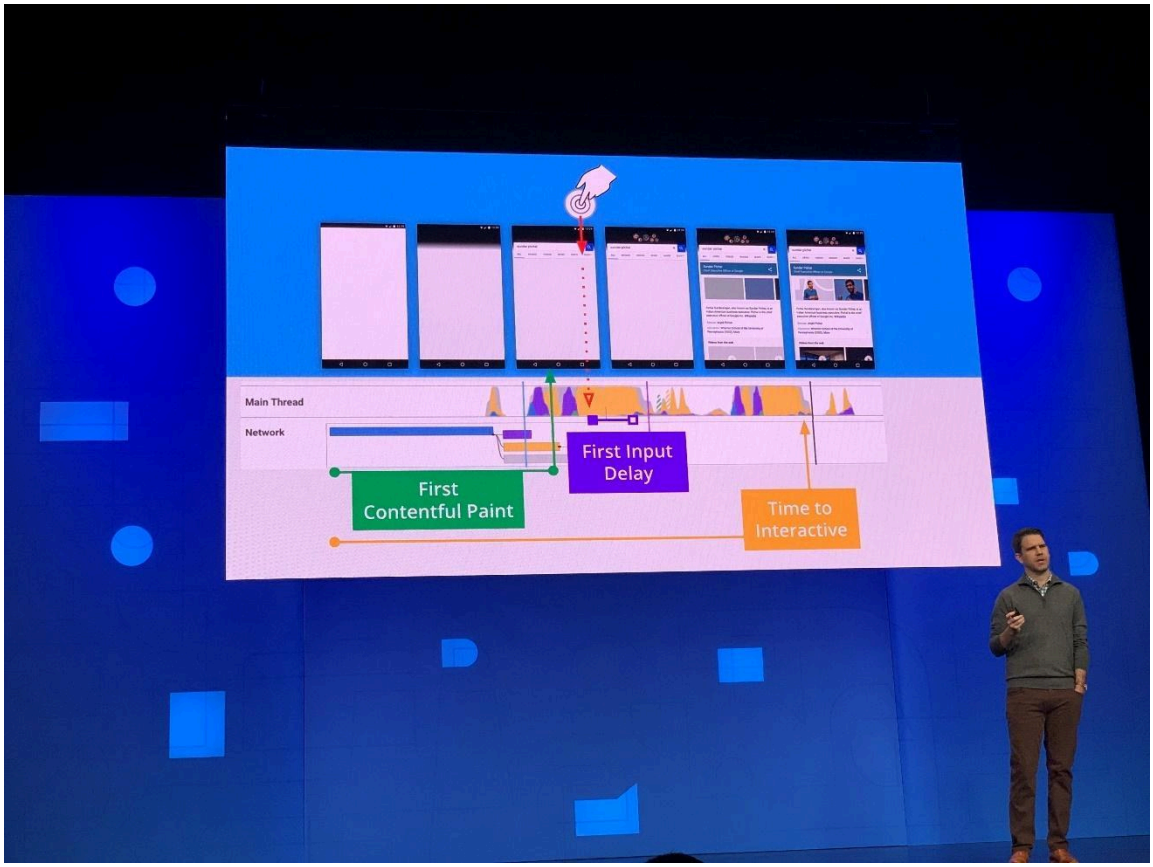
For example, when you visit some websites, you don't interact with any functions of the website —rather, you scroll through, look for what you came for, and press the back button.

On a web page like that, there won't be any FID recorded.

Meanwhile, when you take a look at the time between the first contentful paint and the responsiveness of site functionalities due to a user's interaction, you'd notice disparities.

These disparities are often based on the complexity and size of the JavaScript needing to be downloaded, parsed, and executed.

Paul Irish, at the Chrome Dev summit 2018 —while explaining the user interaction phase with a web page, showed that FID is more of an intersection between FCP (First Contentful Paint), and TTI (Time to Interactive).



But there's more to how FID is being measured. For the most part, delays taking longer than usual typically occur between the time it takes the page to render (FCP), and when it's ready for interaction (TTI).

Let's go into detail on the relation of FID with TTI and FCP:

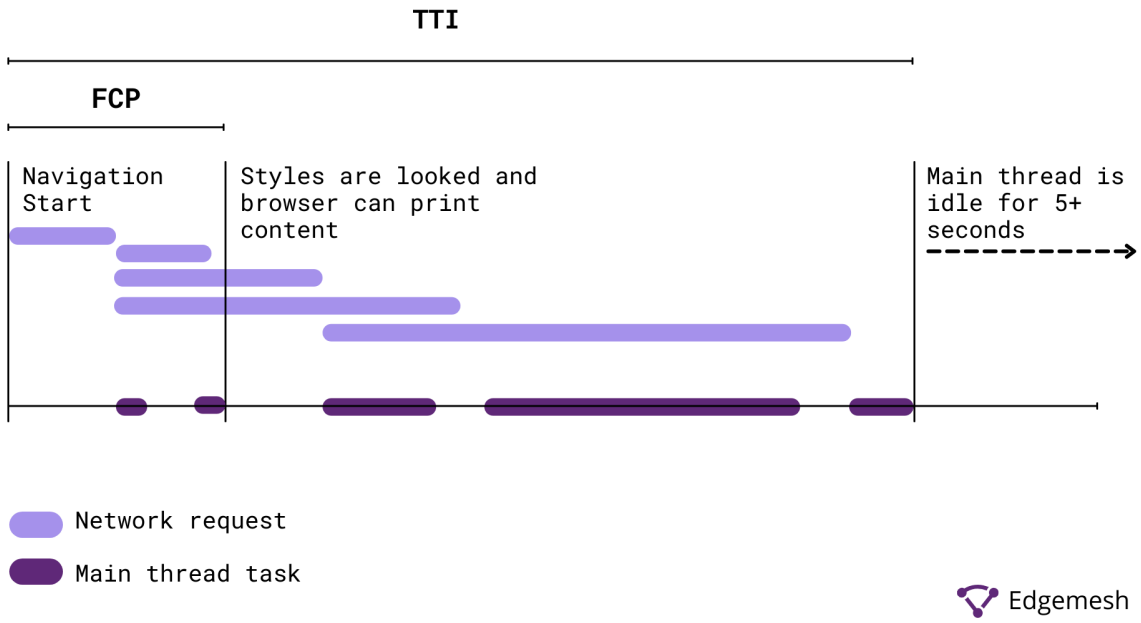
How First Input Delay Works

The concept of a website loading structure is that a typical web page loads in parts —not in full. The response from the server is based on the browser's request —and the user initiates this request on that web page.

As a website loads, the interaction between the browser and the server allows for HTML codes, stylesheets (CSS), and JavaScript files on the web page to load, parse, and render for the user.

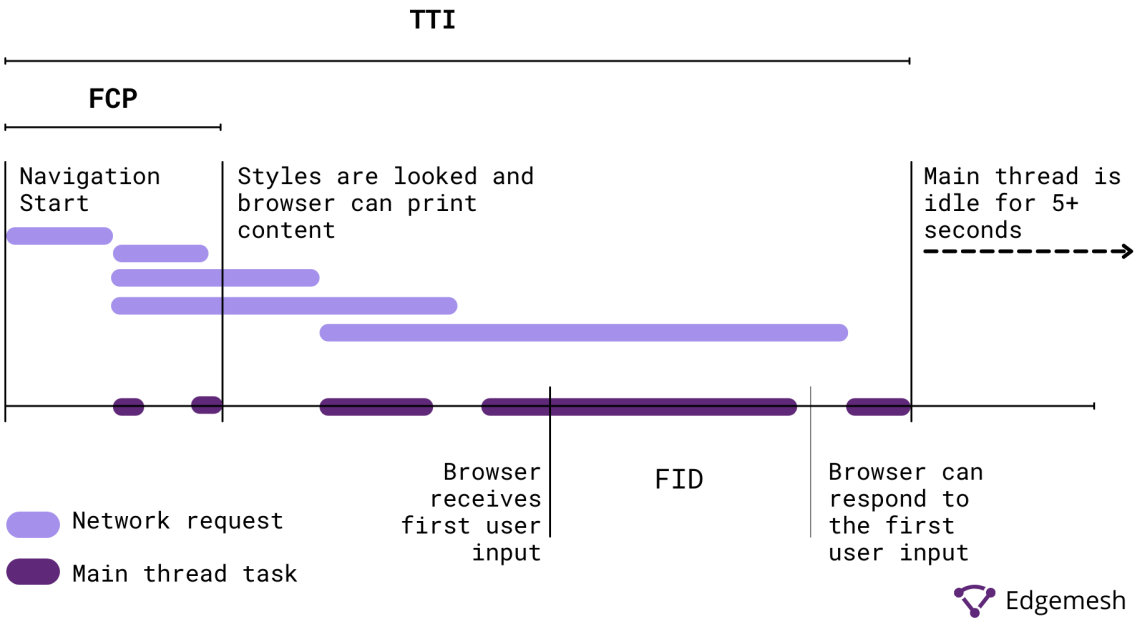
In the process of all these tasks going on, the browser's main thread sometimes gets blocked — making inputs from a user unresponsive until the said task is finished or cleared.

Here's how the whole process goes down:



In the above example, a series of JavaScript tasks (marked in red) between the FCP and TTI block the main thread — making user input on the browser unresponsive.

If a user tries to interact with the page as it is currently processing the uncleared JavaScript task, here's what happens:



As the user tries to interact, the browser receives the input, but it's not processed until the current task blocking the main thread is cleared.

After the task is cleared, the input from the user can be processed. The period between when there's an input and a response to that input is where FID comes in.

Key Note

Let's refer back to another example.

This time, the user interacts with the page early on (idle period) in the process where there was no JavaScript blocking the main thread.

Without this blockage, even if the page isn't completely loaded, the browser will respond to the user's input.

5 Commonly Asked Questions On FID

1. What happens if the user never interacts with the website?

Regardless of your site's traffic or bounce rate, the fact is that not all who visit will interact with it.

A percentage of users who visit your website would only scroll or zoom on certain elements —and as we discussed earlier, these actions don't count as FID.

Still, the interactions vary for users depending on the task at hand.

— For some, their first interaction will be at a bad time —when the main thread is busy with the task it has yet to execute.

— For others, their first interaction will be at a good time — when the main thread is at its idle period.

These variations in response time to interaction mean that different users will experience zero, low, or high FID values for the same page.

Thus, the analysis of the FID results is different from the other two Web Core Vital Metrics (LCP and CLS). *Later in this article, it will be discussed in detail*.

2. Why is only the Input Delay measured and not the whole page loading event?

Simple answer — FID is a metric solely focused on the "response delay" of the browser to a user's first input on a web page.

FID does not measure the time it takes an event to complete on a web page — nor does it measure the time taken for the browser to update the UI after completing the tasks on the main thread.

Though this event time isn't measured by FID, it is important as it affects user experience. Nonetheless, taking this into account to modify the metric could make the web experience even worse for users.

Modifying the metric will make developers create a workaround that leads to excessive use of asynchronous callback via `setTimeout()` or `requestAnimationFrame()`) to separate it from the task associated with the event.

This will result in a drastic improvement in the metric score, but also lead to a slower response which affects the overall user experience.

3. What if an interaction from a user doesn't have an event listener?

The parameter used for FID primarily measures the time between when an input event is received from a user and the main thread's next idle period.

This means, in cases where an event listener has not been registered on the main thread, FID is still measured.

This is because, in a web page loading event, many interactions from a user don't require an event listener to run —instead, they require the main thread to be idle for them to run.

A few examples are these HTML elements that require the tasks on the main thread to be complete before they can run:

- Anchor links (<a>)
- Dropdowns (<select>)
- Checkboxes, text fields, and radio buttons (<input>, (<textarea>)

4. Why is the first input only the one considered?

The first delay is considered the first impression for users on a website and it directly contributes to whether users will bounce from that website based on their experience.

Typically, any input delay at any point can result in a poor user experience.

But when it comes to users' perception of a website, its responsiveness to their **first input** creates an impression around the reliability, structure, and quality of the website.

5. What counts as a first input, and what doesn't?

First Input Delay is based on the metric that measures a web page's responsiveness to a user interaction **during load**.

Due to this, what counts are measurements and tracking on actions such as taps, clicks, and key presses.

What doesn't count are continuous actions such as scrolls and zooms on a web page.

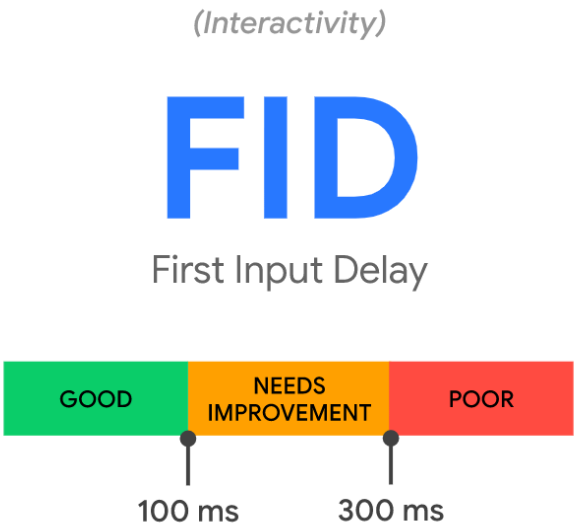
With all these questions answered, the next discussion is "**what is considered a good FID score?**".

What Is a Good FID Score?

According to Google's RAIL model for measuring web performance, the respective scores at each point of input delay (*measured in milliseconds*) is split into three categories, namely:

- Good

- Needs Improvement
- Poor



Good

A good FID score is 100 milliseconds or fewer.

Scoring this will show a green block in the report. This indicates that your page is adequately responsive to a user's first input, and also provides an overall good user experience.

Needs Improvement

An FID score ranging between 100 to 300 milliseconds needs an improvement.

The orange block is displayed for a web page whose FID score spans within that range.

This is considered a mid-point between a good FID score and a poor one because users are not entirely enjoying their experience on the page due to the input delay —but it's at an unnoticeable amount.

Poor

Any FID score above 300 milliseconds is poor.

The worst user experience becomes increasingly noticeable when the FID score is way above 300ms. The first input by users gets a great amount of delay before the browser responds —all of which deters the overall user experience on that web page.

When it comes to single-page applications, the FID score is dependent on what is rendering the page.

The rendering of single page applications is divided into two, namely:

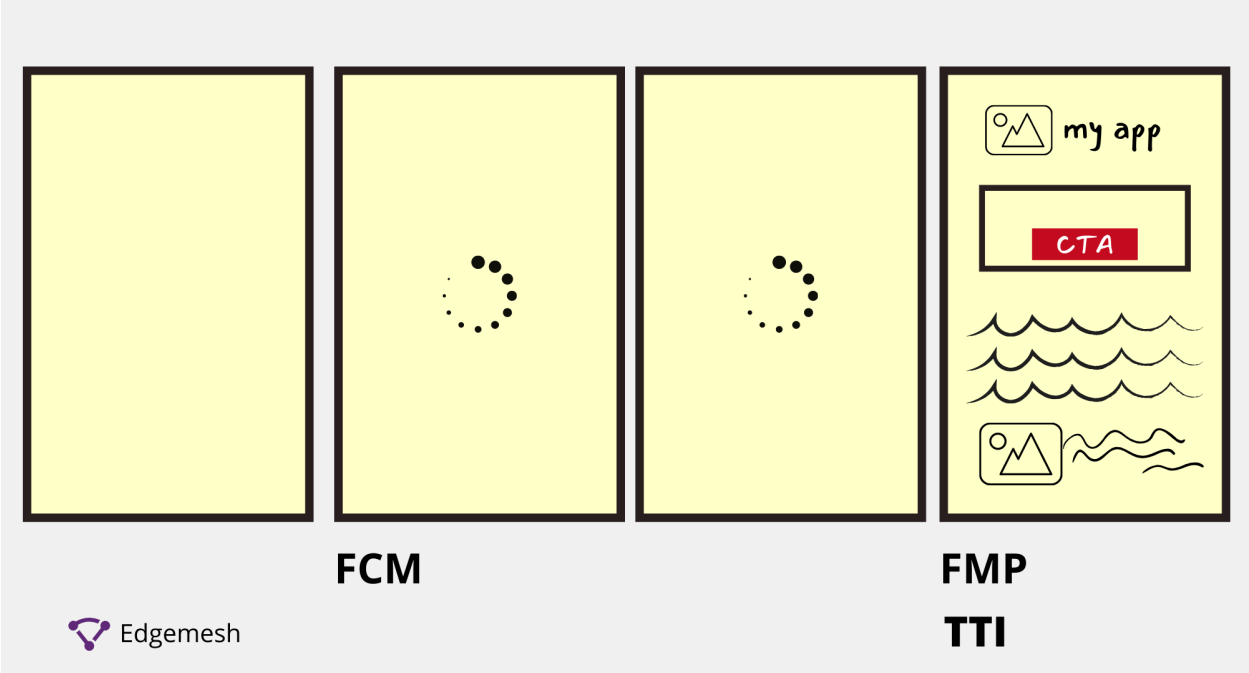
- Client-Rendered Single Page Application
- Server-Rendered Single Page Application

i. Client-Rendered Single Page Application

Single-page applications rendered by the client-side don't typically have a bad or recognizable input delay.

This is because JavaScript controls what the application displays on the page —and due to this, nothing meaningful (*or interactive enough*) will show on the page until the complete application is loaded.

As a result, single page applications like this will have poor FMP (First Meaningful Paint), but a good FID.

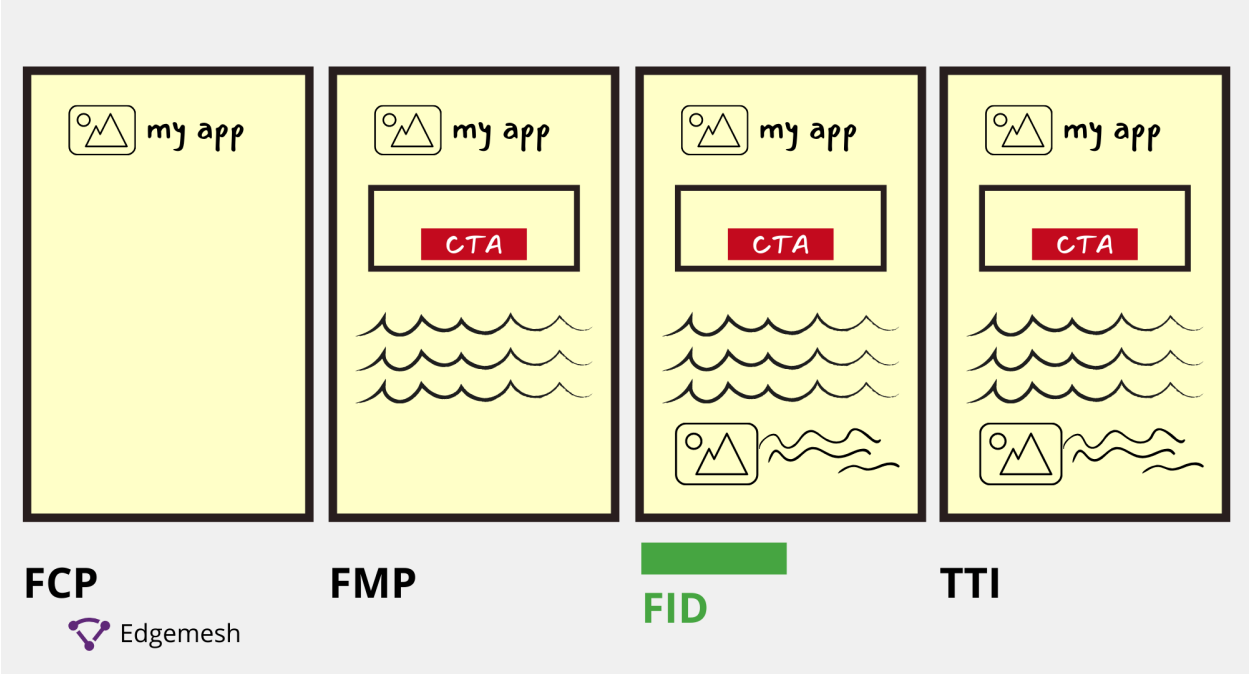


ii. Server-Rendered Single Page Application

It’s the opposite for Single Page Applications rendered by the server-side.

For pages to render in this case, the user requests the server — the server then processes the HTML, CSS, and JavaScript resources exactly as it is demanded and displays them on the page for the user.

Because of this, the First Meaningful Paint is comparatively faster than that of an application rendered on the client-side but results in poor input delays.



Relation of FID with Other Web Metrics

The introduction of web metrics focuses on understanding the users’ experience while browsing on the web.

From landing on a website down to exiting it —there are a series of interactions involved, all of which are measured to estimate a website’s performance in response to those reactions.

In this article, FID relates to three other key web metrics that play a role in its results on a web page.

I. FID and Time to Interactive (TTI)

The relationship between FID and TTI is somewhat close in terms of measuring the interactivity of a web page —but they mean different things.

Time to Interactive (TTI) can be defined as the time taken for a web page to be fully interactive.

Based on this definition, you can see the close similarities in them, and perhaps even think they are the same in some cases, but they aren't.

At any moment, a page can become fully interactive, but users are likely to interact with the page way before it completes its loading event.

TTI falls short in terms of recognizing a user's input before a page is fully interactive —hence creating a gap in the user experience chain.

FID comes to fill this gap by measuring how a web page responds to a user's input **when they decide to interact with it**.

Users don't naturally care how long it takes your web page to become fully interactive. Rather, they decide that themselves by choosing to interact with it at any time.

Upon interacting, users decide whether it's a good or bad experience.

II. FID and First Contentful Paint (FCP)

A user's input is triggered by what piece of content is displayed on the web page— and this is where FID and FCP are connected.

First Contentful Paint (FCP) is a user-centric metric that measures the time it takes the browser to render (display) the first piece of content (*texts, images, animations, or videos*) to a user.

As discussed earlier, first impressions are crucial to the user experience.

The type of content first displayed (FCP) on a web page, to a user, directly correlates to their interaction (FID) — and how fast they both work determines the user's experience.

III. FID and Total Blocking Time (TBT)

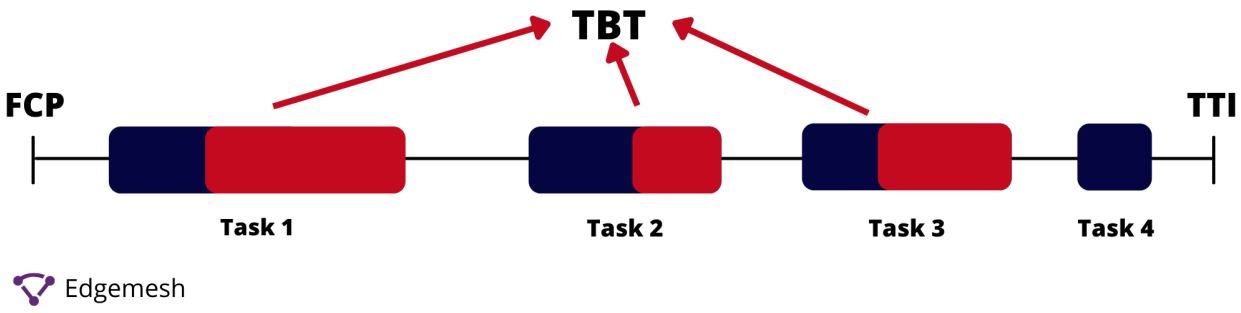
Early on in this article, we discussed how FID is influenced by the tasks on the main thread. This is where TBT comes in.

Total Blocking Time (TBT) is a web performance metric that measures the total amount of time between FCP, and TTI, where the main thread was blocked longer than 50ms to prevent responsiveness to a user's input.

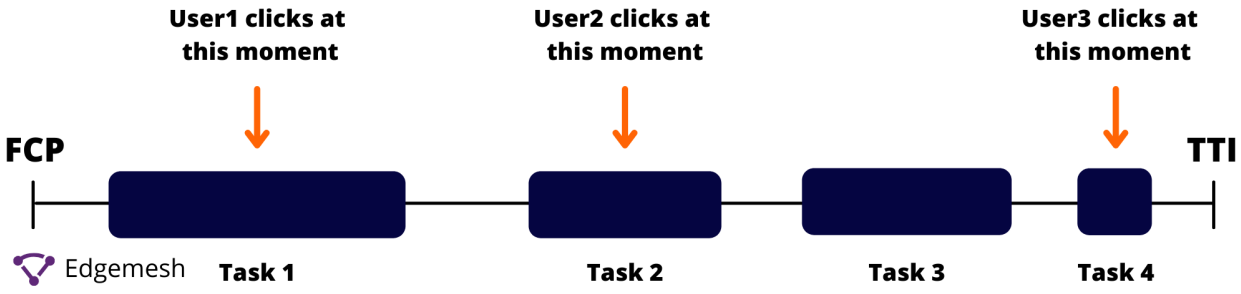
In simpler terms, this metric measures the time long tasks on the main thread take and hinder the usability of a page.

A closer look at the definition and purpose of TBT shows it's easily the same thing that FID measures —but there's a difference.

While TBT measures the unresponsiveness of a web page, it does so **without any input from a user**. Rather, it sums up all long tasks blocking the main thread.

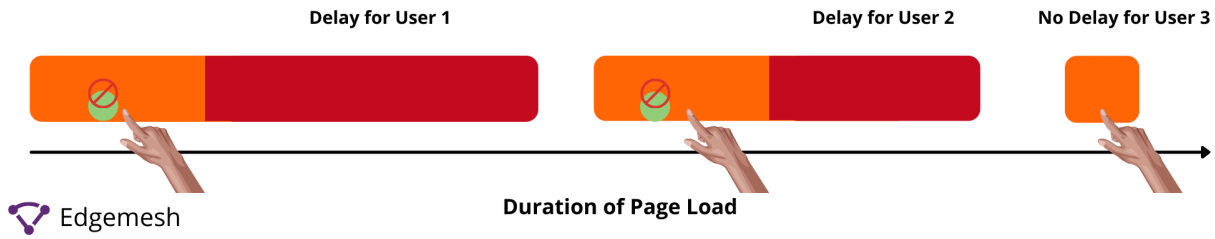


On the other hand, FID measures the unresponsiveness of the page to the user's input at any point between the FCP and TTI.



The FID depends on when different users interact with the website.

Some users scroll through the page before interacting — others interact immediately.



How to Measure First Input Delay (FID)

Measuring FID is different from the other web metrics that can be simulated using lab tools.

Behavioral, network and other on-page actions can be simulated with other metrics, but a user's interaction cannot.

Because FID is a metric that requires a real user's input, it can only be measured on the field using a Field Tool.

Some field tools to do this include:

- PageSpeed Insights
- Search Console (Core Web Vitals Report)
- Chrome User Experience Report
- Web vitals JavaScript library
- Lighthouse in DevTools Using the Total Blocking Feature

In this article, we're going to use PageSpeed Insights and JavaScript to measure FID.

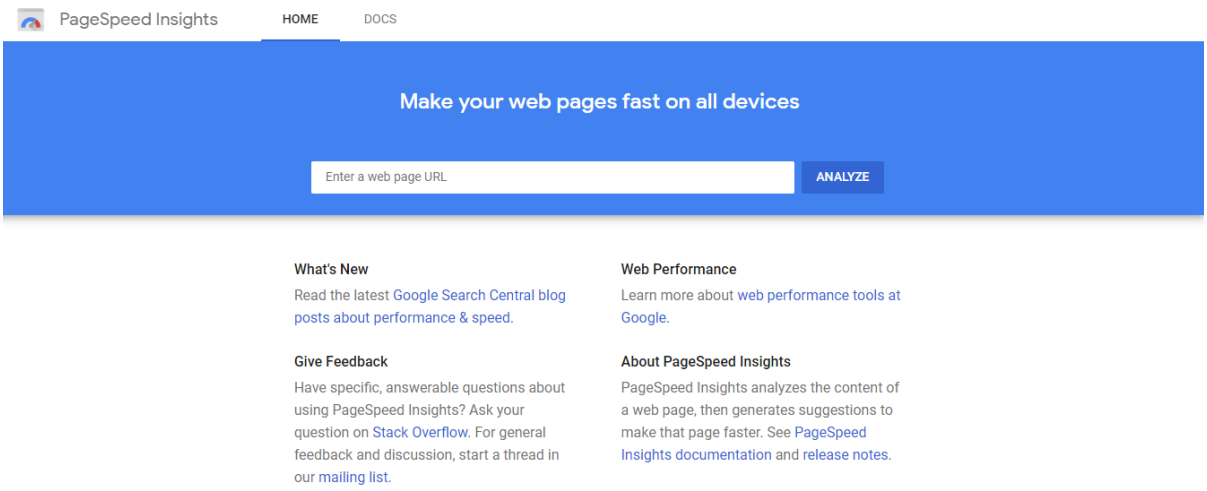
Using PageSpeed Insights to Measure FID

PageSpeed Insights is a performance tool that works both as a field tool and a lab tool.

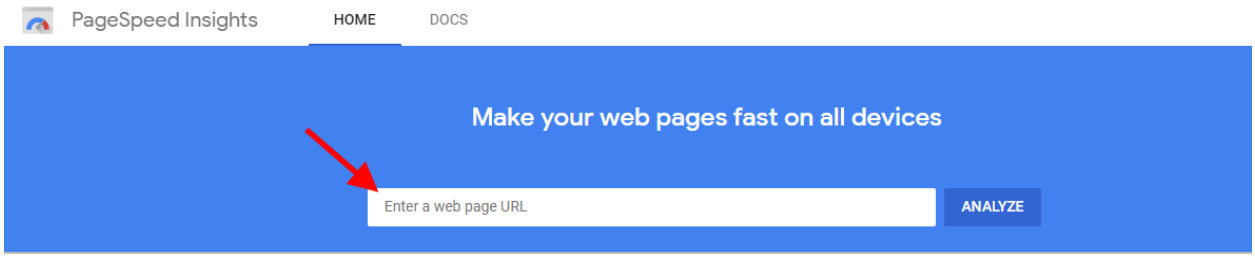
In the case of FID, PageSpeed Insights works well because it runs based on real user interaction data gathered on the website.

The downside to using PageSpeed Insights for measuring FID: It doesn't work on websites without good enough traffic from real users — *you'll see this later*.

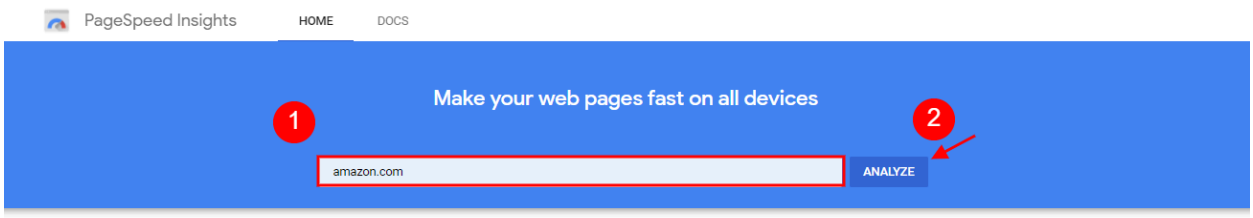
- Head over to [PageSpeed Insights] and you'll see something like this.



- Input your web address — or any functioning website address for which you'd like to know its FID and click on "analyze."



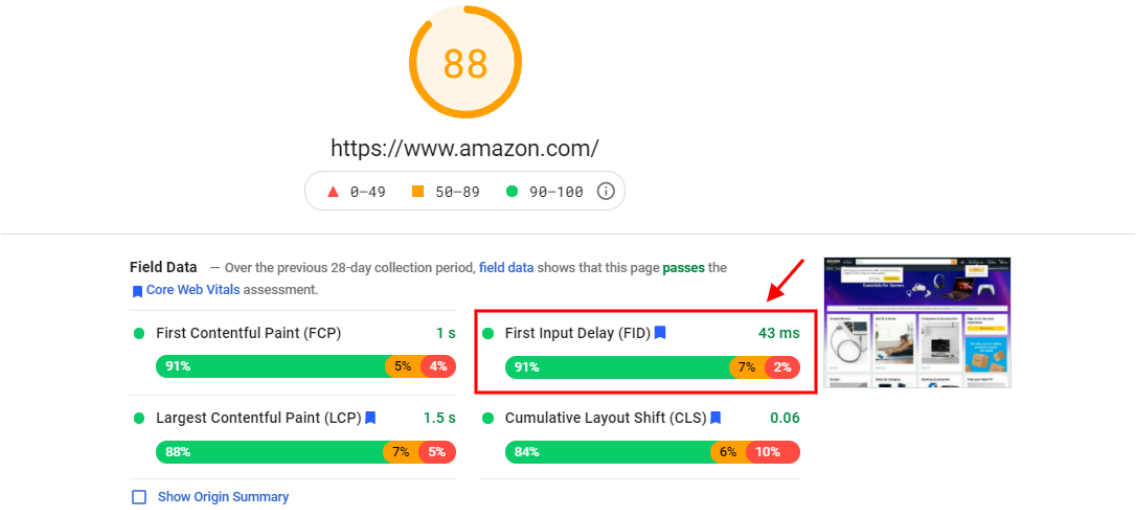
- Let's run a quick FID test using [Amazon.com].



- In using PageSpeed Insights to find your FID, you can toggle between finding the FID of your website while on mobile, desktop, or both.

- This is due to the disparities in interaction on mobile versus desktop.

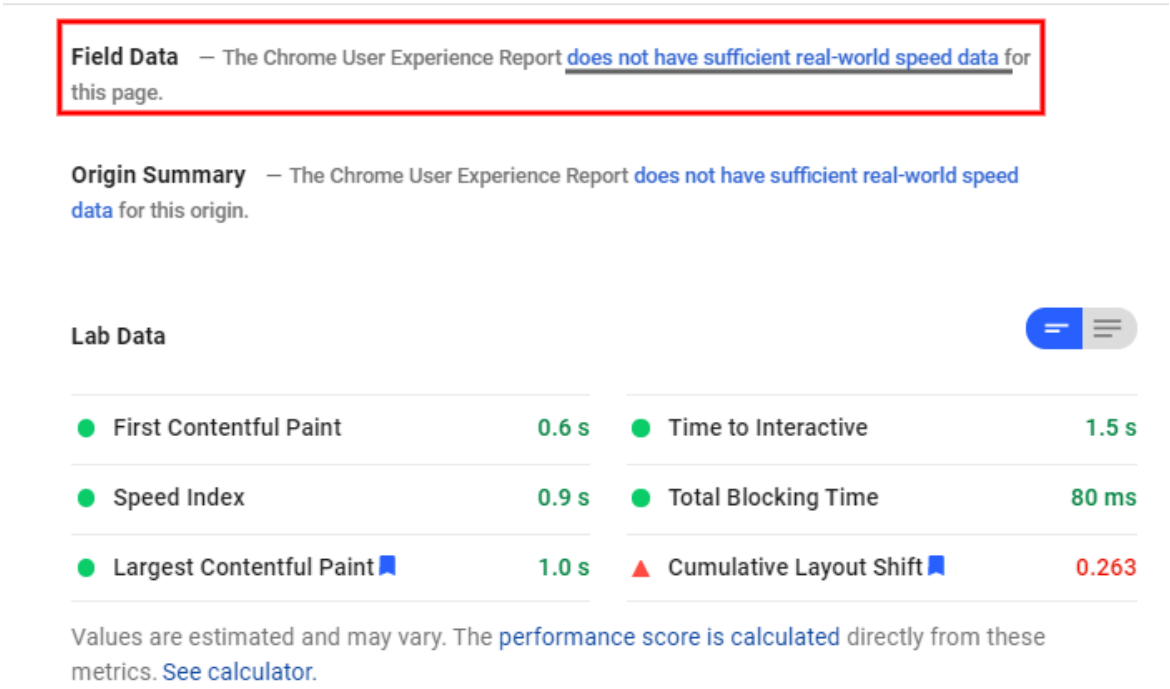
- Scroll down, you'll find the **field data** below —and in there is your First Input Delay score.



- Based on this report, Amazon passes the Core Web Vitals' assessment with an FID score of 43ms (*Google recommends keeping it to under 100ms*).

Let's go back to where PageSpeed Insights falls short.

If you run a website that doesn't have enough interaction from real users, you won't get any FID report on PageSpeed Insights.



Meanwhile, if you're good with programming languages, Philip Whalton of web.dev has made it possible to measure your FID using JavaScript

Using JavaScript to Measure FID

To measure the FID of your web page, you can add JavaScript to the page. There are two ways of going that:

- Web Vitals JavaScript Library
- Adding a Performance Observer Interface

I. Web Vitals JavaScript Library

The web vitals library is a modular library used for measuring all Web Vitals metrics and other related metrics based on interaction from real users.

It measures these metrics in a way that matches how they're measured by Chrome and reported to other Google tools such as *Chrome User Experience Report, PageSpeed Insights, and Search Console's Speed Report*.

To do this, you have to import the library so as to have FID printed out in the console.

Import it using this code:

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

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

Or you can preferably refer back to the source code for `getFID()` to get a complete example of measuring FID in JavaScript.

II. Adding a Performance Observer Interface

Another way to track your FID with JavaScript is to create and add a Performance Observer Interface to your page.

The performance observer interface gathers low-level timing information asynchronously as it's gathered by the browser used.

To create this, you have to use the Event Timing API. This way, the performance observer listens to first-input entries alone and then logs them on the console.

Here's the code for doing that:

```
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});
```

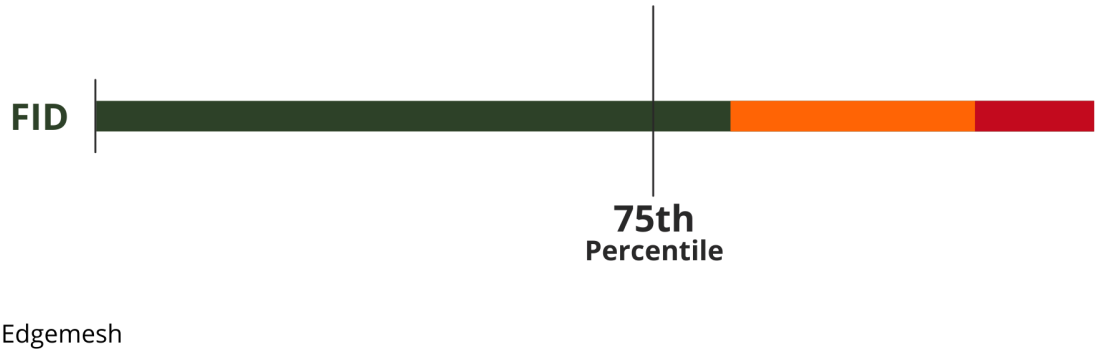
Note: This code only shows you how to log entries from a user's `first-input` to the console and then calculate the delay.

Analyzing FID Data

The values obtained from the FID data have some disparities due to the distribution of user interaction at different time intervals.

This is why Google recommends looking at the 75th percentile to get an estimate of what % of users had what % of delays.

It's this same method used in the field report when measuring for your FID.



A guideline while analyzing is to ensure that at least 75% of users interacting with your website fall within the "green bar."

What Causes Poor First Input Delay?

The delay or latency in the response to a user's input can be attributed to one aspect: the blocking of the main thread by JavaScript elements.

As a result, there are two main causes of poor First Input Delay, namely:

- Large JavaScript Bundles
- Long-running JavaScript Tasks

I. Large JavaScript Bundles

JavaScripts having large bundles take a lot of time for the browser to process, parse, evaluate, and execute due to their size.

This causes a delay in input response because the browser has to first load all JavaScript involved in the loading — regardless of size.

II. Long-running JavaScript Tasks

A poor FID can also result from poorly coded or unoptimized JavaScripts. These JavaScripts, even if completely loaded, won't clear off the main thread to allow other tasks to run — making it a long-running JavaScript task.

With these tasks still on the main thread, user inputs will continue to remain blocked or delayed until it's cleared.

In some cases, depending on how long it takes tasks to clear on the main thread, it'll cause the page to re-render excessively.

So far, we've learned about what FID is, what causes it,and various ways to measure it. Now, let's get into how you can optimize your FID score for a better on-page experience for users.

How to Improve First Input Delay?

After measuring your FID and knowing where you stand on the FID score scale your next step is to improve it.

5 ways you can use to improve your FID:

I. Remove Non-Critical and Unused Third-Party Scripts

Script from third-party apps and web integrations are a major main thread blocker. In some cases, they are likely to load and run before your scripts start executing their tasks.

With this interruption comes a delay in the browser's response rate to user input.

Solve this by first removing all third-party scripts. Then remeasure your FID.

After you get your FID and the score is good, before adding back the third-party scripts, prioritize which is more important to the user and which isn't.

Each time you add a third-party script, check your FID score. This will let you know which script is affecting your FID.

Scripts from ads and pop-ups are the major culprits of poor FID scores; it'll be best to remove them as they don't serve any purpose to the user's experience.

II. Utilize Web Workers

Reducing the workload on the main thread is a great way to get more tasks executed and improve your FID.

With web workers, you can let JavaScript codes run in the background threads without interrupting the operation in the main thread or the user interface.

Web workers can also help you perform I/O using `XMLHttpRequest` (although the `responseXML` and `channel` attributes are always null) or `fetch` (with no such restrictions).

III. Use the "Idle Until Urgent" Approach

The "idle until urgent" approach is an evaluation strategy coined by Philip Walton of web.dev (Google) when he was trying to optimize the FID of his website.

This approach takes into account two of the most popular code evaluation strategies:

- Eager Evaluation
- Lazy Evaluation

i. Eager Evaluation

This strategy lets your code run right away. It results in a longer page load time, but once it's fully interactive, there won't be any input delay.

Though this could help improve your FID, there's a downside.

If there's an interaction from a user while the code is still evaluating, the browser must wait until that evaluation is done before it responds to the user's input.

This becomes a problem for user experience because it's likely to create an impression that your website is bad or of low quality due to bad page load time.

ii. Lazy Evaluation

This strategy does the opposite of eager evaluation. Like the name "lazy" implies, your code only runs when needed.

It helps you avoid running all your codes at once —especially when it's not necessary or requested by the user.

There's also a downside to this strategy. Though your page will load very fast, you run the risk of having an input delay when a user makes a request.

The Idle Until Urgent approach offers the best of both worlds.

With this approach, you're able to run your code during idle periods on the main threads to get the full extent of it.

It guarantees that urgently needed code is run immediately when it's requested.

This way you improve your FID by reducing the Total Blocking Time on the main thread.

To go into full detail on how "Idle Until Urgent" works, read the full article here.

IV. Break-up and Accelerate Long JavaScript Tasks

Users can't interact when a web page is still processing, parsing, or loading its JavaScript tasks on the main thread.

A way to solve that is by breaking up long JavaScript tasks into smaller ones. This method is called "code splitting."

With code splitting, you can break up long JavaScript bundles into smaller bits that can run and execute on the main thread quickly.

By doing this, users' input can easily be processed in-between the loading of the JavaScript tasks.

It's recommended you keep tasks running on the main thread below 50ms to minimize delay and improve overall page speed.

V. Modify the Use of Polyfills

Polyfills serve as a bridge in communicating JavaScript codes made for new browsers work on old browsers.

With Polyfills, your website can run effectively —including all functionalities on both new and old browsers.

However, you shouldn't have to load them unless they are needed. Doing so costs you more milliseconds that results in an input delay.

You should modify their use by delivering different JavaScript bundles based on browser-type requests.

Conclusion

User experience while interacting with your website should be your top priority.

Ensuring your website is fully functional and quickly responsive to users' input — without delay — helps them more effectively interact with you.

Optimizing your FID is a great way to improve the user experience. Doing this trickles down the effects to your bottom line, as you'll increase conversions and reduce bounce rates.

Now that you know all about FID, and how improving it can help your business bottom line —my next question is:

What's your current FID score, and what part of website optimization do you need help with?

Opt-in for a free 14-day trial to get a feel for what speed means to your business.