PEC 3

Herramientas HTML y CSS II

Gerson Esquembri Moreno

10 de enero de 2020

Proceso de desarrollo

Para obtener el código, simplemente se ha clonado el repositorio provisto en GitHub (UOC Boilerplate), a un directorio local, y para empezar la práctica con código funcional, se realizó el primer commit a GitHub.

El entorno de desarrollo utilizado se compone de:

- Visual Studio Code, para el desarrollo del código.
- Git bash, para la ejecución de los comandos de compilación del código.

La versión de producción de la página se encuentra en el directorio /dist.

Para la puesta en marcha del proyecto, tenemos varias opciones:

- Podemos ejecutar el comando `npm run dev`, se compilará el proyecto y se abrirá en localhost, útil a la hora de desarrollar, ya que los cambios en el código se van mostrando en la página cada vez que se guardan.
- Con el comando `npm run build`, se compilará el proyecto para producción, poniéndose el código compilado en el directorio /dist.

En cuanto a la instalación de módulos, el módulo principal que se ha necesitado para esta práctica es *Tailwind*. Para su instalación, se han seguido los siguientes pasos:

- Para instalar el módulo en el proyecto, se ha ejecutado el siguiente comando, en la carpeta raíz del proyecto: **`npm install tailwindcss --save-dev`** .
- A continuación, se ha creado el fichero de configuración de *Tailwind,* mediante el siguiente comando: **`npx tailwind init`** .
- Seguidamente, para incorporar el código generado por *Tailwind* al proyecto, se deben incluir las siguientes directivas al principio del fichero de estilos principal, `main.css`: **`@tailwind base;`** (estilos base o globales) **`@tailwind components;`** (clases de componentes de *Tailwind*), y la siguiente al final: **`@tailwind utilities;`** (clases de utilidad de *Tailwind*).

- Por último, para compilar los ficheros CSS con *Tailwind,* se debe incluir en el fichero **`postcss.config.js`** la siguiente línea:

```
module.exports = {
  plugins: [
    require("tailwindcss")("./tailwind.config.js"),
      ...
  ]
};
```

Con estos pasos, ya tenemos listo *Tailwind* para utilizarlo en el proyecto.

Para el desarrollo del código, se ha seguido la metodología *mobile-first,* como se recomienda en la documentación de *Tailwind,* ya que a la hora de definir los tamaños de los elementos de la página para diferentes tamaños de pantalla, lo más sencillo es hacerlo definiendo primero el tamaño de los elementos para la pantalla más pequeña, y a continuación hacerlo para pantallas más grandes.

En cuanto a la personalización de ciertos colores del tema de la página, se ha modificado el fichero **`tailwind.config.css`**, añadiendo las siguientes líneas:

```
    colors: {
      'body-blue': '#1c3485',
      'custom-orange': '#ff7b57',
      'custom-green': '#beff70'
    }
```

¿Qué diferencias hay entre el enfoque de tipo CSS semántico (el que usaste en las otras PEC) al respecto del CSS de utilidades? ¿Cómo afectó esto a tu proceso de desarrollo? ¿Y a tu código?

El **CSS semántico** se basa en la aplicación de reglas de nombrado de los selectores, para definirlos según lo que son, no cómo son (es decir, por su función en la página, y no por su apariencia).

El **CSS de utilidades** se basa en la definición de clases CSS pequeñas y con un solo propósito, con nombres basados en su función visual. Por ejemplo, *w-1/3,* se utilizaría para definir que el ancho del elemento sobre el que se aplica la clase es un tercio del ancho del elemento padre.

El proceso de desarrollo, gracias al *Utility CSS,* ha sido bastante simple, ya que no hay que preocuparse por la creación de selectores de estilos para cada elemento, sino simplemente aplicar los que nos proporciona *Tailwind.* Dicho esto, el código se simplifica muchísimo, ya que no es necesario crear ningún fichero CSS, más allá del principal en el que se incluyen las librerías de *Tailwind*.

¿Qué diferencias encontraste entre usar una librería de componentes y una librería de utilidades?

Una librería de componentes es, básicamente, una carpeta basada en la nube, que contiene elementos prediseñados para incluir en una página web, por ejemplo. Dichos componentes suelen ser elementos “completos” (por ejemplo, un botón, con un tamaño, color, efectos... predefinidos).

Una librería de utilidades se reduce a clases CSS más simples, con una única función. Por ejemplo, siguiendo el caso anterior de un botón, en vez de disponer del botón completo, podríamos aplicar propiedades como el tamaño, la posición, colores, etc, por separado al mismo.

¿Qué componentes decidiste extraer y por qué?

Los componentes extraídos con la directiva **@apply** de *Tailwind,* son:

```
.member-name {
  @apply text-custom-green font-bold text-xl;
}
.member-role {
  @apply text-custom-orange font-semibold;
}
.link {
  @apply w-full text-right text-custom-orange font-bold text-xl;
}
@screen md {
  .link {
    @apply w-1/5;
  }
}
.member-desc {
  @apply w-4/6 text-center p-2;
}
```

Se han escogido los mismos ya que se repiten bastantes veces en la página de miembros del grupo (nombres, roles y descripciones), y los links del encabezado y pie de página, que aparecen en todas. En el caso de los links, para que sean *responsive,* se ha aplicado la directiva **@screen,** para poner el *breakpoint* en las pantallas de tamaño medio (*md*), y la directiva **@apply** sobre el ancho de los links.

Link a Netlify

https://romantic-sinoussi-223675.netlify.com/

Link al repositorio de GitHub

https://github.com/gersonesmo/HerramientasHTML2PEC3.git