PEC 1.
DESARROLLO DE UNA WEB

Sergi Rourera Llauradó

2n cuatrimestre – 15/11/2020

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

Universitat Oberta de Catalunya

Índice

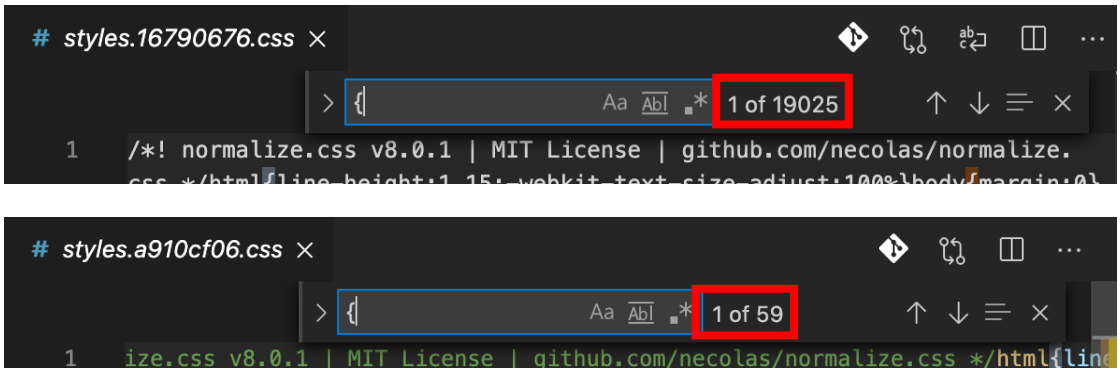
Inicialización del proyecto

En primer lugar, se han realizado las actividades del modulo dos y se han utilizado éstas como *boilerplate* para iniciar la PEC. Los requisitos especificados en las actividades del módulo incluían desarrollar un boilerplate desde cero con ***Parcel***, beneficiándonos de la fácil configuración que ofrece, a la cual se le ha añadido la dependencia *rimraf* para borrar (y de este modo actualizar) los ficheros de la carpeta que genera el *build*. Usando la dependencia *npm run all* se han creado scripts en package.json con tal de automatizar el proceso de actualización del *build*.

Al *build* de Parcel se le ha configurado el preprocesador de código css ***PostCSS***, que puede utilizar plugins, como *autoprefixer*, que se utiliza para mantener la compatibilidad con los navegadores especificados. Otro plugin utilizado ha sido postcss-custom-properties que se utiliza para transformar las variables nativas de CSS en propiedades literales.

Adicionalmente, y un poco como experimento personal (que espero que no me salga mal), ya que me interesa profesionalmente (en mi próximo trabajo se utiliza), he querido usar también el framework CSS ***tailwindcss***, que puede configurarse también como plugin de *PostCSS*. Reduce en gran cantidad el código programado en *css*, pero al final esta asignatura va de herramientas CSS (código en si ya lo hicimos), así que espero que no haya tomado una mala decisión al probar de integrarlo. Este framework está empezando a ser cada vez mas conocido dentro del desarrollo frontend y consiste en un super set de reglas CSS (configurables, o mejor dicho sobreescribibles, al ser CSS y ejecutarse en cascada) que permiten desarrollar rápidamente código HTML estilado haciendo uso de clases generales con estilados (un ejemplo seria *p-6*, que es una clase que añade cierto *padding*; o *py-3*, que añadirá más *padding*, pero únicamente arriba o abajo).

Tailwindcss permite purgar y quitar el css no utilizado, así que en el procesado de *PostCSS* solo adjuntará aquellas reglas necesarias. Adjunto la comparativa: la primera imagen muestra el numero de reglas CSS añadidas en nuestro fichero CSS procesado sin ninguna propiedad especificada para purgar el código innecesario; y a la segunda, especificando la propiedad *purge['*.html']* en el fichero de configuración de *tailwindcss*.



Para añadir fuentes de google se ha utilizado el plugin de *PostCSS* llamado *postcss-google-fonts*, el cual con la sintaxis `@google-font 'fuente deseada' opciones`, al procesarlo, genera el *import* necesario para que la fuente esté disponible en el sistema.

Por último, por gusto personal, agradezco trabajar con preprocesadores CSS como *scss*, así que he instalado el paquete postcss-scss que actúa como *parser* de *scss* en los ficheros de estilos que procesa *PostCSS*. De este modo podemos usar directamente todas las ventajas de *scss*, como principalmente la anidación.

Estructura

Para estructurar correctamente el código, se ha creado una carpeta *src* que contendrá el código fuente del sitio web: páginas *html*, estilos *scss* y ficheros *js*, cada uno en su carpeta correspondiente. Adicionalmente contiene una carpeta para las imágenes y logos.

La carpeta de *js* y la de *css* pueden contener más de un fichero, pero siempre va a haber un fichero principal que importa los demás, ya que, de este modo, Parcel unifica los ficheros en uno sólo y evitamos realizar más peticiones en producción.

Los ficheros *html* van a compartir siempre tanto el `<header>` como el `<footer>`, y lo que se modificará entre páginas será el contenido del `<main>`, separado por secciones con `<section>`.

Fuera de esta carpeta *src* se encuentran todos los ficheros de configuración tales como el *package.json*, el fichero de configuración *.postcssrc*, etcétera.

Entornos

Se han diferenciado dos entornos: desarrollo y producción. Ambos procesan el código de forma similar, tomando los mismos puntos de partida con *Parcel* y usando *PostCSS* con sus *plugins*.

Las dos únicas diferencias que existen entre desarrollo y producción son, en primer lugar, que para el entorno de desarrollo, *Parcel* levanta un servidor local y se queda observando los cambios en los ficheros para volver a *buildear* en cuanto hay un cambio, mientras que en producción únicamente se ejecuta el *build* de la aplicación; y en segundo lugar, que para el entorno de desarrollo se generan *sourcemaps* para poder depurar el código en el navegador, mientras que en producción no se generan.

Ambos scripts limpian el código procesado anteriormente antes de volver a procesarlo.

Git y Servidor público

En todo momento se ha utilizado el sistema de control de versiones Git. Básicamente se ha generado un repositorio tanto local como remoto que almacena todo el código y su control de versiones:

https://github.com/srourera/PEC1-HTML-CSS

Se ha desarrollado encima de la rama máster, y en cuanto se ha adquirido el *boilerplate* inicial, se ha creado una rama que parte de allí, para tener el *boilerplate* disponible fácilmente para futuros proyectos.

Se ha integrado Git con el proveedor de hosting *Netlify*, que permite crear un servidor público gratuito y proporciona un servicio de *CD* (*Continuous Deployment*), es decir, cada vez que se publica un cambio en la rama máster en el repositorio remoto, *Netlify* ejecuta `npm run build` y publica el directorio *dist* en la siguiente URL pública:

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

Diseño

El diseño del sitio web se ha enfocado con el paradigma *mobile-first*. Realizarlo de este modo no ha requerido mucho esfuerzo, ya que *tailwindcss* ya está enfocado a *mobile-first*, proporcionando clases que implementan *mediaqueries* con *min-width*.

Con *tailwindcss* el fichero *css* se libera en gran medida haciendo uso de sus clases reusables, sin embargo, el código *HTML* queda menos limpio y cada etiqueta *html* contiene bastantes clases. Por ese motivo, *tailwindcss* se considera un framework *utility-first*. Con esto me excuso diciendo que, desde mi punto de vista, no considero que sea la manera más recomendada de programar, ya que quizá añade complejidad a la lectura del *HTML*, sin embargo, quería darle una oportunidad porque está empezando a pedirse cada vez más en las oportunidades laborales.

Las decisiones de diseño que se han ido tomando a lo largo del desarrollo de las páginas, se comentan en los apartados de cada página.

Imágenes

Las imágenes de portadas de libros se han obtenido de portales de literatura (Casadellibro, MegustaLeer, y Amazon) cuya intención no ha sido otra que realizar esta práctica de estudios universitarios. Por otro lado, la imagen del fondo de la pantalla principal se ha obtenido de Unsplash , que es un portal de imágenes de calidad y gratuitas.

Secciones compartidas

Como se ha comentado anteriormente, hay secciones del código *html* que se comparten en todas las páginas, como es el caso del `header` y del `footer`.

Se ha optado por un diseño ciertamente minimalista con tonos grises y negros. Se ha considerado oportuno encapsular la página web con un `header` de fondo negro por arriba, y un `footer` de fondo negro por abajo. Sin mucha complejidad en ninguno de los dos elementos, el `header` consta del logo seguido del nombre del sitio a un lado, y al otro lado un menú que solo aparece en dispositivos de considerable anchura y que hace scroll a la sección que corresponde; el `footer`, por otro lado, solo contiene el nombre del desarrollador y un apunte del origen de las imágenes utilizadas.

Tanto el `header` como el `footer` se han reutilizado exactamente igual y en la misma posición para todas las páginas, sin embargo, éstos no son los únicos elementos que se han compartido entra páginas. Hay otro 'componente' que contiene las portadas de los libros al que se le han aplicado unos estilos específicos para mantener el *aspect ratio* deseado y para sobreponer una imagen que provoca un efecto de libro encima de la imagen real de la portada. Este 'componente' se ha utilizado en todo momento en que se quería mostrar la portada de un libro: tanto en la sección principal como en cualquiera de los muchos listados de libros implementados en el sitio, y adicionalmente en la página de detalle del libro.

Página principal

La página principal es la que más tiempo ha requerido. Consta de tres secciones de contenido: una parte superior con una imagen de fondo y un texto con una llamada a la acción y el libro del mes actual; seguidamente un listado de los últimos 4 meses, que puede expandirse para mostrar los últimos 8, e incluso un botón para acceder a la página de todos los meses; y, por último, una sección de enlaces a los diferentes géneros (o mejor dicho subgéneros) de los libros recomendados.

Básicamente, la primera sección es la que más tiempo ha requerido y más problemas ha dado. Consta de una imagen grande que ocupa toda la anchura de la pantalla incluso cuando el dispositivo es muy grande. El principal problema ha sido con la altura de la imagen, ya que inicialmente se quería hacer que siempre ocupara toda la altura de la pantalla, pero eso ha traído muchos problemas ya que, en dispositivos móviles, el 100% de la pantalla varia (ya que la barra de navegación se oculta al hacer scroll). Además, el comportamiento de `height:100%` es diferente en Safari y en Chrome. Finalmente se ha decidido hacer el comportamiento deseado inicial para dispositivos grandes y para dispositivos móviles el `height` se calcula en función del contenido.

A continuación, se encuentra un grid de 4 responsivo de los libros mensuales más recientes con el 'componente' explicado anteriormente para mostrar las portadas, con un enlace a la página de todos los libros. Y, por último, una sección que contiene un listado de subgéneros que redirigen a las páginas de cada uno de los subgéneros, que contiene el listado de libros mensuales filtrado por el subgénero en cuestión.

Página todos los libros

Esta página es una funcionalidad extra añadida para extender la sección de la página principal que contiene los libros de los últimos meses. Se ha considerado oportuno que, dado que la web recomienda un libro cada mes, exista un historial de todos los libros recomendados desde el inicio de la plataforma.

Esta página contiene, nada más y nada menos, que un listado ordenado de todos los libros recomendados mensualmente en un grid responsivo y utilizando el 'componente' para las portadas de los libros.

Página de subgénero

La página de cada subgénero es muy parecida a la página de todos los libros. De hecho, la única diferencia es que, en la página de subgéneros, para hacer una diferenciación y que el usuario inconscientemente se pudiera ubicar mejor dentro del sitio, se ha encapsulado el listado de los libros filtrados dentro de un *box* un poco más oscuro.

Únicamente se pedían dos páginas de categorías, pero al declararse 8 enlaces en la página principal (tampoco cobraba mucho sentido, para mi, sólo tener enlace a dos géneros), como el código era realmente casi el mismo y no se necesitaba cambiar contenido de texto (tipo párrafos explicando en que consiste el género, ni nada de eso), ya que lo único que cambia son los libros que aparecen, finalmente se han implementado las 8 páginas.

Página de detalle del libro

No se ha hecho lo mismo con las páginas de detalle del libro, no se han implementado todos los libros (hay 20 en la página de todos los libros). A diferencia de las páginas de subgénero, las páginas de detalle si que incluían bastante contenido de texto y de información específica del libro que tendría que haberse hecho por cada uno. En el enunciado pedía 2 páginas de categoría y para cada una de ellas, una página de detalle del libro, es decir, 2 páginas de detalle en total. Finalmente se han realizado 8 páginas de detalle, una para cada subgénero (el primer libro del subgénero).

La página de detalle de cada uno de los libros, realmente se enfoca un poco más al mes en que nos encontramos que en el libro en sí (título y *url* de la página y sección principal hacen referencia al mes).

La primera sección añade un párrafo introductorio explicando el motivo de la recomendación del libro, con el mes en sí, como título.

Justo debajo encontramos la portada del libro flotando a la izquierda, con su sinopsis a la derecha y su ficha técnica debajo. La solución final, después de hacer diferentes pruebas con *inline* y con *flex*, ha sido hacer la imagen flotante para que cuando la sinopsis supere la altura de la imagen, caiga a la izquierda y ocupe toda la anchura disponible.

Se ha añadido como extra, una sección con valoraciones de críticos literarios (algunas inventadas, ya que no se han encontrado las reales) y un listado de plataformas donde adquirir el libro recomendado.

Finalmente, se ha añadido una sección en que se muestran los libros del mismo subgénero con el 'componente' utilizado para mostrar las portadas de los libros.

Principalmente ha surgido un problema en este punto: tanto la página de todos los libros como la de los subgéneros, no usan ninguna clase extra para aplicar estilos que no aparezca en *index.html*, así que el problema no ha surgido hasta implementar la página de detalle. Como se explica en el apartado de ***Inicialización al proyecto***, *Tailwindcss* permite purgar (limpiar) las clases CSS que ofrece a únicamente aquellas que realmente usamos, de este modo reducimos el fichero CSS a únicamente aquello necesario. Sin embargo, tenía configurado que limpiara estos ficheros: **./src/*.html**. Ya que, al principio del proyecto, no había estructurado carpetas como /libros, que contiene los ficheros html de las páginas de detalle, así que *Tailwindcss* estaba procesando únicamente aquellas clases existentes en los ficheros de la raíz de src, por lo que muchas de las clases que añadía para el desarrollo de la página de detalle, el navegador parecía ignorarme.

Finalmente, he configurado correctamente la regla (y he visto que, de hecho, es la configuración propuesta en la documentación de *Tailwindcss*) para que procesara todos los ficheros existentes en: **./src/**/*.html**.

Conclusiones

Sin mucho más que añadir, realmente ha sido una práctica muy interesante, ya que usualmente he trabajado que frameworks y utilidades que nos lo dan todo montado, y ha sido una buena experiencia configurarlo todo desde cero.