Total Blocking Time (TBT)

Philip Walton

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

Note: Total Blocking Time (TBT) is an important lab metric (/articles/user-centric-performance-metrics#in_the_lab) for measuring load responsiveness (/articles/user-centric-performance-metrics#types_of_metrics) because it helps quantify the severity of how non-interactive a page is prior to it becoming reliably interactive—a low TBT helps ensure that the page is usable (/articles/user-centric-performance-metrics).

What is TBT?

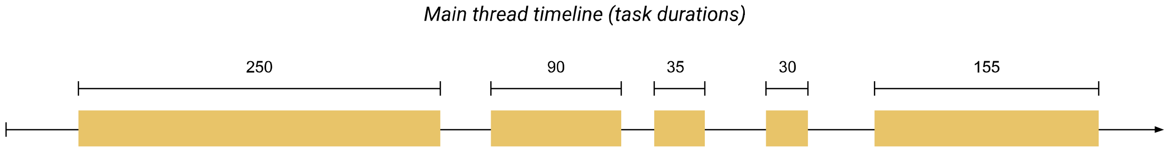
The Total Blocking Time (TBT) metric measures the total amount of time after First Contentful Paint (FCP) (/articles/fcp) where the main thread was blocked for long enough to prevent input responsiveness.

By default, Lighthouse stops monitoring TBT after Time to Interactive (TTI) (/articles/tti), as do some other lab tools that measure page load. See How does TBT relate to TTI? (#how_does_tbt_relate_to_tti).

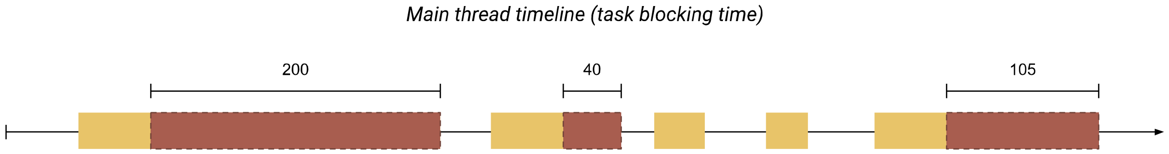
The main thread is considered "blocked" any time there's a Long Task (/articles/custom-metrics#long_tasks_api)—a task that runs on the main thread for more than 50 milliseconds (ms). We say the main thread is "blocked" because the browser cannot interrupt a task that's in progress. So in the event that a user *does* interact with the page in the middle of a

The *blocking time* of a given long task is its duration in excess of 50 ms. And the *total blocking time* for a page is the sum of the *blocking time* for each long task that occurs after FCP for the measured timeframe (typically TTI for page load tools, or the total trace time for other tooling).

For example, consider the following diagram of the browser's main thread during page load:



The above timeline has five tasks, three of which are Long Tasks because their duration exceeds 50 ms. The next diagram shows the blocking time for each of the long tasks:



So while the total time spent running tasks on the main thread is 560 ms, only 345 ms of that time is considered blocking time.

	Task duration	Task blocking time
Task one	250 ms	200 ms
Task two	90 ms	40 ms
Task three	35 ms	0 ms
Task four	30 ms	0 ms
Task five	155 ms	105 ms
Total Blocking Time		**345 ms**

How does TBT relate to TTI?

TTI considers a page "reliably interactive" if the main thread has been free of long tasks for at least five seconds. This means that three, 51 ms tasks spread out over 10 seconds can push back TTI just as far as a single 10-second long task—but those two scenarios would feel very different to a user trying to interact with the page.

In the first case, three, 51 ms tasks would have a TBT of **3 ms**. Whereas a single, 10-second long tasks would have a TBT of **9950 ms**. The larger TBT value in the second case quantifies the worse experience.

This example shows why TBT is often a better metric than TTI as it is less prone to outliers. This is even the case when TTI is used as the endpoint for TBT.

How to measure TBT

TBT is a metric that should be measured in the lab (/articles/user-centric-performance-metrics#in_the_lab). The best way to measure TBT is to run a Lighthouse performance audit on your site. See the Lighthouse documentation on TBT (https://developer.chrome.com/docs/lighthouse/performance/lighthouse-total-blocking-time/) for usage details.

Lab tools

- Lighthouse (https://developer.chrome.com/docs/lighthouse/overview/)
- WebPageTest (https://www.webpagetest.org/)

Note: While it is possible to measure TBT in the field, it's not recommended as user interaction can affect your page's TBT in ways that lead to lots of variance in your reports. To understand a page's interactivity in the field, you should measure First Input Delay (FID) (/articles/fid) and Interaction to Next Paint (INP) (/articles/inp).

What is a good TBT score?

To provide a good user experience, sites should strive to have a Total Blocking Time of less than **200 milliseconds** when tested on **average mobile hardware**.

For details on how your page's TBT affects your Lighthouse performance score, see How

How to improve TBT

To learn how to improve TBT for a specific site, you can run a Lighthouse performance audit and pay attention to any specific opportunities (https://developer.chrome.com/docs/lighthouse/performance/#opportunities) the audit suggests.

To learn how to improve TBT in general (for any site), refer to the following performance guides:

- Reduce the impact of third-party code (https://developer.chrome.com/docs/lighthouse/performance/third-party-summary/)
- Reduce JavaScript execution time (https://developer.chrome.com/docs/lighthouse/performance/bootup-time/)
- Minimize main thread work (https://developer.chrome.com/docs/lighthouse/performance/mainthread-work-breakdown/)
- Keep request counts low and transfer sizes small (https://developer.chrome.com/docs/lighthouse/performance/resource-summary/)

Last updated 2023-11-17 UTC.