Insoportablemente rápido

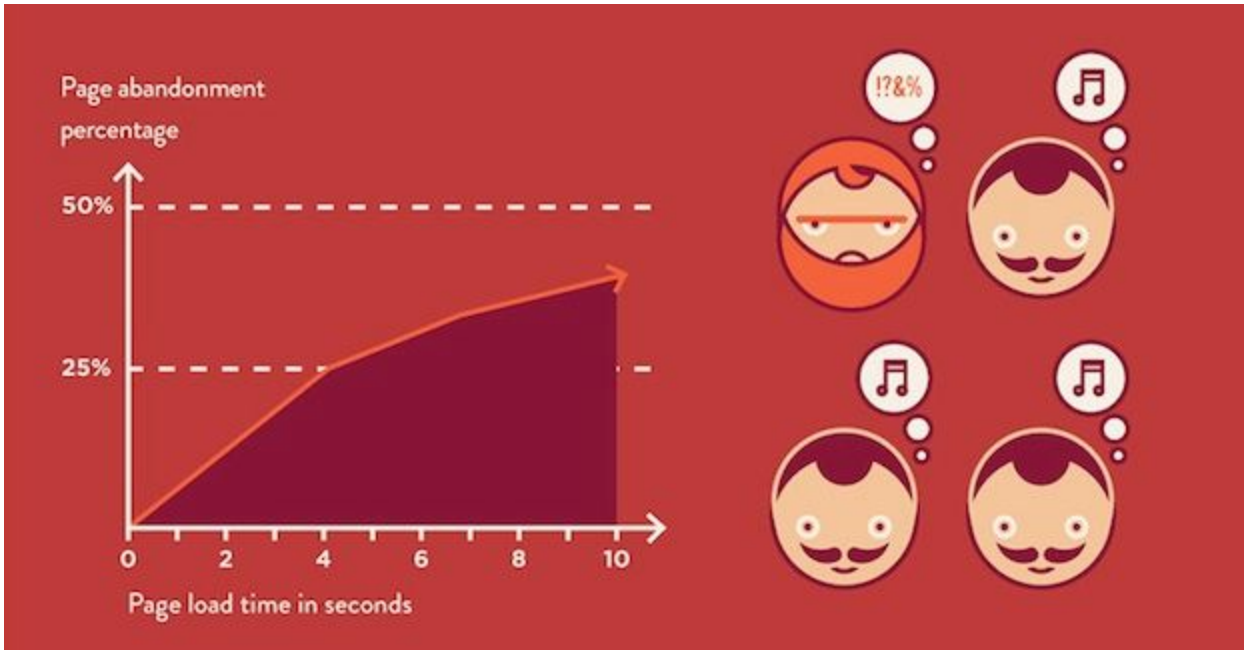
¿Porqué esto es importante?

Casos que ejemplifican:

El caso Walmart: de entre sus análisis sobre performance y UX, Walmart descubrió que:

- Por cada un segundo que mejoraban el tiempo de carga, se producia un 2% en la tasa de conversión (esta tasa se refiere al porcentaje de usuarios que toman una acción sobre nuestra web. La más común es el porcentaje de usuarios que deciden comprar algo, pero también podemos hablar sobre el tiempo que pasan en el sitio leyendo cierto artículo por ej)
- Mejorar estos ítems les generaba mayor retorno de los usuarios.
- Por cada 100 ms de mejora, incrementaron en un 1% sus ganancias.

Amazon: Cómo un segundo le puede costar a Amazon 1.6 billones en ventas.



Según un estudio de OnlineGraduatePrograms.com, una de cada cuatro personas abandona un sitio web, si su página tarda más de cuatro segundos para cargar. Dice el estudio que, cuatro de cada 10 estadounidenses abandonan el acceso a un sitio móvil que no se puede cargar en sólo tres segundos. Loco, dado que los sitios de compras tienden a ser “image-centric”, por lo que pueden tardar más tiempo en cargarse.
El cincuenta por ciento no volvería de nuevo a un sitio que los mantuvo esperando algo. Así que es mejor que puedan experimentar rapidez la primera vez si se desea que vuelvan.

Amazon calcula que una desaceleración de carga de página de sólo un segundo podría costar $ 1.6 billones en ventas cada año. Google ha calculado que al disminuir sus resultados de búsqueda por tan sólo cuatro décimas de segundo les podría costar 8 millones de búsquedas al día, que serían muchos millones menos de anuncios en línea por día.
El 85% de los usuarios de Internet esperan que los sitios carguen en sus teléfonos igual o más rápido que en la computadora. Hasta ahí, casi todo conocido, pero la cosa se pone compleja cuando nos enteramos que el 57% de ellos tiene problemas para hacerlo y la causa principal : un tercio de los usuarios se queja de la lentitud al cargar las páginas.
ver(http://blogs.tn.com.ar/dev/como_sobrevivimos_al_mobilegeddon_y_los_12_jinetes_del_seo_movil/)

AOL: experimentó causas de pérdida de usuarios por delays. Específicamente en la reducción de page views. El top ten de visitantes se dio en las páginas más rápidas páginas con una mejora de 7.5 % de páginas por visitantes. En la parte inferior del ranking se notó que las más lentas crearon una disminución del 5% de visitas.
Su conclusión: cuanto más lento su sitio, menos page views.

Qué es Pagespeed Insights

- **Definición:**
    - Page Speed Insights mide el rendimiento de una página para dispositivos móviles y dispositivos de escritorio. Se chequea la url dos veces, una vez con un user-agent móvil, y una vez con un user-agent de escritorio del usuario.
    - El Score de PageSpeed va de 0 a 100 puntos. Una puntuación más alta es mejor y una puntuación de 85 o más indica que la página está funcionando bien. PageSpeed Insights se mejora continuamente es por eso que la puntuación puede cambiar, ya que se añaden nuevas reglas o mejora su análisis.
    - PageSpeed Insights solo considera los “network-independent” aspectos sobre rendimientos de web: la configuración del servidor, la estructura HTML de una página, y su uso de los recursos externos, tales como imágenes, JavaScript y CSS. La implementación de las sugerencias debería mejorar el rendimiento relativo de la página. Sin embargo, el rendimiento absoluto de la página todavía será dependiente de la conexión de red de un usuario.
    - PageSpeed Insights se centra en mejorar el rendimiento móvil
    - PageSpeed Insights mide cómo la página puede mejorar su desempeño en:
        - time to above-the-fold load: El tiempo transcurrido desde el momento en que un usuario hace una request hasta el momento en que el primer contenido se hace visible por el navegador.

- time to full page load: El tiempo transcurrido desde el momento en que un usuario hace una request a una página hasta el momento en que la página está totalmente cargada por el navegador.

- Reglas de PSI:
  - Temas de velocidad
    - ***Avoid landing page redirects:*** las redirecciones generan delays en la carga del sitio y mientras estas ocurren nada es mostrado al usuario. En muchos casos las mismas pueden ser eliminadas sin cambiar el funcionamiento de la web. Si tu web requiere de redirecciones hacelas desde el lado del servidor, con esto minimizas el tiempo de las llamadas RTT (Round-time-trip): contabiliza el tiempo desde que el usuario envía una request hasta que el servidor responde a través de la red. Incluye el tiempo de ida y vuelta en la red pero excluye el tiempo de decarga completa de los bytes. Ver más en: https://gtmetrix.com/avoid-landing-page-redirects.html
    - ***Eliminate render-blocking JavaScript and CSS in above-the-fold content:*** Por defecto, la ejecución de JavaScript es "parse-blocking": cuando el navegador se encuentra con un script en el documento debe hacer una pausa en la construcción del DOM, entregar el control a la ejecución del JavaScript y dejar que la secuencia de comandos se ejecute antes de finalizar con la construcción del DOM.
      - Añadiendo la palabra clave async al tag script indicaremos al navegador que no debe bloquear la construcción DOM mientras espera que el script esté disponible. ver más en: https://developers.google.com/web/fundamentals/performance/critical-rendering-path/adding-interactivity-with-javascript.
      - Cargar el CSS asincrónico: Algunos navegadores dejan al usuario con una página en blanco mientras esperan que el archivo se descargue. Sin embargo, aquí es donde el plugin loadCSS entra en acción. LoadCSS cargará el CSS de forma asíncrona, lo que significa que su página web no se bloquerá en su carga mientras que se está esperando para descargar y mostrar el CSS. Ver más en: http://deanhume.com/home/blogpost/loading-css-asynchronously/7104
      - https://github.com/filamentgroup/loadCSS

- ***Leverage browser caching:*** Obtener recursos en la red es lento y caro: la descarga puede requerir múltiples roundtrips (idas y vueltas) entre el cliente y el servidor, lo que retrasa el procesamiento y puede bloquear el rendering del contenido de la página, y también incurre en costos de bajada de datos para el visitante. Todas las response del servidor deben especificar una directiva de almacenamiento en caché (caching policy) para ayudar al cliente a determinar cuándo puede volver a utilizar un recurso previamente descargado. Entre tantas posibilidades Google nos habla de dos:
    - Cache-Control define cómo y por cuánto tiempo la response puede puede ser cacheada por el navegador y otros caches intermedios.
    - ETag ofrece un token de revalidación que se envía automáticamente por el navegador para comprobar si el recurso ha cambiado desde la última vez que fue solicitado. Para ver más: https://developers.google.com/speed/docs/insights/LeverageBrowserCaching
    - http://httpd.apache.org/docs/2.2/mod/mod_expires.html

- ***Minify CSS, JS, HTML:*** Minificar se refiere al proceso de eliminación de datos innecesarios o redundantes sin afectar a cómo el recurso es procesado por el navegador - por ejemplo, comentarios de código, formateo del mismo, la eliminación de código no utilizado, utilización de nombres de variables y funciones más cortos, y así sucesivamente.
    - Ver más en: https://developers.google.com/speed/docs/insights/MinifyResources.
    - http://yui.github.io/yuicompressor/
    - https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer#minification-preprocessing--context-specific-optimizations

    - Diferencia entre GZIP y Minificar: Minificación hace cosas como la eliminación de espacios en blanco, la eliminación de los comentarios, la eliminación de puntos y comas no necesarios, reduce longitudes de código hexadecimal y cosas por el estilo. El archivo permanece con código perfectamente válido. No se podra leer como developer o trabajar con él, pero no se está rompiendo ninguna regla. El navegador puede leerlo y usarlo al igual que pudo hacerlo con el archivo original.

    - Minificación crea un nuevo archivo. Por ejemplo, si se tiene un `style.css` entonces se podría minificar y utilizar como `style.min.css`.

    - En cambio, Gzipping encuentra todos los strings repetitivos y los reemplaza con punteros a la primera instancia de ese string.
    - Esto puede ser muy eficaz para reducir el tamaño del archivo, sobre todo con el código, ya que el código es muy repetitivo.
    - En la web, gzipping se realiza directamente por el servidor. Es una cuestión de configurar el servidor para hacerlo. Una vez hecho esto, gzipping automáticamente sucede, no hay ningún trabajo en curso que se tenga que hacer. El servidor comprime el archivo y lo envía a través de la red así. El navegador recibe el archivo y lo “unzipea/deszipea” antes de usarlo. No hay mucho estudiado acerca de la sobrecarga de compresión y descompresión, por lo que sólo suponemos que es insignificante y los beneficios son óptimos. En apache (web server) hablamos de configurar el “mod_deflate”.
    - Ver nota completa en https://css-tricks.com/the-difference-between-minification-and-gzipping/

    - Tenemos por otro lado el concepto conocido como “uglify”: transformar el código en una forma ilegible, renombrando variables y funciones para esconder la funcionalidad. Es no reversible.
    - Ver: http://stackoverflow.com/questions/19694448/grunt-whats-the-difference-between-concat-and-uglify-and-minify.

- ***Prioritize visible content:*** La recomendación de PageSpeed es dividir el CSS en dos partes; una parte “inline” que es responsable para estilizar el “above-the-fold” y el resto, que puede ser diferida o cargada asincrónica.
    - Determinar qué partes de nuestro CSS son críticas necesita de la inspección de la web en tamaños "móviles" y "de escritorio", evaluando sobre los elementos de la primera vista y las reglas CSS. Esto parecía una tarea de enormes proporciones, pero hay herramientas que lo hacen:
    - http://jonassebastianohlsson.com/criticalpathcssgenerator/
    - https://gist.github.com/PaulKinlan/6284142
    - https://github.com/fatso83/grunt-penthouse
    - ver más: https://css-tricks.com/authoring-critical-fold-css/

- **Reduce server response time:** desde el front-end es lo menos manejable pero veamos algunos tips:
    - Google dice: "Se debe reducir el tiempo de respuesta del servidor debajo de los 200 ms".
    - Factores de tiempo de respuesta del servidor:
    - **Cuatro cosas principales se evalúan para determinar el tiempo de respuesta del servidor:**
    - **Uso de recursos web:** Si cada una de sus páginas web utilizan menos recursos, se podría mejorar el tiempo de respuesta del servidor y no gastar dinero
    - **Web server:** Si se cambia de software o configuración del servidor web es probable que se pueda mejorar el tiempo de respuesta del servidor.
    - **Web hosting**: Si se mejora la calidad y el alcance del alojamiento web podría mejorarse el tiempo de respuesta del servidor, pero seguramente implicará gastar dinero
    - **Cosas que sí podríamos encarar como developers:**
    - **Combinar archivos CSS externos:** Todos el CSS puede estar en un archivo con lo que la página estaría llamando menos recursos por página.
    - **Combinar archivos javascript externos:** como el CSS, los javascripts que utilizamos podrían ser ubicados en el html o en un archivo js combinado. Con demasiada frecuencia no lo son y esto crea llamadas externas innecesarias.
    - **Lazy load / defer imágenes:** el usar el lazy load permite a la web que se muestre de forma rápida sin esperar a la descarga de cada imagen.
    - ver más en: https://varvy.com/pagespeed/improve-server-response.html.

- **Temas de UX**
    - ***Evitar Plugins:*** Cuando hablamos de plugins, nos referimos a aquellos que ayudan al browser a procesar determinado tipo de contenido, como por ejemplo el Flash Player. Siendo que gran parte de los teléfonos móviles no soportan este tipo de plugins y que muchos pueden causar cuelgues, crashes y problemas de seguridad. La recomendación es evitar estas llamadas en los contenidos para móviles y usar las alternativas nativas de los browser y dispositivos, de manera de asegurarnos de que nuestro contenido esté disponible para todos los dispositivos.
    - ***Configurar el viewport:*** El viewport controla cómo se muestra un sitio en un dispositivo móvil, sin eso los browsers lo

renderizarían como si se tratara de la pantalla de un monitor, reduciendo el contenido para que entre en la pantalla. Por eso configurarlo resulta súper importante, usando este tag en el head: `<meta name=viewport content="width=device-width, initial-scale=1">` Para tener en cuenta, las pantallas de los dispositivos no solo varían en sus dimensiones, sino también en la densidad de píxeles. El viewport permite al browser interpretar la relación entre las distintas formas de entender los pixels, es decir la relación los pixels físicos de la pantalla del dispositivo y los pixels establecidos como unidad de medida en nuestro CSS. DIP (por sus siglas en inglés *Device-independent pixel*) es la escala que permite compatibilizar estas medidas en una referencia uniforme, de manera que se vean similares a simple vista. Más en: https://en.wikipedia.org/wiki/Device_independent_pixel, https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag, https://docs.webplatform.org/wiki/tutorials/understanding-css-units

- ***Todo el contenido visible en el viewport:*** En general, tanto en móviles como en desktop, los usuarios están acostumbrados a scrollear en forma vertical. Forzar al usuario a hacer un scroll horizontal para ver el contenido como resultado de un mal diseño resulta en una experiencia frustrante. Por supuesto, siempre existe la posibilidad de desafiar lo establecido, siempre y cuando la intención sea manifiesta y clara para los usuarios y que aporte a la experiencia general y donde no quede contenido oculto por error. Lo ideal para evitar el scroll horizontal, es privilegiar el uso de anchos y posiciones relativas. CSS media queries nos pueden ayudar a hacer ajustes finos para distintos anchos y posiciones de pantalla. Ver algunos consejos en: http://www.sitepoint.com/10-ways-make-website-mobile-friendly/
- ***Fuentes con tamaño legible:*** Considerando que los usuarios que consumen nuestros contenidos en dispositivos móviles, es necesario prever que en muchos casos, ese consumo será interrumpido e intersticial. Por eso, el tamaño de los textos es crucial para permitir retomar la actividad y realizar las acciones principales fácilmente en cualquier contexto en que se encuentren. La recomendación es establecer un tamaño base de 16px CSS y luego establecer a partir de eso relaciones de escala para los distintos formatos y necesidades. Otras cuestiones a tener en cuenta, además de los tamaños, son la economía de familias tipográficas a usar y el buen contraste. Ver más consejos en:

http://chaione.com/why-the-font-should-you-care-typography-in-mobile-apps/