

Web Vitals

Essential metrics for a healthy site



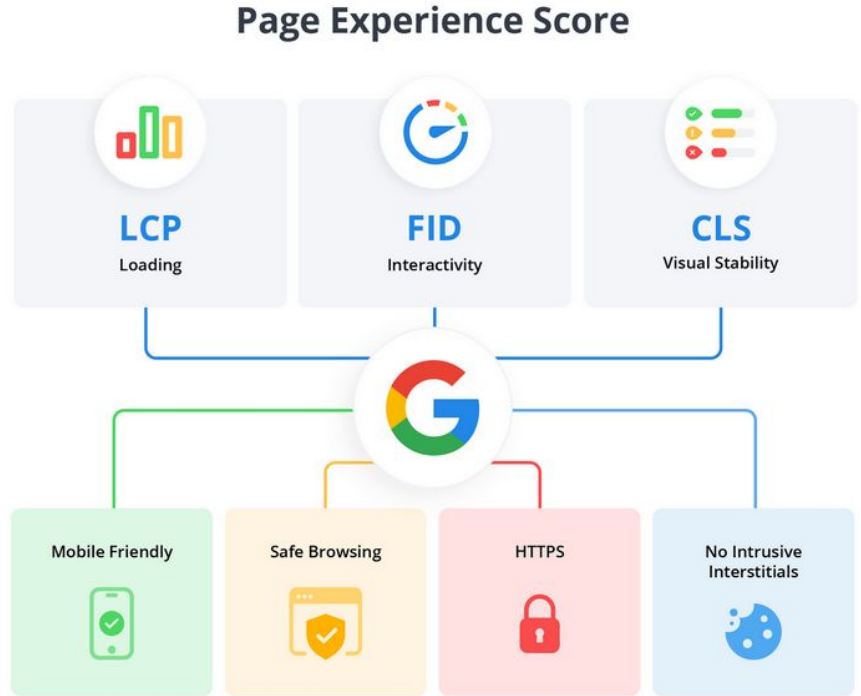
Objectives

- Google's Page Experience
- What is 'Web Vitals'
- Core Web Vitals
 - LCP
 - FID
 - CLS



Google's Page Experience

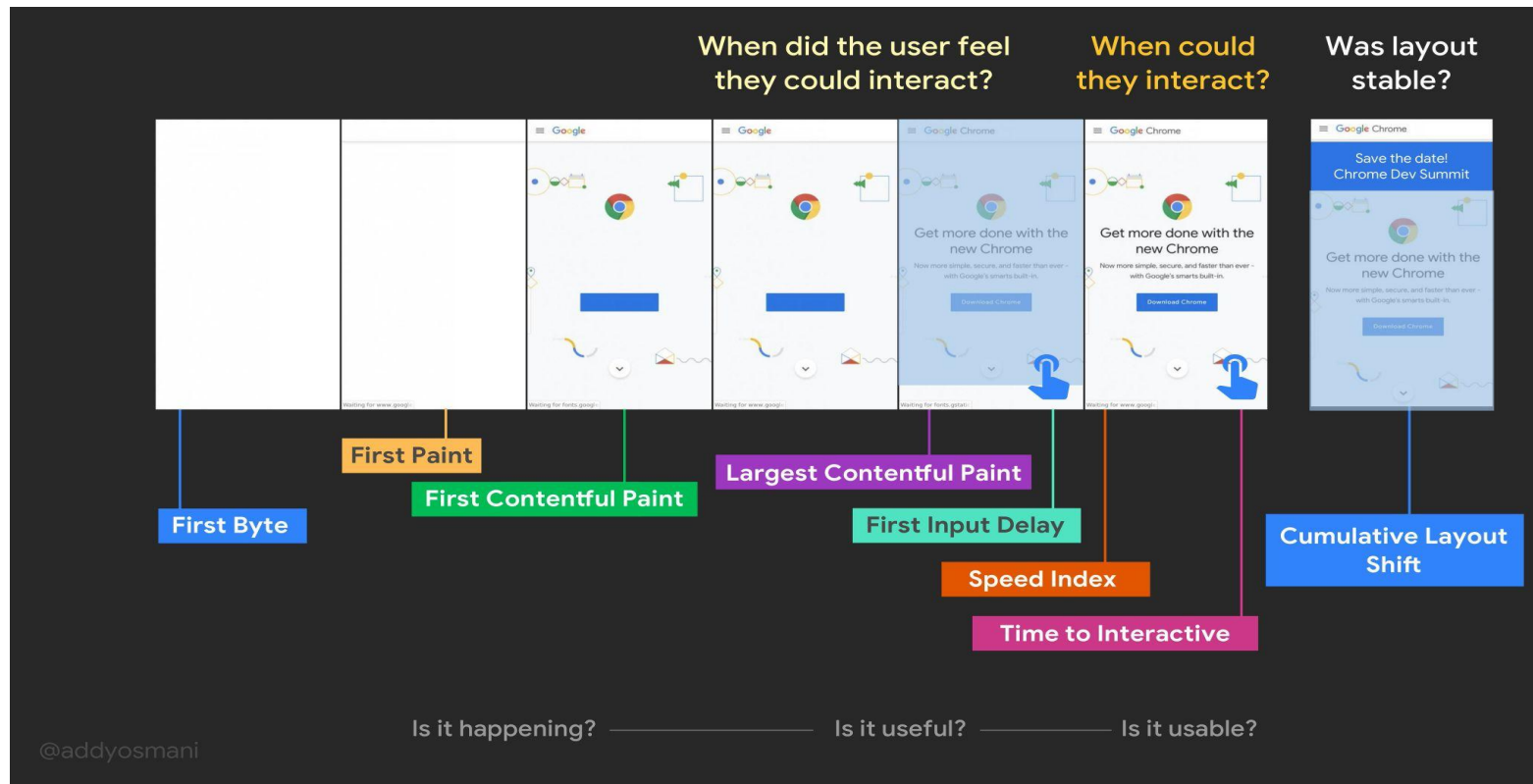
- It's a set of signals that measure how users perceive the experience of interacting with a web page both on mobile and desktop devices.
- It includes :
 - Core Web Vitals
 - Other signals (Boolean checks)



Web Vitals

- An initiative by Google which provides quality signals that are essential to delivering a great user experience on the web.
- This initiative aims to simplify the landscape, and help sites focus on the metrics that matter most, Core Web Vitals.
- It consists of the Core Web Vitals and the non-Core Web Vitals.

Web Vitals Cont.



Core Web Vitals



(Loading)



(Interactivity)



(Visual Stability)

Core Web Vitals



(Loading)

LCP

Largest Contentful Paint



(Interactivity)

FID

First Input Delay



(Visual Stability)

CLS

Cumulative Layout Shift



CLS

Cumulative Layout Shift



- Measures layout stability to ensure user's experience smooth and natural interactions.
- It helps quantify how often users experience unexpected layout shifts—a low CLS helps ensure that the page is delightful.
- $\text{layout shift score} = \text{impact fraction} * \text{distance fraction}$

Problem of Layout Shifting

Order confirmation

You have selected 56 items. Is this correct?

Yes, place my order

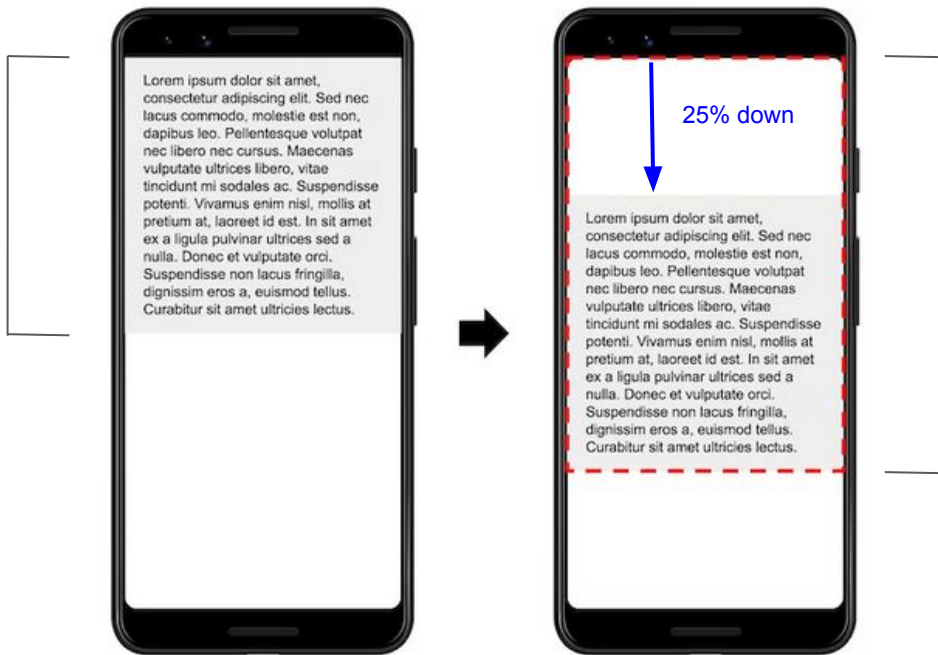
No, go back



Layout Shift Score

$$\text{layout shift score} = 0.75 * 0.25 = 0.1875$$

50% of viewport



Optimize Cumulative Layout Shift

- Always include size attributes on your images and video elements.
- Statically reserve space for the ad slot.
- Avoid placing ads near the top of the viewport.
- Preloading web fonts.
- Use CSS transform property.



Optimize Cumulative Layout Shift

1. Size attributes of images and video elements

- Always include width and height size attributes on your images and video elements.

The following "pixel" dimensions would ensure a 640x360 area would be reserved. The image would stretch to fit this space.

```

```

- When Responsive Web Design was introduced, developers began to omit width and height and started using CSS to resize images instead. Space could only be allocated for an image once it began to download.

```
img {  
  width: 100%; /* or max-width: 100%; */  
  height: auto;  
}
```

Optimize Cumulative Layout Shift

1. Size attributes of images and video elements

- This is where aspect ratio comes in. The aspect ratio of an image is the ratio of its width to its height.

Knowing the aspect ratio allows the browser to calculate and reserve sufficient space the associated area.

- Modern browsers now set the default aspect ratio of images based on an image's width and height attributes.

```
<!-- set a 640:360 i.e a 16:9 - aspect ratio -->  

```

```
img {  
  aspect-ratio: attr(width) / attr(height);  
}
```

Optimize Cumulative Layout Shift

1. Size attributes of images and video elements

- When working with responsive images, srcset defines the images you allow the browser to select between and what size each image is. To ensure width and height attributes can be set, each image should use the same aspect ratio.

```

```

Optimize Cumulative Layout Shift

Size attributes of images and video elements



Optimize Cumulative Layout Shift


2. Statically reserve space for the ad slot

In other words, style the element before the ad tag library loads.


```
<style>
.container {
  display: block;
  width: 720px;
  height: 90px;
  background: #ccc;
  overflow: hidden;
}
</style>

<div class="container">
  <iframe src="...">
</div>
```


Container



Content in Container

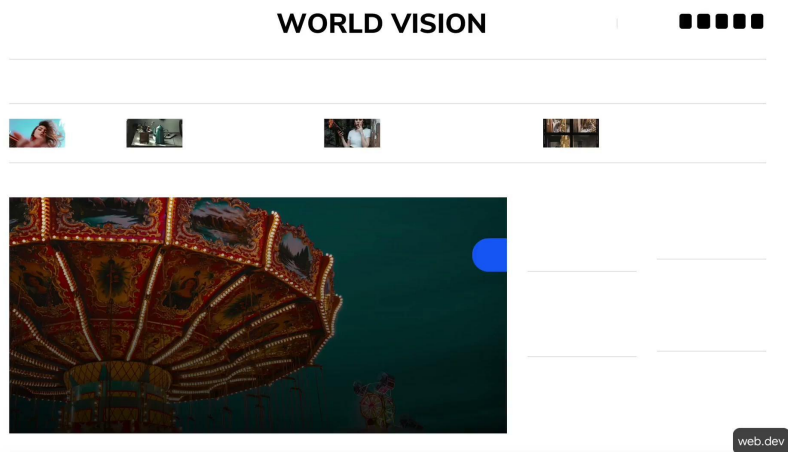


Content in Container - different size

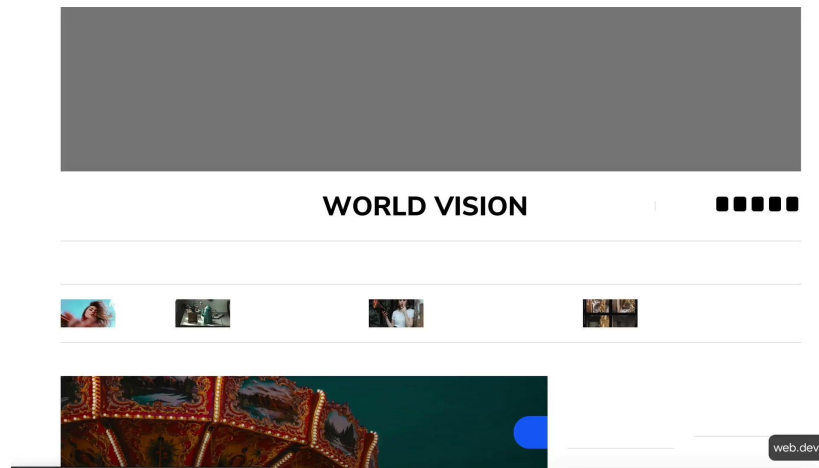


Optimize Cumulative Layout Shift

Statically reserve space for the ad slot



Ads without sufficient space reserved.

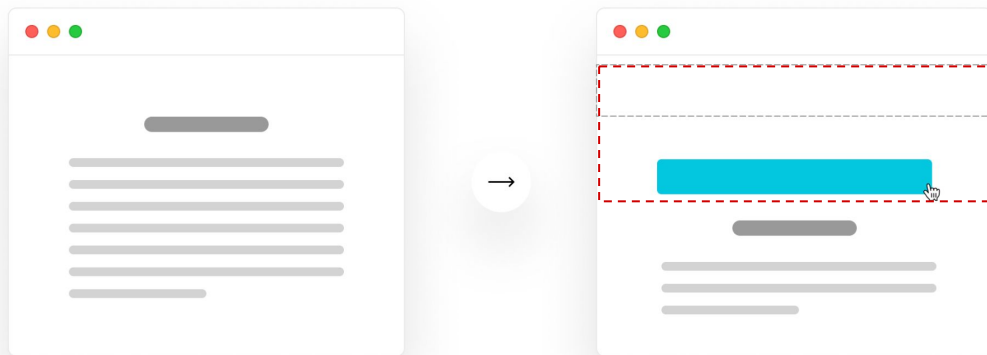


Ads with sufficient space reserved.

Optimize Cumulative Layout Shift

3. Avoid placing ads near the top of the viewport

- Ads near the top of the viewport may cause a greater layout shift than those at the middle.
- This is because ads at the top generally have more content lower down, meaning more elements move when the ad causes a shift



Optimize Cumulative Layout Shift

4. Preloading web fonts

A preloaded font will have a higher chance to meet the first paint, in which case there's no layout shifting and FOIT/FOUT will be prevented.

```
<link rel="preload" href="/webfontname" as="font" type="font/format" crossorigin>
```

Flash of Invisible Text (FOIT)



Flash of Unstyled Text (FOUT)



Optimize Cumulative Layout Shift

5. Use CSS transform property

CSS transform property allows you to animate elements without triggering layout shifts:

- Instead of changing the height and width properties, use `transform: scale()`.
- To move elements around, avoid changing the top, right, bottom, or left properties and use `transform: translate()` instead.

```
transform: scale(0.7);
```

```
transform: translate(42px,18px);
```

FID

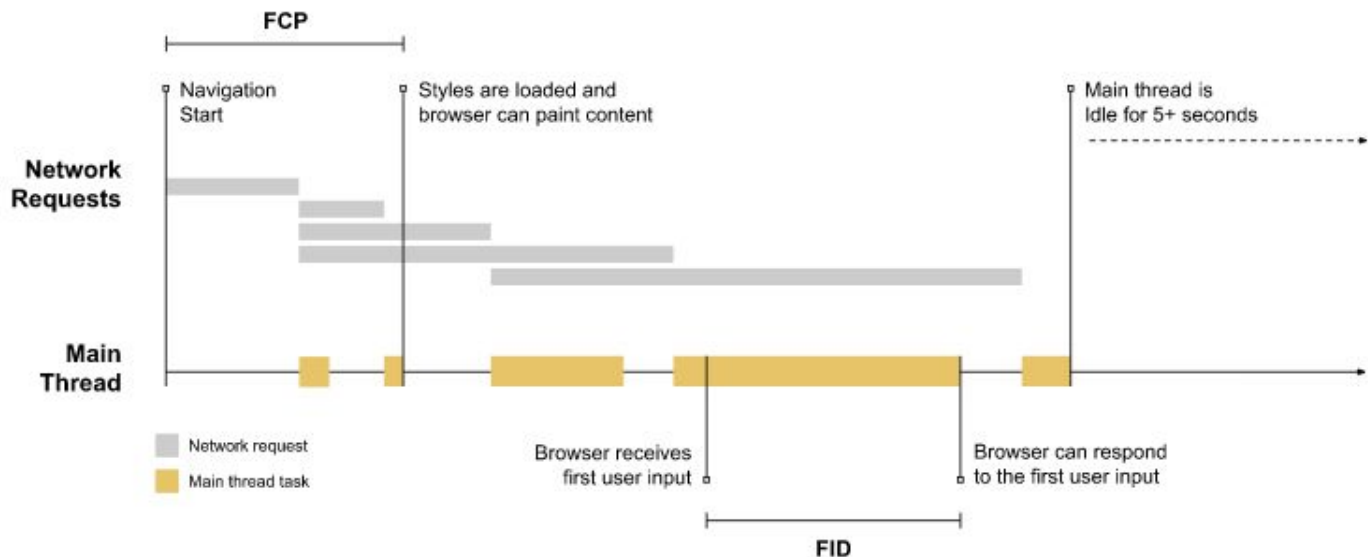
First Input Delay



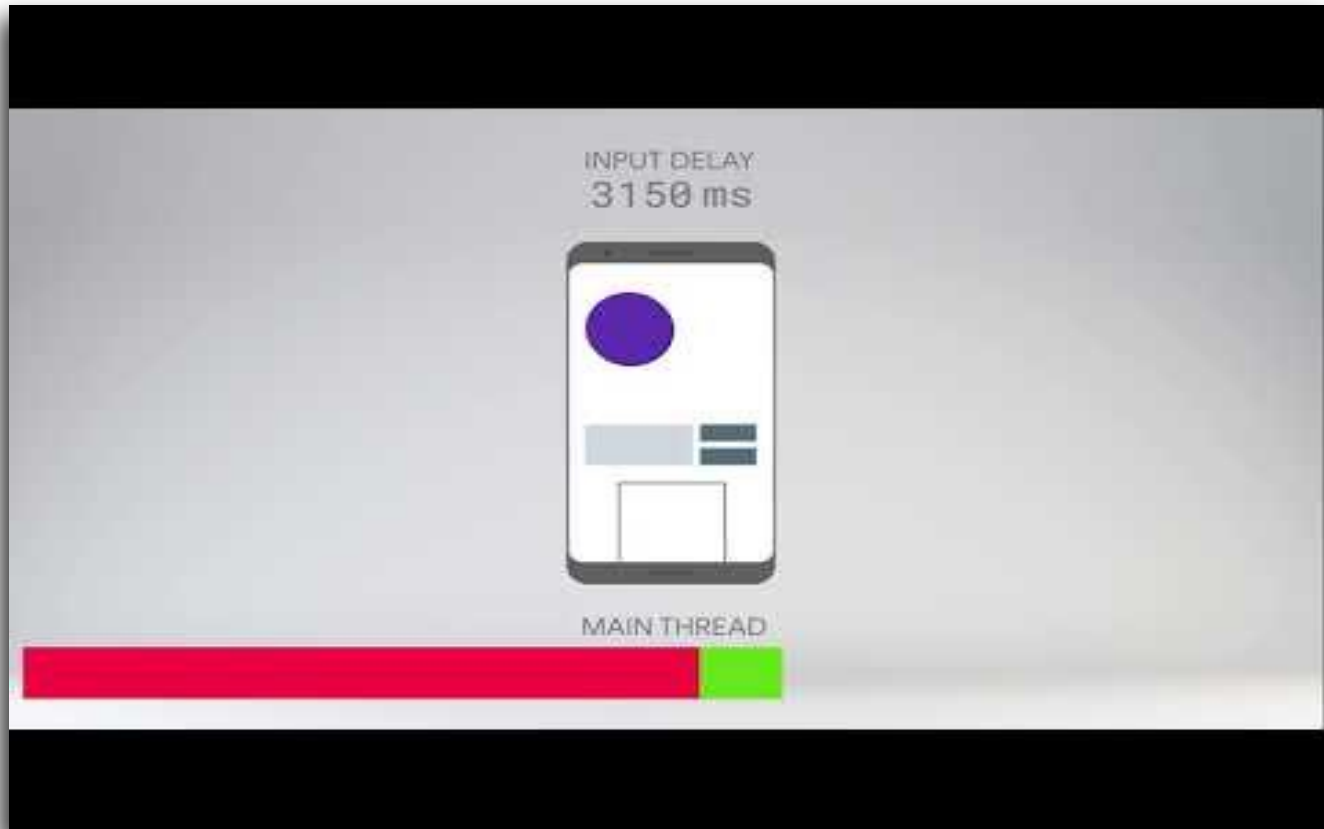
- Measures your user's first impression of your site's interactivity and responsiveness.
- It's the time from when a user first interacts with a page to the time when the browser is actually able to respond to that interaction.
- It only focuses on input events from discrete actions like *clicks, taps, and key presses*.
- You are not guaranteed to have FID value. No user interaction, no FID value is recorded.

Why does input delay happen?

- Input delay happens because the browser's main thread is busy doing something else, so it can't (yet) respond to the user (ex. It is busy parsing and executing a large JavaScript file loaded by your app) .



Why does input delay happen?

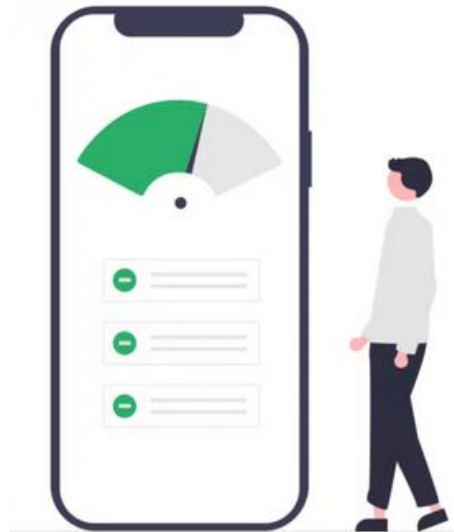


What causes poor FID & how to optimize it?

The main cause of a poor FID is **heavy JavaScript execution**. Optimizing how JavaScript parses, compiles, and executes on your web page will directly reduce FID.

To improve this:

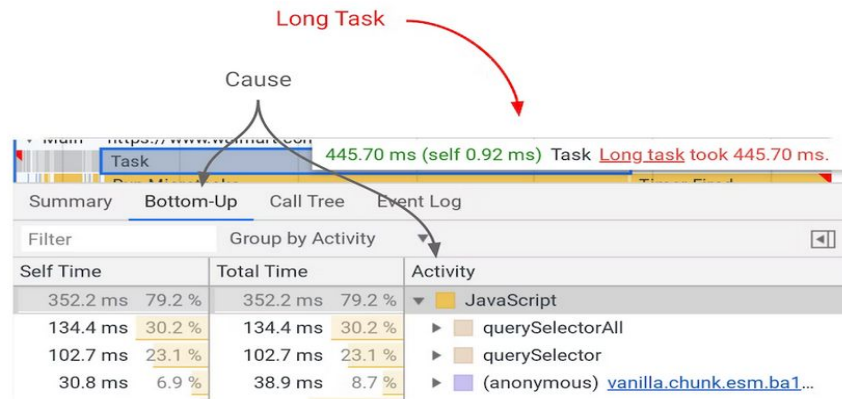
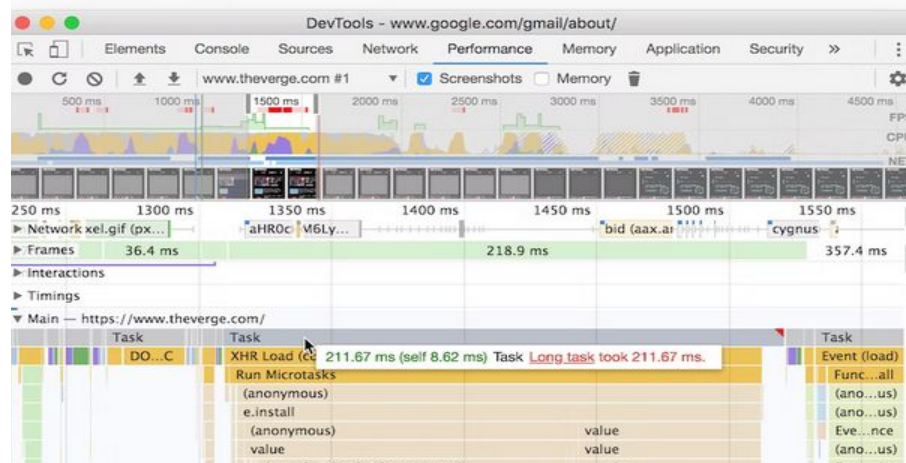
- Break up Long Tasks.
- Reduce JavaScript execution time.



Optimize First Input Delay

1. Break up Long Tasks:

- Long task could be any piece of code that blocks the main thread for 50 ms or more, causing the UI to "freeze".
- It is sign of loading and executing more than a user may need right now.
- Splitting up long tasks can reduce FID.



Code splitting by dynamic imports

- It's the concept of splitting a single large JavaScript bundle into smaller chunks that can be conditionally loaded .
- Send only the code that is needed for the initial route when the user loads an application.
- Hence, minimizing the amount of script that needs to be parsed and compiled.
- And as a result, minimizing main thread work and having a good FID.

```
import moduleA from "library";

form.addEventListener("submit", e => {
  e.preventDefault();
  someFunction();
});

const someFunction = () => {
  // uses moduleA
}
```

```
form.addEventListener("submit", e => {
  e.preventDefault();
  import('library.moduleA')
    .then(module => module.default) // using the default export
    .then(() => someFunction())
    .catch(handleError());
});

const someFunction = () => {
  // uses moduleA
}
```

Optimize First Input Delay

2. Reduce JavaScript execution time

To reduce the amount of JavaScript executed on your page:

- Defer unused JavaScript
 - Code-splitting
 - defer / async scripts
- Use web workers
- Preloading important JavaScript files



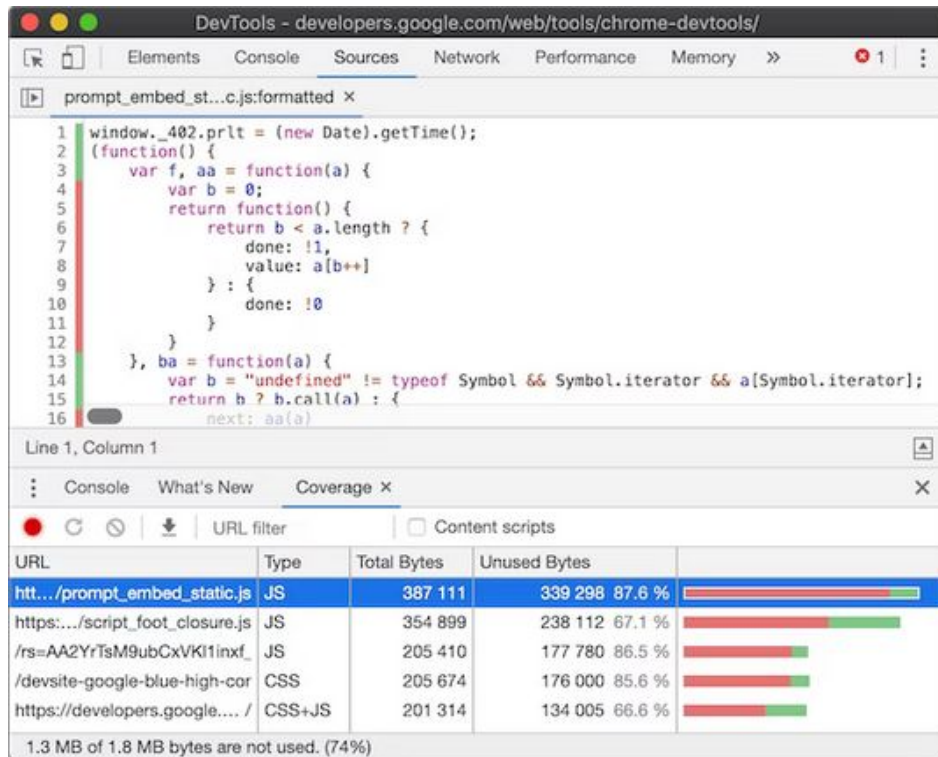
Browser



The Server

Knowing unused JavaScripts

The Coverage tab in Chrome DevTools can tell you how much JavaScript is not being used on your web page.



Defer unused JavaScripts

Aside from code-splitting, always use `async` or `defer` for scripts that are not necessary for above-the-fold content.

All third-party scripts should be loaded with either `defer` or `async` by default.

Use web workers

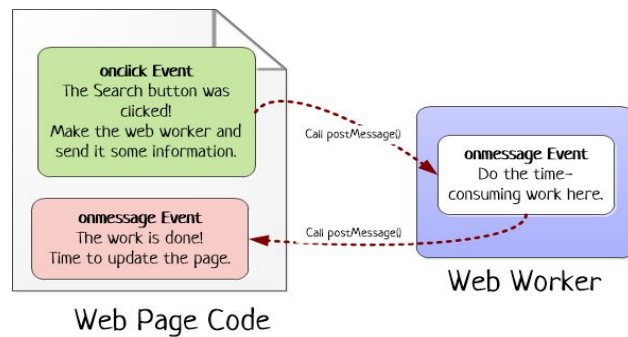
A blocked main thread is one of the main causes of input delay. Web workers make it possible to run JavaScript on a background thread.

Preloading important JavaScript files

This prevents the scripts from blocking the main-thread at critical phases of the page load, enabling your users to interact faster with the page.

“Load now, execute when needed” .

```
<script defer src="..."></script>  
<script async src="..."></script>
```



```
<link rel="preload" href="script.js" as="script">
```

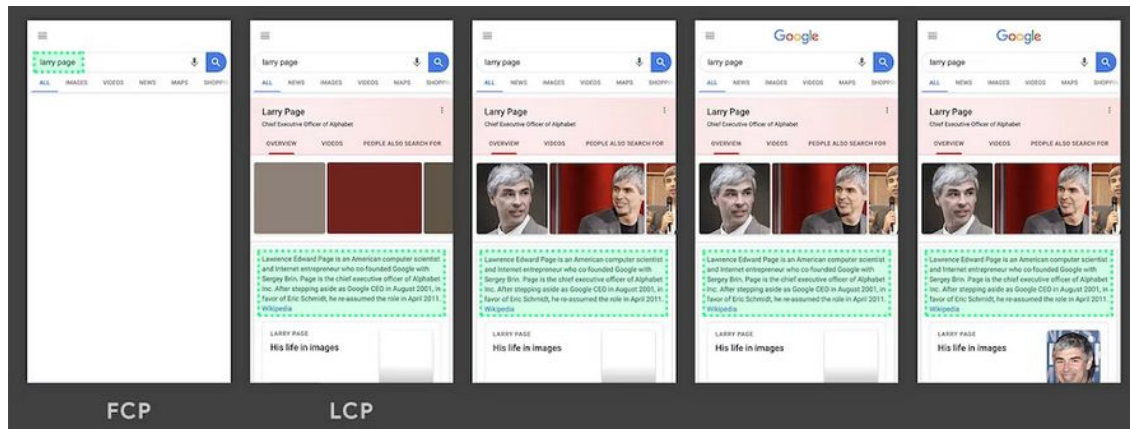
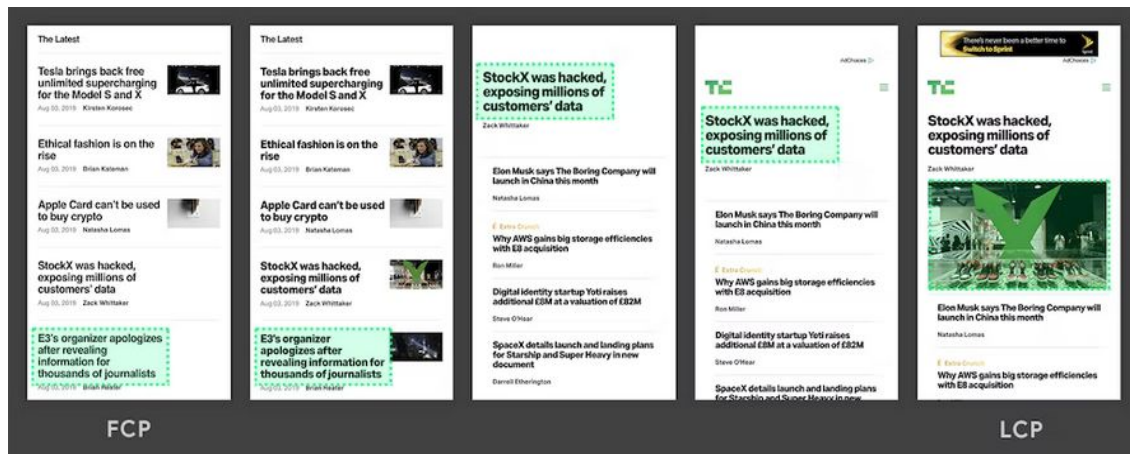
LCP

Largest Contentful Paint



- Measures the render time of the largest image or text block visible within the viewport.
- The types of elements considered for LCP are images, videos, block-level elements with text and element with background image.
- The browser will stop reporting new entries for LCP as soon as the user interacts with the page .

LCP Examples



What causes poor LCP?



Slow Server Response Time



Render-blocking JS & CSS

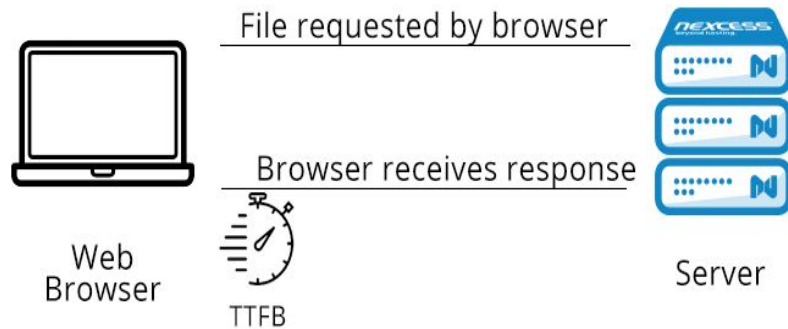


Slow Resource Load Time

Optimize Largest Contentful Paint

1. Reduce server response time

- Analyze and optimize your servers.
- Compress text files.
- Configure Caching.
- Use a Content Delivery Network (CDN).



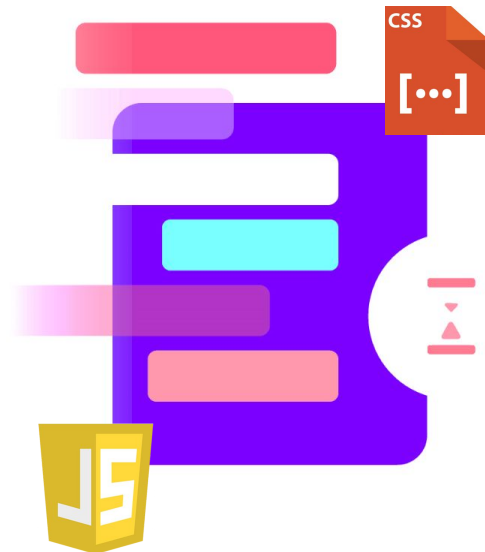
Content Delivery Network (CDN)



Optimize Largest Contentful Paint

2. Remove render-blocking resources

- Don't load unnecessary bundles.
- Inline critical CSS.
- Minify and compress the content.



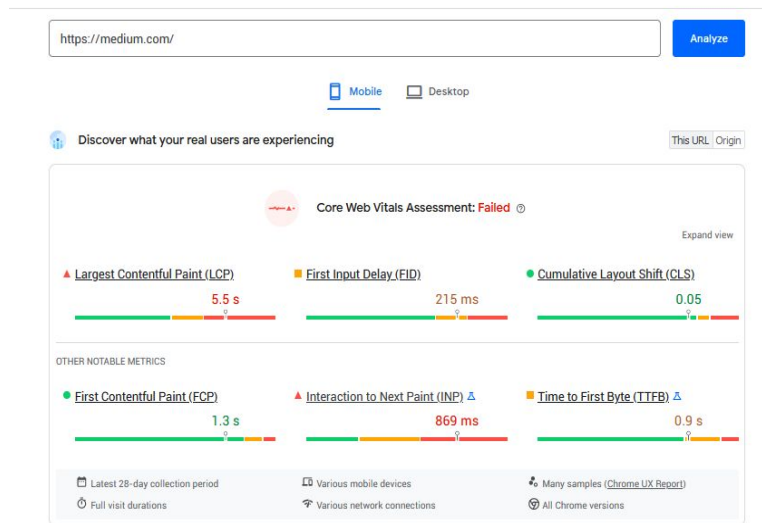
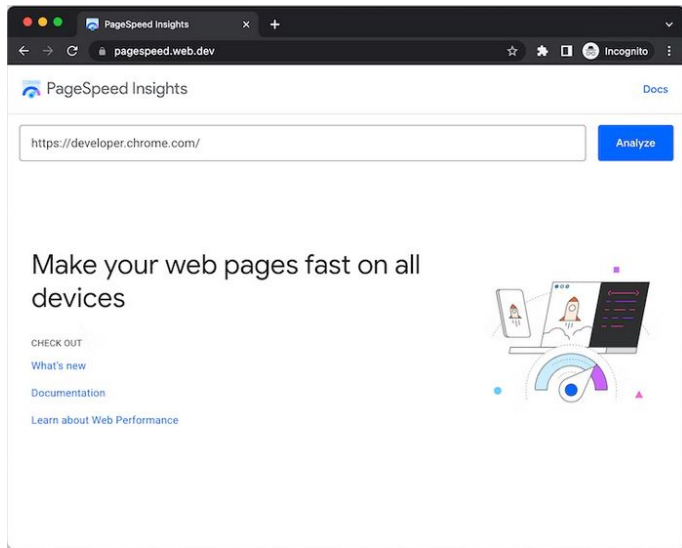
Optimize Largest Contentful Paint

3. Reduce resource load time

- Reduce the size of the resource
- Reduce the distance the resource has to travel.
- Reduce contention for network bandwidth.
 - `fetchpriority="high"`



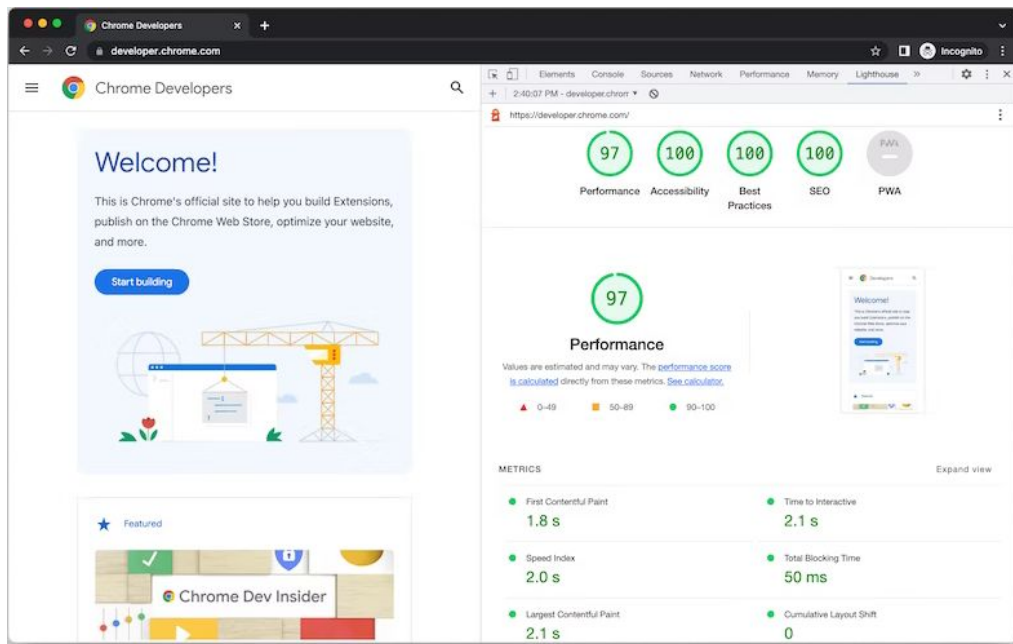
Measure core web vitals with Lighthouse tool



Run Lighthouse on PageSpeed Insights

Visit: <https://developer.chrome.com/docs/lighthouse/overview/#devtools>

Measure core web vitals with Lighthouse tool



Run Lighthouse on in Chrome DevTools

Visit: <https://developer.chrome.com/docs/lighthouse/overview/#psi>

Thanks for your attention!