esultat d'imatges per a "uoc icon"HERRAMIENTAS HTML & CSS

­

**PEC 3.**

**RENDIMIENTO WEB**

Sergi Rourera Llauradó

2n cuatrimestre – 08/01/2021

Trabajo realizado para la asignatura **HERRAMIENTAS HTML & CSS**

Universitat Oberta de Catalunya

El © del trabajo sigue siendo de los autores. No se permite la reproducción total o parcial de este documento si no se cita explícitamente su procedencia.

Índice

Introducción 4

Informe de optimización 4

Tabla con Regular 2G 8

Preguntas de teoría X

Conclusiones X

Introducción

Toda la PEC 3 se ha basado en el código obtenido de la PEC 1 + PEC 2. Donde se consiguió crear una plataforma de libros sobre un boilerplate realizado en los módulos de la asignatura y que configura varios plugins y optimizaciones eficientes para el sitio web.

Toda la infraestructura del proyecto se levanta en Netlify, que proporciona un *Continuous Deployment* manejado desde Github. El código *live* deployado se encuentra en la siguiente URL:

<https://musing-panini-9fe2a9.netlify.app/>

En todo momento se ha utilizado el control de versiones Github para desarrollar, así pues, todas las optimizaciones realizadas en esta práctica se pueden encontrar en el repositorio siguiente, junto con todo el background realizado en la PEC1 y PEC2:

<https://github.com/srourera/PEC3-HTML-CSS>

Informe de optimización

**Primer análisis:**

Se ha analizado el sitio web utilizando Google PageSpeed Insights, con un resultado bastante óptimo de **84** en mobile y **95** en desktop.

Aún con este resultado, la plataforma de análisis indica que la página tarda unos 3 segundo en mostrar el primer contenido, lo cual marca como naranja, ya que podría optimizarse.

Aunque la plataforma de análisis no ha mostrado demasiadas propuestas de mejora, si que propone eliminar aquellos recursos de bloquean el renderizado, así como CSS y JavaScript. Además, indica la posibilidad de optimizar mediante los tamaños de imagen y el formato de las imágenes.

Por otro lado, sugiere también utilizar la propiedad *font-display* de CSS para hacer que el texto aparezca, aunque no se haya descargado la fuente.

Después de revisar en Network el *timeline* (imagen 1), se ha decidido intentar utilizar *font-display* para las fuentes (excepto font-awesome) e intentar hacer la carga del script de font-awesome de forma asíncrona para evitar este primer bloqueo en el renderizado.

En caso de que el renderizado también dependiera de las fuentes de google (por culpa del script de font-awesome no queda claro en el *timeline*) también se intentarían importar de forma asíncrona.

Gráfico

Descripción generada automáticamente

Imagen 1

**Segundo análisis:**

Tras modificar el script de font-awesome, haciéndolo asíncrono, y de aplicar font-display a las fuentes para evitar el bloqueo del texto hemos visto que las imágenes y otros recursos empezaban a descargarse mucho antes, ya que no tenían que esperar al fichero js.

Imagen que contiene Tabla

Descripción generada automáticamente

Con el cambio, en el análisis de Google PageSpeed Insights, se ha subido a 96 en desktop, sin embargo, ha bajado a 84 en mobile y aún no está claro el motivo.

Bien es cierto, que el análisis ha dejado de avisarnos del bloqueo de ficheros js en el renderizado, e incluso ha dejado de sugerirnos que apliquemos el font-display en las fuentes.

Aún así, sigue sugiriendo que quitemos el import bloqueante de css, con lo cual no estoy tan de acuerdo ya que un primer renderizado sin estilado que acaba transformando a la página final con estilo, me parece poco bonito. No se intentará optimizar demasiado por esta rama, ya que de todos modos, es de los procesos que menos afecta ya que tiene un tiempo de respuesta pequeño.

Por otro lado, se ha decidido probar de quitar el plugin de google-font para postcss para evitar importar la fuente desde css y poder hacerlo directamente desde el html, para poder paralelizar el proceso.

**Tercer análisis**

Tras quitar el plugin de google-font, la carga de las fuentes se paraleliza:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En este caso, pero, el análisis de Google PageSpeed Insights sigue indicándonos la misma puntuación.

Por último, sigue indicando que las imágenes son demasiado grandes, y aunque ya se realizó en la PEC anterior toda la gestión de imágenes responsivas, se le va a echar un vistazo, quizá intentando cargarlas de forma *lazy*.

**Resultado final**

El resultado final tras aplicar lazyload a las imágenes (aunque hay ciertos navegadores que no lo soportan y simplemente lo ignorarán), el resultado ha acabado siendo bastante favorable con un **87** en mobile y un **98** en desktop.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente

Tabla con Regular 2G

.

:

Preguntas de teoría

**Describe, de forma esquemática, cuáles son las principales fases de renderizado de una web.**

.

:

**Desde un punto de vista de rendimiento, ¿tiene más sentido situar los tags STYLE dentro del HEAD de la web, o dentro del FOOTER? Y desde el punto de vista del desarrollador, ¿cuáles son los compromisos que ello implica?**

.

:

**¿Y en el caso de los elementos SCRIPT?**

.

:

**¿Qué diferencia hay entre los valores de las columnas "transferred" y "Size" en la pestaña Network de las herramientas de desarrollo de Firefox Developer Edition? ¿Qué otras columnas son útiles para las personas encargadas de mejorar el rendimiento web?**

.

:

Conclusiones

.

: