Herramientas HTML y CSS - PEC 1

30 de octubre del 2020

Proceso de desarrollo

Primeros pasos y configuración

En primer lugar, he creado un directorio nuevo para este proyecto. Ahí he instalado el UOC boilerplate desde el repositorio https://github.com/uoc-advanced-html-css/uoc-boilerplate/

Para hacerlo, he clonado el repositorio mediante:

```
git clone  https://github.com/uoc-advanced-html-css/uoc-boilerplate/
```

Para todas las operaciones relacionadas con Git, utilizo el terminal de Git Bash para Windows.

Estoy utilizando Windows en lugar del entorno de Linux ya que los comandos de npm funcionan también en el nuevo terminal.

No he necesitado instalar npm pues ya lo tenía instalado de antes. A continuación, he ejecutado

```
npm install
```

y posteriormente he instalado la primera dependencia del proyecto, FontAwesome, mediante

```
npm install --save @fortawesome/fontawesome-free
```

Siguiendo con el proceso, he utilizado con éxito la dependencia utilizando el @import de SASS y tras ejecutar

```
npm run dev
```

todo funcionaba sin problemas. Finalmente, para realizar el deploy, he eliminado la carpeta y archivo de git existentes debido a clonar, y he inicializado un nuevo repositorio mediante los pasos descritos en este enlace. Tras pushear, he utilizado Netlify y lo he vinculado con GitHub para publicar mi web. Con esto listo, he empezado el desarrollo.

Sobre metodología y estilos

Sobre la metodología y estilos, me he basado en BEM. De todos los que vi, su división en bloques, elementos y modificadores me pareció muy coherente, y soy de los que prefiere que el

CSS tenga cierta semántica ya que facilita la lectura. No obstante, tras terminar la práctica debo decir que me he encontrado con ciertas barreras en BEM.

El principal problema que he encontrado es que su convención en bloques puede ser algo verbosa. Para los elementos, utilizamos el nombre del bloque, __ y el nombre del elemento. Para los modificadores utilizamos un --. Esto hace que podamos encontrarnos con clases como form__input--disabled que son algo largos. Para evitar alargar de más, no concatenamos elementos (en lugar de hacer bloque__elemento__subelemento, hacemos bloque__subelemento).

Esto desde un punto de vista de legibilidad me parece lógico. Ahora bien, mi problema es que pienso que hay situaciones en las que algo que debería ser un elemento (porque depende del bloque superior semánticamente), se convierte necesariamente en un bloque si tiene hijos que también dependen de él, pudiendo acabar a veces con bloques que tal vez deberían ser elementos.

Por ejemplo, si tenemos una lista de lenguajes, puede tener sentido que un lenguaje sea un elemento de esta si el lenguaje no tiene sentido fuera de esta lista. Pero si el elemento lenguaje tiene hijos dependientes de él, debemos tratar el lenguaje como un bloque (pese a no serlo realmente) o tratar a los hijos del lenguaje como elementos de la lista de lenguajes, lo que rompe un poco la jerarquía lógica y semántica.

Sin duda, no hay estándar perfecto y más práctica con el estándar ayudarán a resolver este tipo de situaciones con el mismo.

Para seguir la convención, he utilizado stylelint como se explica en la documentación de la asignatura, y le he añadido el plugin “stylelint-selector-bem-pattern” de este repositorio. También he añadido que ignore las reglas mencionadas en la documentación, así como @include y @mixin de SASS, que no me las reconocía. Mi archivo de configuración quedaría pues así:

```
module.exports = {
    "extends": "stylelint-config-standard",
    "plugins": [
      "stylelint-scss",
      "stylelint-selector-bem-pattern"
    ],
    "rules": {
      "plugin/selector-bem-pattern": {
        "componentName": "[A-Z]+",
        "componentSelectors": {
          "initial": "^\\.{componentName}(?:-[a-z]+)?$",
          "combined": "^\\.combined-{componentName}-[a-z]+$"
        },
        "utilitySelectors": "^\\.util-[a-z]+$"
      },
      "selector-nested-pattern": "^&",
      "indentation": 2,
      "no-descending-specificity": null,
      "no-eol-whitespace": null,
      "declaration-empty-line-before": null,
      "value-keyword-case": null,
      "at-rule-no-unknown": [
        true,
        {
          "ignoreAtRules": [
            "tailwind",
            "apply",
            "responsive",
            "variants",
            "screen",
            "use",
            "include",
            "mixin"
          ]
        }
      ]
    }
  }
```

Sobre el diseño

Para el diseño, he decidido hacer una página de CV que sirviese como portfolio (o pudiese servir con futuras ampliaciones). En lugar de imitar un currículum tradicional, he preferido optar por un enfoque algo más artístico en este aspecto. Pienso además, aunque no se aplique del todo puesto que se trata de la PEC1 y de que es una asignatura sin Javascript, que un portfolio puede aportar más que un CV tradicional ya que te permite demostrar más conocimientos (que tienes y que te faltan) que el primero.

Lo he estructurado en 3 secciones, una principal con foto, breve descripción y enlaces de contacto, una de mi stack y mis lenguajes, y otra con mi experiencia y certificaciones.

Como futuras ampliaciones, haría cambios para reflejar más mi personalidad, tal vez me gustaría que contase algo más sobre mi, y añadiría tal vez una sección de hobbies.

En este caso he optado por algo bastante estándar ciñéndome al enunciado, pero se me ocurren distintos estilos que me gustaría explotar, desde un diseño más futurista, a algo muy “cassolà” y “friendly”, con estilos y colores más cálidos, y algo que tengo pendiente (por ser preferencia personal mía, algo con estética más steampunk/cyberpunk).

He utilizado la siguiente paleta de colores:

Aunque para otras ocasiones creo que me gustaría utilizar colores de contraste más vibrantes que creo que podrían funcionar bien con los azules oscuros.

Sobre la metodología o enfoque, he decidido utilizar un enfoque desktop-first. El motivo son en realidad dos. El primero, que al tratarse de un CV, he pensado que un recruiter o alguien que estuviese analizando este tipo de cosas, trabajaría seguramente más desde un ordenador que desde el móvil, por comodidad para él, por lo que priorizar su experiencia desde un ordenador tiene más sentido. El segundo es que, tratándose de un portfolio, creo que centrarse en el

ordenador antes que el móvil es lógico puesto que te permite hacer más “florituras”. Centrarse en móvil sería en caso de querer mostrar específicamente nuestras habilidades para los diseños responsive (lo que es igualmente muy válido).

Aún así, el diseño funciona en ambos.

Sobre las dependencias

He añadido una única dependencia externa de un paquete que me gusta mucho. Su nombre es Devicon, esta es su página. Se trata de iconos de muchísimos lenguajes y herramientas tecnológicas, tanto en .png como .svg y customizables. La he instalado mediante

```
npm install --save devicon
```

Y he hecho un import en el main.scss para añadirla.

Sobre las distintas partes

Para el header, he optado por uno vertical y fijo con las tres secciones en la zona derecha. Bastante minimalista y sin que moleste mucho. En versión móvil, se sitúa de forma horizontal en la zona superior mostrando los 3 enlaces. He aprovechado que se trata de una única página para utilizar este diseño. En caso de mostrar más enlaces o tener que expandirlo, utilizaría un menú desplegable en móvil, con un hamburger y probablemente que ocupase toda la página, con una leve transparencia y una animación de entrada con un transition: ease-in-out .7s o similar desde alguno de los lados.

Sobre el resto de secciones, he intentado no caer en abusar de Flexbox para todo y he utilizado Grid para los layouts (en este caso dos/una columna, muy sencillos) y Flexbox para los componentes de los mismos.

He utilizado una cuadrícula para el stack y para los lenguajes, me he tomado la libertad de añadir un par de líneas de Javascript para poder animar según la posición del scroll, de manera que las barras de progreso (hechas mediante pseudo-elementos) se rellenan cuando el usuario hace scroll sobre la sección.

Para la experiencia, he tomado inspiración de las típicas tarjetas de contacto, resumiendo cada uno de mis empleos en una tarjeta que se da la vuelta al hacer hover para mostrar la descripción del trabajo.

Para el móvil he reorganizado los layouts de forma que pasan de ser dos columnas a una para mejor visibilidad.

El fondo es un degradado sencillo, colocado con background-attachment: fixed para que de el efecto de profundidad, al desplazarse el contenido y no el fondo. En este caso no, porque quería

que fuese más minimalista y tampoco tenía imágenes buenas a mano, pero esto podríamos aprovecharlo para acentuarlo aún más con algún efecto de parallax (sin abusar, posiblemente solo sobre una de las secciones).

Como había varias listas, he utilizado un mixin de SASS para añadirles el mismo padding y eliminar los carets. Lo he declarado en un archivo externo de _utils.

Al no ser excesivamente largo el proyecto, he dividido el CSS en _home y _header, el header porque de convertirse en uno desplegable se habría extendido y sería más legible así, y la _home, podría haberla separado tal vez en cada sección, pero según las dimensiones del proyecto tampoco me parecía que ganásemos mucho.

Sobre la compilación para producción

Para compilar para producción he utilizado npm run build, habiéndome asegurado antes de que el CSS pasaba los checks de Stylelint mediante npm run stylelint.

Repositorio: https://github.com/CarlosRM/CV

URL pública: https://master--upbeat-lamport-9fd222.netlify.app/