



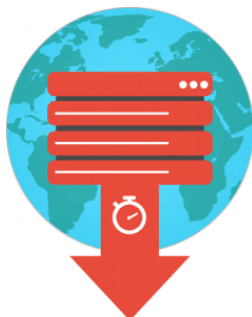
## 1.INTRODUCCIÓN OPTIMIZACIÓN WEB

La optimización de velocidad del sitio web, es el principal factor que dicta el éxito de los negocios online modernos. Que las páginas web lentas se carguen frustra a los visitantes en la búsqueda de alternativas ¡la impaciencia es una virtud digital!

Una rápida velocidad de carga de la página amplifica la participación de los visitantes, la retención y aumenta las ventas. La respuesta instantánea del sitio web conduce a tasas de conversión máximas y cada 1 segundo de retraso en la carga de la página disminuye la satisfacción del cliente en un 16 por ciento, las visitas en un 11 por ciento y las tasas de conversión en un 7 por ciento.

### ¿Pero qué es la velocidad de Página?

El término se refiere esencialmente a la **velocidad** con la que las páginas Web o el contenido de los medios se **descargan de los servidores de alojamiento de sitios web** y se muestran en el navegador web solicitante. El tiempo de carga de la página es la duración entre hacer clic en el vínculo y mostrar todo el contenido desde la página Web del navegador solicitante.



Hay tres aspectos básicos necesarios para entender la velocidad de la página en el contexto de la experiencia del usuario y el rendimiento del sitio web:

- La visión del **tiempo que se tarda en entregar el material solicitado** junto con el contenido HTML que lo acompaña al navegador.
- **Respuesta del navegador a solicitudes** de carga de página.
- **La vista de los usuarios finales conforme la página web solicitada se visualiza** en el navegador.

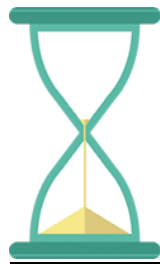
El rendimiento del sitio web influye en el posicionamiento en los buscadores (SEO, Web Vitals [link](#))

## ¿Lento? ¿Qué Tan lento?

Cualquier cosa más lenta que el parpadear los ojos (400 milisegundos).

## ¿Y qué sucede cuando no lo hacen?

1 en 4 visitantes abandonaría el sitio web si tarda más de 4 segundos en cargarse. **El 46 % de los usuarios no vuelven a visitar los sitios web de bajo rendimiento.** Los dueños de sitios web tienen apenas 5 segundos para interactuar con los visitantes antes de que consideren irse. El 74 % de los usuarios que acceden al sitio móvil lo dejaría si tarda más de 5 segundos en cargarse.



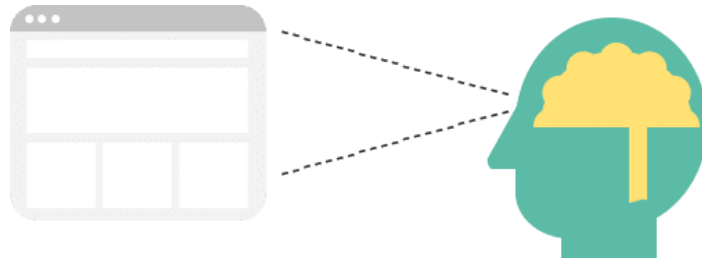
## ¿Y qué pasa cuando se aceleran?

Cuando Mozilla aumentó la velocidad de la página en 2.2 segundos, las cifras de descargas de Firefox aumentaron un 15.4 %! Walmart [vio un aumento del 2 por ciento en tasas de conversión por cada 1 segundo](#) de mejora en los tiempos de carga de página.



## La Neurociencia y el Ritmo

Los investigadores de Google sugieren que tiempos de carga de página de menos de 100 milisegundos dan a los visitantes la ilusión de respuesta instantánea,



1 segundo de tiempo de carga de la página es suficiente para mantener el flujo continuo de pensamiento, A los 10 segundos de retraso, la atención del visitante apenas se mantiene. La sensación de impaciencia, frustración y sentimiento de abandono suele ser lo suficientemente fuerte como para impedir que los visitantes vuelvan a visitar sitios web tan lentos de nuevo.

## El Enigma Psicológico Pragmático

Un usuario promedio pasaría varios minutos más navegando por sitios web de respuesta rápida pero irrelevantes, en lugar de esperar unos segundos más para que sitios Web relevantes y lentos respondan.

La experiencia es similar al incidente del aeropuerto de Houston de hace unos años que llevó a una solución inteligente:

*“Los pasajeros en el aeropuerto tenían que tomar un paseo de 1 minutos a la reclamación de equipaje y esperaron allí por 7 minutos para recibir sus maletas. El centro de relaciones con el cliente fue finalmente inundado con quejas por el servicio lento.*

*Así que a los ejecutivos del aeropuerto se les ocurrió una solución astuta para «resolver» el problema, trasladaron las puertas de llegada lejos y llevaron el equipaje al carrusel más externo.*

*Los pasajeros ahora tenían que tomar un paseo de 6 minutos a la zona de reclamo de equipaje donde recibieron su equipaje en 2 minutos. ¿El resultado? Cero quejas”.*

Del mismo modo, los ascensores contienen espejos para que los pasajeros se mantengan ocupados peinándose y arreglándose y no pensar en el tiempo que pasaron viajando.

El mismo principio general se aplica para la página web.

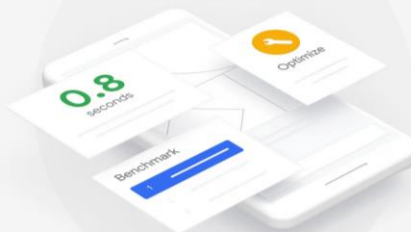
A pesar de lo anterior, nuestro objetivo será **reducir la duración de la espera** en la entrega del contenido solicitado a sus visitantes online con un sitio web de respuesta rápida, lo que es posible cuando el sitio web está diseñado para ofrecer una respuesta instantánea y de alta calidad y perfecta experiencia de usuario.

Para evaluar nuestra web en un entorno móvil podemos usar:

<https://www.thinkwithgoogle.com/feature/testmysite/>

[Test My Site](#)

Improve your mobile site  
to boost your business.



## 2. MECANISMOS IMPORTANTES: Tips que pueden hacer un rendimiento menor de nuestra web.

La optimización de la velocidad se implementa idealmente en todas las etapas del desarrollo del sitio web, y no sólo después de construir todo el sitio, que es sólo cuando los propietarios de sitios web se dan cuenta de la necesidad de impulsar la optimización del rendimiento del sitio web.

### 2.1 ERRORES COMUNES

- [Servicio de Alojamiento Web Mediocre:](#)

Los servidores sin optimizar que ejecutan miles de sitios web públicos en una única pila de servidores son particularmente dañinos para los sitios web de comercio electrónico caracterizados por picos de tráfico web incontrolables, contenido multimedia y archivos de sitios web grandes.

- [Demasiados Widgets o Plugins](#)

Los widgets y plugins permiten cambios convenientes en sitios web existentes junto con una ligera carga en el rendimiento del sitio web. Incluso con el más pequeño de los widgets como el botón de Google+, la carga de rendimiento del sitio web en términos de tiempo de carga de la página puede aumentar hasta 2 segundos en algunos casos.

Mantener los plugins limitados a un mínimo.

- [Demasiados Anuncios y Servicios Externos](#)

Vender demasiados bienes a anunciantes de terceros degrada drásticamente el rendimiento del sitio web. Si tiene muchos servicios externos debe cargar todos y esperar para recibir información cada vez que carga la página.

El costo de generar ingresos con publicidad en banners es casi el 33 por ciento de latencia adicional.

## 2.2 TIPS QUE PUEDEN HACER UN RENDIMIENTO MENOR DE NUESTRA WEB

- **Fallback:** es un valor predeterminado que puede establecerse en tus estilos por si no existe soporte para el valor o propiedad que realmente quieres usar. Existen básicamente dos tipos de fallbacks:
- **Colocando propiedades por defecto** (aparte de las nuevas), propiedades que sí pueda interpretar el navegador antes de las propiedades nuevas (probar la web en navegadores antiguos <https://www.browserling.com/>).
- **Colocando feature queries** (o mejor conocido como @supports ):

```
@supports (display:flex) and (not (display:grid)) {  
    //Si soportas flex y no soportas grid, interpreta  
    el contenido de las llaves.....}
```

## 2.3 PROBLEMAS FRECUENTES

Los problemas que suelen surgir con más frecuencia están relacionados con los siguientes puntos:

- **Especificidad y selectores.**

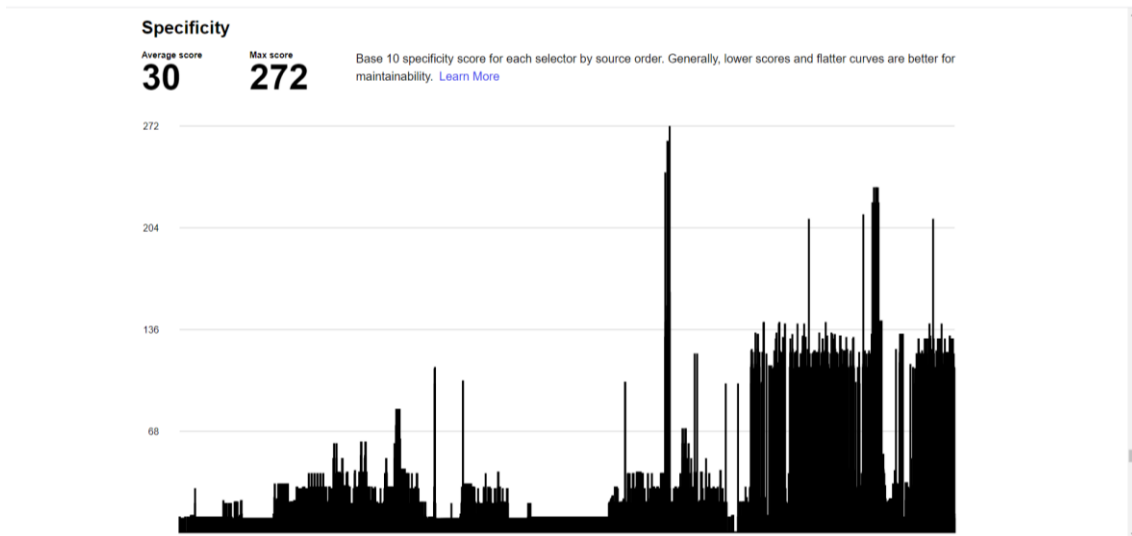
Cuanto más específico sea nuestro CSS, si hay alguna modificación en el estilo, menos ramas del árbol CSSOM se verán afectadas y la carga (reflow) será más rápida.

Calcular la especificidad: <https://specificity.keegan.st/>

El proyecto **CSSSTATS** nos proporciona un análisis que puede usarse para mejorar la consistencia, realizar un seguimiento del rendimiento y diagnosticar áreas complejas.

<https://cssstats.com/>

Lo interesante que nos proporciona en cuanto a la especificidad es un gráfico que nos muestra una puntuación media de la especificidad: En general, las puntuaciones más bajas y las curvas más planas son mejores para la mantenibilidad. [Aprende más](#)



- **Layout Thrashing**

**¿Qué es Layout Thrashing?** Después de que se carga la página web o durante la carga de la página web, un archivo javascript puede volver a dibujar o mutar los elementos de la página web mediante la creación de layout thrashing. Para evitar el diseño de JavaScript, se debe evitar la modificación del DOM por JavaScript o cambiar los elementos de estilo con JS repetidamente. (Scope -> CSS in JS :-( )

**Diferencias entre Reflow y Layout thrashing:**

REFLOW: es un proceso de cálculo del tamaño y la posición del elemento DOM en una página web. Ocurre cuando:

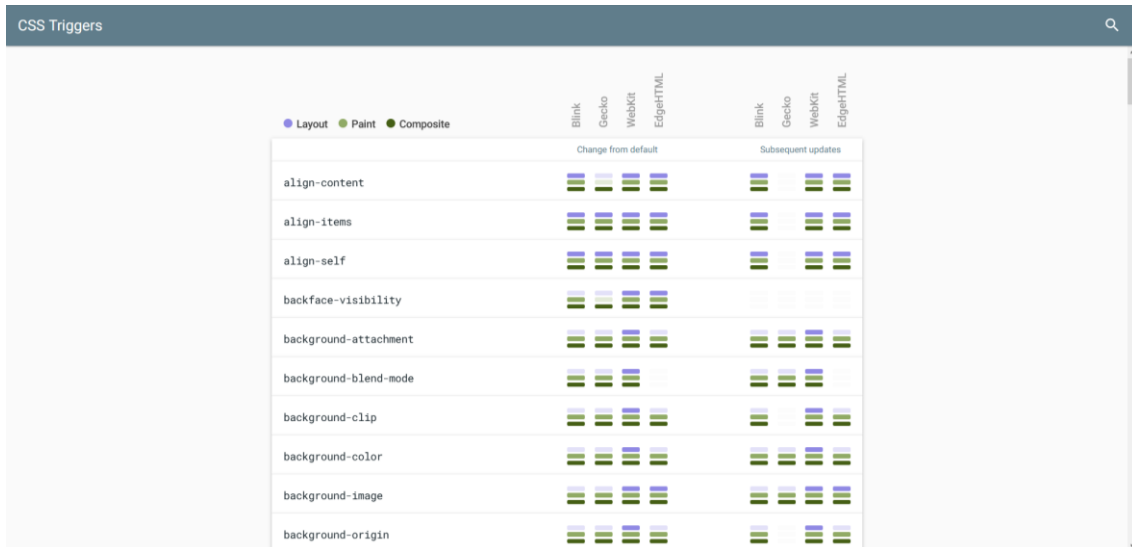
- ✓ Un elemento se agrega, elimina o actualiza en el DOM.
- ✓ Modificación del contenido de la página web.
- ✓ Cambiar un estilo CSS.
- ✓ Cambiar el tamaño de la ventana del navegador, etc.
- ✓ Cada vez que cambie algo en el DOM, el navegador volverá a calcular las posiciones y dimensiones de los elementos del DOM y luego volverá a pintar la pantalla.

**Layout Thrashing:**

Los REFLOW son costosos, por lo que los navegadores intentan poner en cola los cambios y aplicarlos en lotes para minimizar los reflow necesarios. Esta estrategia permite que los navegadores apliquen varios estilos y realicen solo un reflow.

## ► propiedades costosas de CSS

Cuando se realizan modificaciones de los valores del estilo por defecto. CSSSTRIGGER(<https://csstriggers.com/>) nos permite visualizar cada propiedad CSS indicando que forzaría si modificamos su valor: **Layout, Paint y Composite** para todos los motores de navegadores que existen.



	Layout	Paint	Composite	Blink	Gecko	WebKit	EdgeHTML	Blink	Gecko	WebKit	EdgeHTML
				Change from default				Subsequent updates			
align-content											
align-items											
align-self											
backface-visibility											
background-attachment											
background-blend-mode											
background-clip											
background-color											
background-image											
background-origin											

## 2.4 COMPRENDER LA LÍNEA DE TIEMPO

¿Qué hace el navegador mientras renderiza? Esta línea de tiempo muy simple se llama **Critical Rendering Path**.

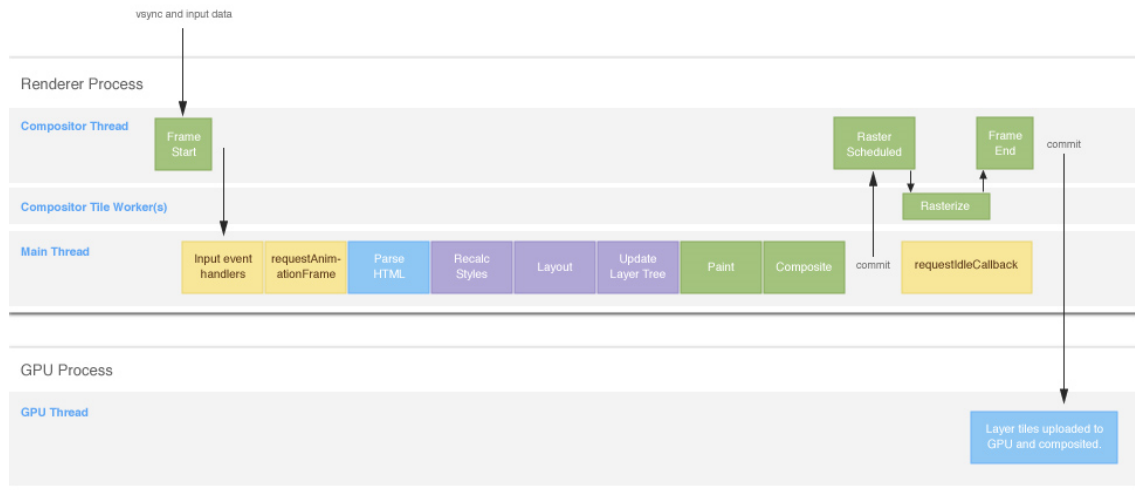


- **Styles:** El navegador comienza a calcular los estilos que se aplicarán en los elementos recalculando el estilo.
- **Layout:** El navegador comenzará a generar la forma y la posición de cada uno de esos elementos: **styles**. Aquí es donde el navegador establece las propiedades de la página, como el **ancho** y el **alto**, así como sus **márgenes**.
- **Paint:** El navegador ahora comenzará a completar los píxeles de cada elemento en capas. Las propiedades que utiliza son, por ejemplo: **box-shadow**, **border-radius**, **color**, **background-color**.
- **Composite:** Aquí es donde el navegador comienza a dibujar todas las capas en la pantalla.



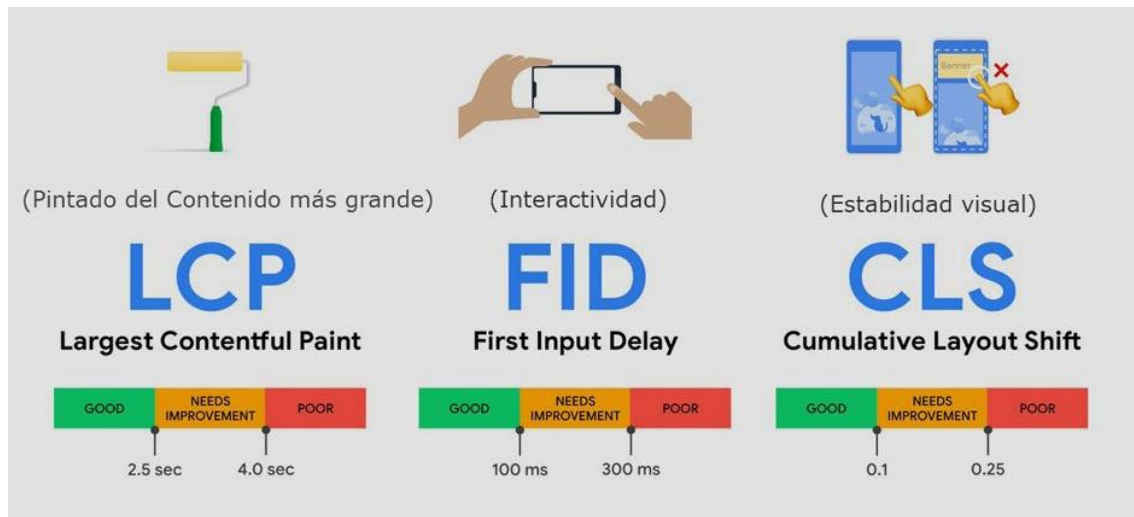
## 2.5 ANATOMÍA DE UN FRAME

Para pintar una frame en una página web se necesitan estos dos procesos:



- **Proceso de renderizado** . Contiene múltiples subprocesos que, juntos, son responsables de varios aspectos para mostrar su página en la pantalla. Estos subprocesos son:
  - **Compositor thread**. Este es el primer subproceso en ser informado sobre el evento vsync (que es como el sistema operativo le dice al navegador que cree un nuevo marco).
  - **Tile Worker**. manejar las tareas de rasterización.
  - **Main**: Aquí es donde el navegador ejecuta las tareas: JavaScript, estilos, diseño y pintura.
- **Proceso GPU** El proceso GPU contiene un solo subproceso, llamado subproceso GPU que realmente hace el trabajo de llevar los datos (como vértices y matrices) a la GPU para pintar los píxeles a la pantalla.

## 2.6 CONCEPTOS LCP, FID Y CLS



Las métricas que usaremos para analizar y optimizar la carga de una página web son:

- **LCP:** Largest Contentful Paint, algo así como el “Pintado de Contenido más grande”.
- **FID:** First Input Delay, algo así como el “Retardo en la primera interacción”.
- **CLS:** Cumulative Layout Shift, algo así como el tiempo que tarda en estabilizarse el diseño.

### LCP Largest Contentful Paint

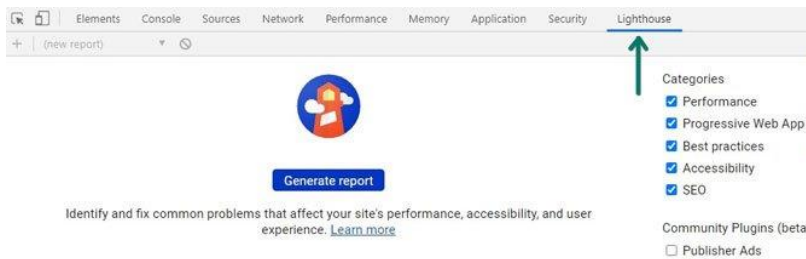


Qué tan rápido empieza a mostrar una web la mayoría de elementos que se ubican en el “above the fold”.

El LCP mide el tiempo que transcurre hasta que se carga el elemento de contenido más grande de una página. Pueden ser imágenes, bloques de texto o elementos con una imagen de fondo. Según Google, el 90 % de los casos es una imagen.

Para esta métrica LCP, los tiempos de carga de hasta 2,5 segundos son buenos, cualquier valor entre 2,5 y 4 segundos necesita mejoras, y cualquier valor inferior a 4 segundos es malo.

“Lighthouse” es la herramienta creada por Google para medir estas métricas.



Ejemplo visual de lo que es el LCP



En 0,7 segundos nos muestra el primer pintado: se empiezan a ver los primeros elementos de la web. Sin embargo, hasta que no llegamos a los 3,5 segundos, lo que se ve en el “above de fold” (el primer pantallazo que vemos de una web sin necesidad de hacer scroll) no queda cargado completamente. Más info sobre LCP [link](#)

**FID: ¿Cuándo puede el usuario interactuar con una página?**



El retraso en la primera interacción (FID) mide el tiempo desde que un usuario interactúa por primera vez con un sitio hasta el punto en el que el navegador puede responder a esa interacción.

Por ejemplo, los usuarios van a una web e inmediatamente hacen clic en el texto o en un botón sin esperar a que primero la página se cargue por completo. Es frecuente que no suceda nada porque el navegador está ocupado cargando la página.

La métrica FID mide el retraso que ocurre entre la entrada del usuario y la respuesta del navegador. Según Google, cualquier valor inferior a 100 milisegundos es bueno, cualquier valor entre 100 y 300 milisegundos necesita mejoras, mientras que los valores de FID superiores a 300 milisegundos se consideran pobres. Más info: [link](#)

CLS: la estabilidad visual en una web.



El CLS se refiere a la estabilidad visual durante la interactividad en una web. Cuando se carga una web, algunos elementos pueden aparecer primero en la pantalla, pero cuando se carga un elemento que tarda demasiado, empuja a los otros elementos hacia abajo en la página.

Cuando se cargan los primeros elementos, el usuario ya puede empezar a interactuar con ellos, pero de repente, el elemento que tarda más en cargarse aparece en la pantalla y el usuario hace un clic no deseado en ese elemento final.

Podemos ver un ejemplo en el siguiente vídeo.

<https://www.youtube.com/watch?v=1b1S163tDI&t=14s>

Otro ejemplo puede darse cuando el usuario quiere hacer clic en descargar un recurso y de repente se carga un banner de AdSense y el usuario acaba haciendo clic en la publicidad. Esto es malo para Google y puede que haya pensado esta métrica para no perjudicar su negocio de publicidad.

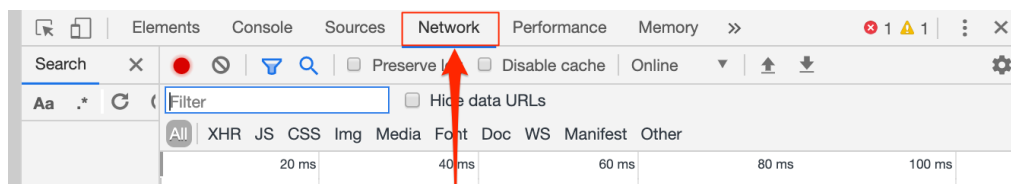
En otras palabras, el CLS muestra si se producen cambios de diseño inesperados mientras el usuario interactúa con la web. Cuanto menor sea esta métrica, mejor.

Para Google, cualquier valor por debajo de 0,1 es bueno y cualquier valor superior a 0,25 es malo. Cualquier cosa intermedia necesita mejorar. Más Info: [link](#)

### 3. CSS EN EL CONTEXTO DEL NAVEGADOR. AUDITORÍA

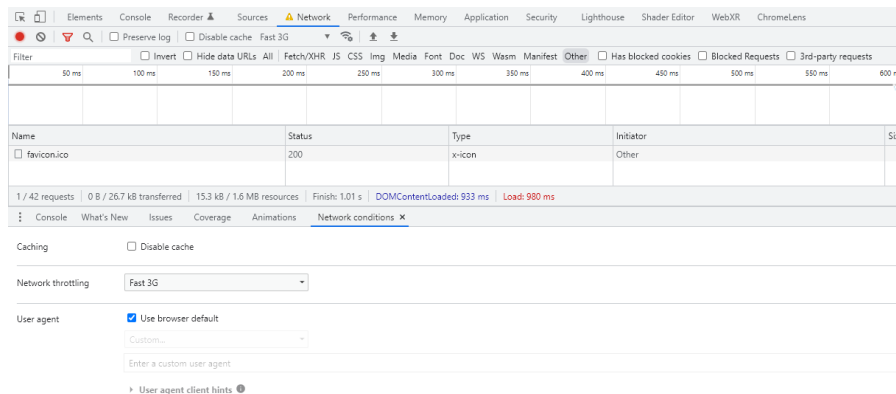
Para empezar a realizar una auditoría de nuestra web, empezaremos usando el navegador en modo incognito, de esa manera nos evitaremos tener problemas con el historial o plugins instalados.

En las devtools, nos iremos a la pestaña Network, pulsamos el botón de grabar y actualizamos la página web que queramos auditar.



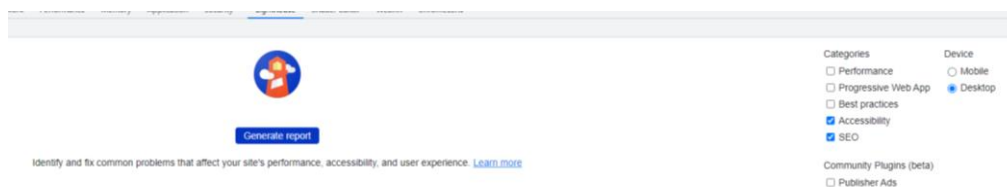
Podremos visualizar el **waterfall de todos los recursos** y las **prioridades** que tienen.

Por otro lado, si queremos testar la carga en otras conexiones más lentas, podemos simular conexiones más lentas con “More network condition”:

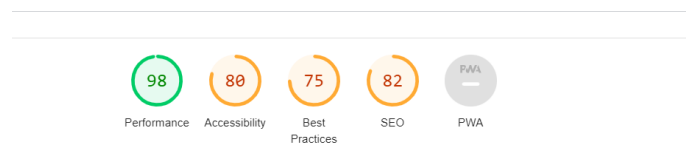


## 3.2 Interpretando una auditoría

Lo primero será ir a la pestaña **Lighthouse**, seleccionamos las categorías, en nuestro caso usaremos **Performance** y el dispositivo a analizar y pulsamos el botón Generar Report.



Como resultado nos aparecerá lo siguiente:



Dentro de Performance, nos aparecerán **6 tipos de métricas**:

## METRICS

● First Contentful Paint

0.7 s

● Time to Interactive

0.9 s

● Speed Index

1.0 s

● Total Blocking Time

0 ms

● Largest Contentful Paint

0.9 s

● Cumulative Layout Shift

0.002

**First Contentful Paint** : FCP mide cuánto tiempo le toma al navegador mostrar la primera parte del contenido DOM después de que un usuario navega a su página. Las imágenes, los <canvas> elementos no blancos y los SVG en su página se consideran contenido DOM; no se incluye nada dentro de un iframe .

Tiempo FCP (en segundos)	Código de colores
0–1.8	Verde (rápido)
1.8–3	Naranja (moderado)
más de 3	rojo (lento)

**Time to Interactive (TTI)**: Medir el TTI es importante porque algunos sitios optimizan la visibilidad del contenido a costo de la interactividad. Esto puede crear una experiencia de usuario frustrante: el sitio parece estar listo, pero cuando el usuario intenta interactuar con él, nada sucede.

Métrica TTI (en segundos)	Codificación de color
0-3.8	Verde (rápido)
3.9–7.3	Naranja (moderado)
Más de 7.3	Rojo (lento)

**Speed Index:** El índice de velocidad mide la rapidez con la que se muestra visualmente el contenido durante la carga de la página, calcula la percepción que tiene el usuario de la velocidad de carga. Lighthouse primero captura un video de la carga de la página en el navegador y **calcula la progresión visual entre fotogramas**.

Índice de velocidad (en segundos)	Código de colores
0–3.4	Verde (rápido)
3.4–5.8	Naranja (moderado)
Más de 5,8	rojo (lento)

**Total Blocking Time:** La TBT mide la cantidad total de tiempo que una página está bloqueada para que no responda a la entrada del usuario, como los clics del ratón, toques de la pantalla o **pulsaciones del teclado**. La suma se calcula sumando la parte de bloqueo de todas las tareas largas entre First Contentful Paint (primer despliegue de contenido) y Time to Interactive (tiempo de interacción). Cualquier tarea que se ejecute durante más de 50 ms es una tarea larga. La cantidad de tiempo después de 50 ms es la parte de bloqueo. Por ejemplo, si Lighthouse detecta una tarea de 70 ms de duración, la porción de bloqueo sería de 20 ms.

Tiempo TBT (en milisegundos)	Codificación de color
0-200	Verde (rápido)
200-600	Naranja (moderado)
Más de 600	Rojo (lento)

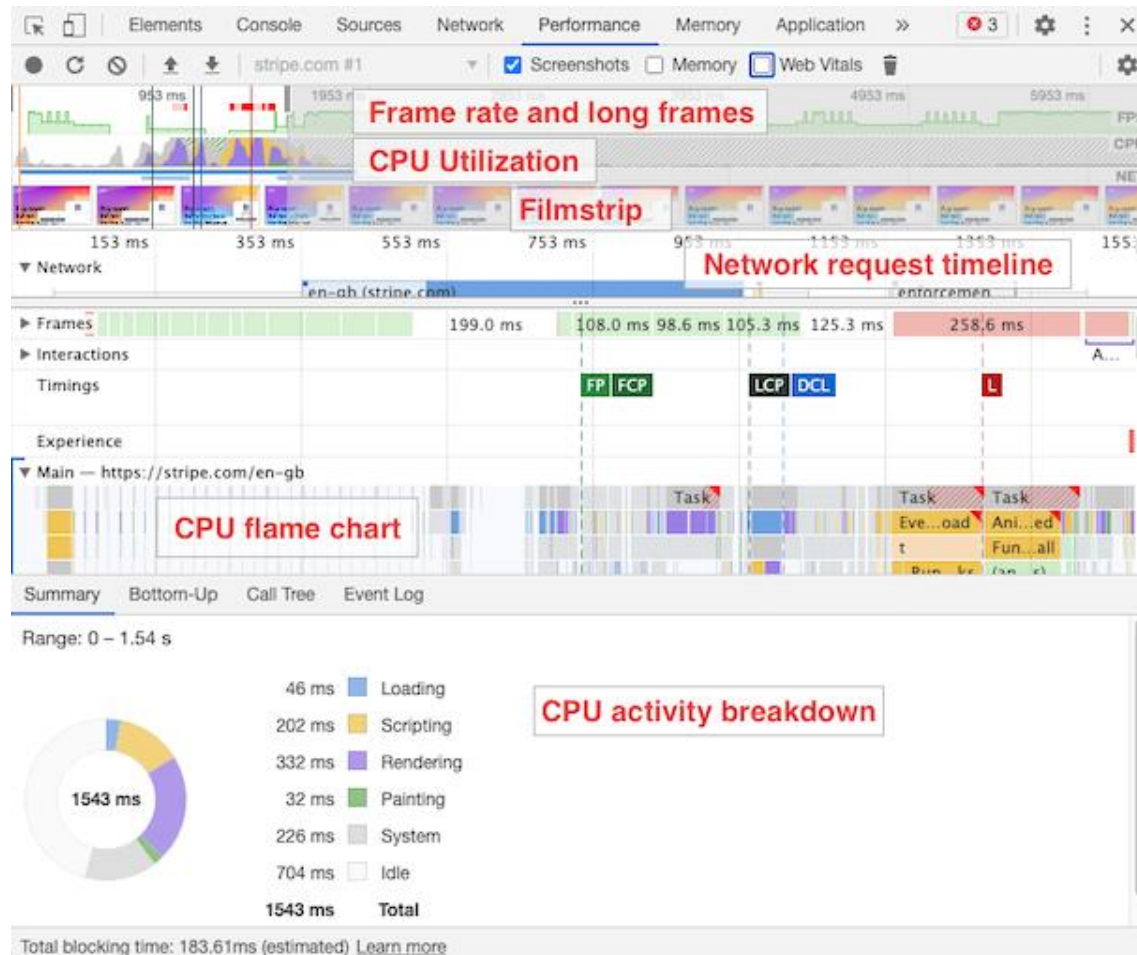
**Largest Contentful Paint y Cumulative Layout Shift** vistos anteriormente.

Recurso: Optimización de Web Vitals con Lighthouse [link](#)

### 3.4 Descripción Performance

La pestaña “PERFORMANCE” de Chrome DevTools está repleta de funciones que permiten auditar el rendimiento de la página en profundidad.

Los resultados obtenidos a través de “performance”, pueden correlacionar diferentes tipos de actividades de la página e identificar las causas de un problema de rendimiento.



#### CPU utilization timeline

Este gráfico muestra qué tan ocupada está la CPU con diferentes tipos de tareas, generalmente en su mayoría JavaScript (amarillo), trabajo de diseño (púrpura) y pintado (verde).



#### Filmstrip

Muestra el progreso de renderizado de su sitio web de una manera intuitiva. Puede pasar el cursor sobre Filmstrip para ver una captura de pantalla de ese momento.

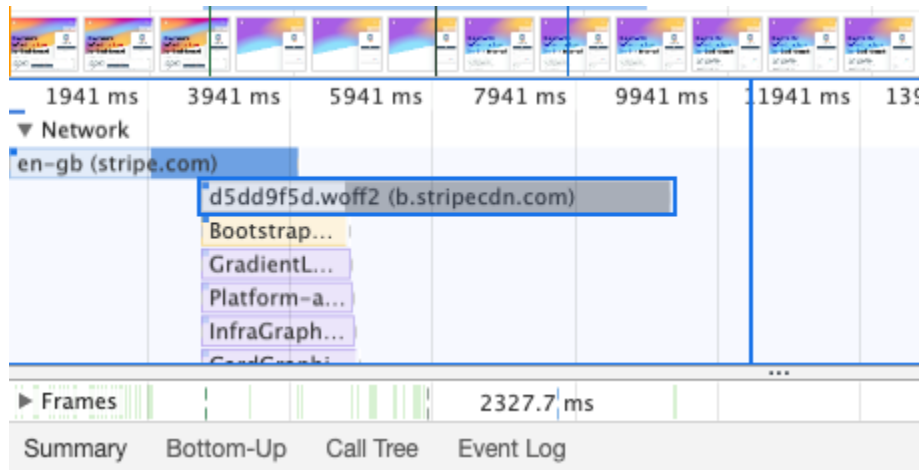




## Network request timeline

Muestra una cascada de solicitudes, comenzando con la solicitud HTML en la parte superior y luego mostrando solicitudes adicionales debajo.

Haga clic en cada solicitud para ver información adicional, como la URL completa, la duración de la solicitud, la prioridad de los recursos y el tamaño de la descarga.



### Network request

URL [d5dd9f5d.woff2](https://b.stripecdn.com/d5dd9f5d.woff2)

Duration 6.62 s (6.62 s network transfer + 2.48 ms resource loading)

Request Method GET

Priority High

Mime Type application/octet-stream

Encoded Data 35.9 kB

Decoded Body 35.7 kB

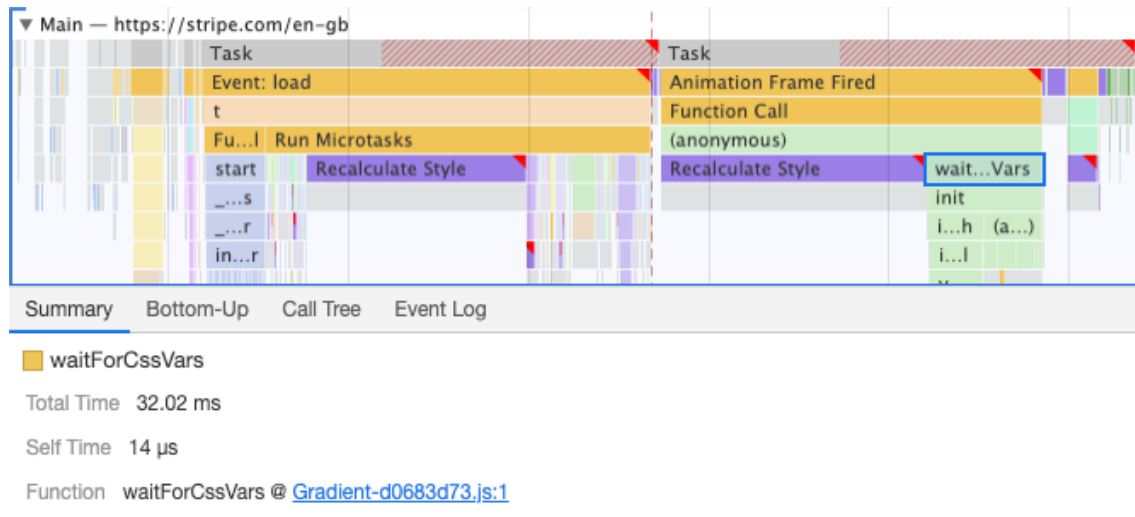
## CPU flame chart

Contiene un gráfico que muestra cómo las tareas de la CPU se dividen en diferentes componentes.

Por ejemplo, puede ver una `waitForCssVars` llamada de función en el gráfico.

Mirando arriba, nos dice que esta función fue llamada por una función anónima, que en el término fue llamada porque se usó como una `requestAnimationFrame` devolución de llamada.

También podemos ver que la `init` función se llama desde dentro `waitForCssVars`.



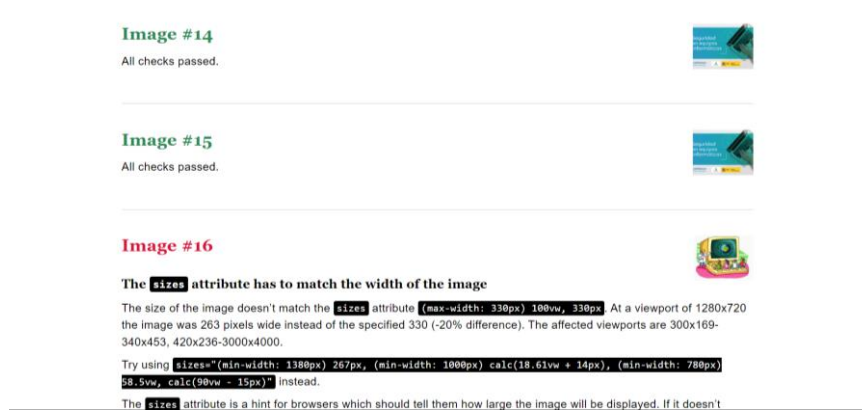
## 4 RECURSOS DE LA WEB: ASSETS

### 4.1 Imágenes:

Para la optimización de las imágenes en la web, tenemos que tener presente lo siguiente:

1. Imágenes con tamaño adecuado al viewport (`srcset`, densidad de pixel)).
2. `RespImageLint`. <https://ausi.github.io/respimagelint/> Herramienta que nos proporciona una auditoría de las imágenes y posibles soluciones.

Para usar la herramienta, arrastraremos el botón a la barra del navegador. Posteriormente lo seleccionaremos en la web que queramos analizar.



3. Tiempo de decodig.
4. Ventajas de SVG con respecto a las imágenes Bitmap: resolución independiente ( responsive), ahorro de peticiones si incluimos svg en código, sprites...

## 4.2 Carga y optimización de fuentes web



Si tu sitio o aplicación web utiliza fuentes web, controlar cómo se cargan puede ser muy importante para mejorar el rendimiento percibido por tus usuarios. La nueva **propiedad font-display** disponible para **@font-face** permite a los diseñadores controlar cómo se muestra el sitio web en función de cuánto tardan en descargarse las fuentes web.

Uso de **Fallback en las fuentes**.

La mayoría de navegadores siguen esta **estrategia**: **se espera un tiempo breve para descargar las fuentes y después de ese tiempo, si la fuente web no está disponible se usa otra fuente alternativa (fallbacks)**. El **problema** es que existen pequeñas variaciones en este comportamiento según el navegador y **efectos FOUT**

Efecto **FOUT** en las fuentes ("Flash of Unstyled Text") es lo que veras a menudo en ese breve momento antes de que un navegador haya tenido la oportunidad de cargar y aplicar fuentes web. Ejemplo: <https://fvsch.com/typekit-url-cache>.

Chrome y Firefox esperan 3 segundos y después muestran la fuente por defecto.

## Los períodos de descarga de las fuentes web, opciones de la propiedad font-display.

La propiedad **font-display** es en realidad un descriptor de @font-face, por lo que debes definirlo en el interior de la declaración @font-face que quieras modificar. Ejemplo:

```
@font-face {  
    font-family: 'Arvo';  
    font-display: auto;  
    src: local('Arvo'), url(https://fonts.gstatic.com/s/arvo/v9/rC7kKhY-eUDY-  
    uclSTIf5PesZW2xOQ-xsNqO47m55DA.woff2) format('woff2');  
}
```

- **Auto:** Indica que debe utilizarse la estrategia por defecto del navegador. La mayoría de navegadores utilizan hoy en día una estrategia similar a block.
- **Block:** Esta estrategia incluye un período «block» corto (la recomendación son 3 segundos) y un período «swap» infinito. En otras palabras, el navegador muestra texto invisible hasta 3 segundos y cambia a la fuente web en cuanto está disponible, por lo que se produce un parpadeo visible.
- **Swap:** Esta estrategia incluye un período «block» de cero segundos y un período «swap» infinito. En otras palabras, el texto se ve nada más cargar la página (no hay texto invisible) pero se cambia a la fuente web en cuanto está disponible, por lo que también se produce un parpadeo visible.
- **Fallback:** Esta estrategia incluye un período «block» muy corto (se recomiendan 100 milisegundos o menos) y un período «swap» corto (se recomiendan 3 segundos). En otras palabras, si la fuente web no se carga rápido, se utiliza la fuente por defecto. Después se espera un tiempo corto a cargar la fuente web y si no se carga, se sigue utilizando la fuente por defecto hasta que se cambie de página.

Podemos comparar la diferencia entre una fuente fallback y la fuente Web Font, si son parecida, el efecto FOUT no será muy exagerado. [Enlace](#)

## Font style matcher

If you're using a web font, you're bound to see a flash of unstyled text (or FOUT), between the initial render of your websafe font and the webfont that you've chosen. This usually results in a jarring shift in layout, due to sizing discrepancies between the two fonts. To minimize this discrepancy, you can try to match the fallback font and the intended webfont's x-heights and widths [1]. This tool helps you do *exactly* that.

Fallback font Georgia	Web font Merriweather
	<input checked="" type="checkbox"/> Download from Google Fonts, or: <a href="#">Upload font</a>
Font size: 16px	Font size: 16px
Line height: 1	Line height: 1