Javier Parellada González

Herramientas HTML y CSS II

02-11-2022

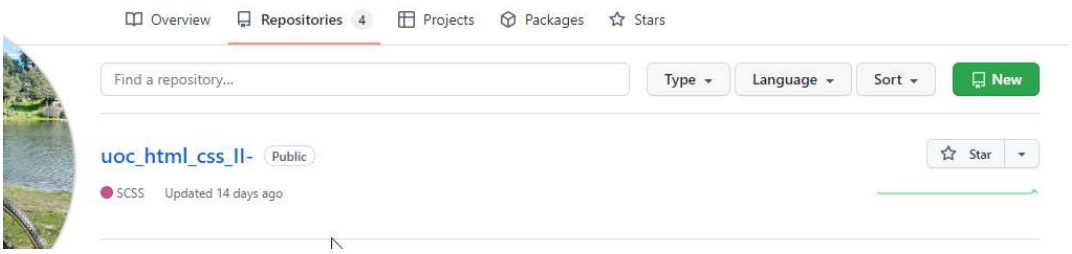
PEC 1

1- Preparación del entorno de trabajo

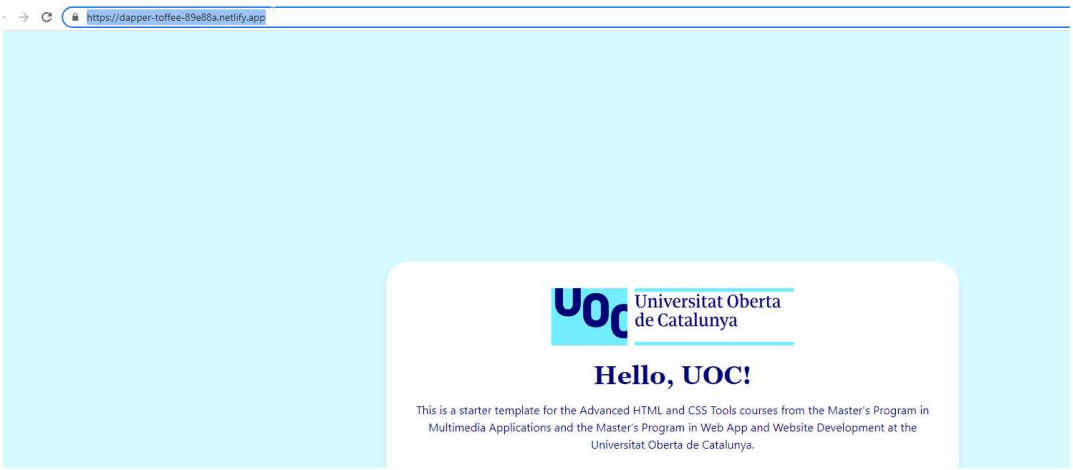
Creo una carpeta dónde voy a realizar la práctica. Abro la consola y me situó en la carpeta de trabajo e escribo git clone https://github.com/uoc-advanced-html-css/uoc-boilerplate.git.

Se descarga el repositorio a mi equipo en local. Ejecuto npm install.

Creo un repositorio en mi cuenta de GitHub para poder realizar las subidas de mi proyecto.



Publico la página en Netlify



Ya tengo el proyecto instalado y listo para trabajar con él.

2- Entorno de desarrollo

Para realizar esta práctica voy a utilizar el boilerplate de la UOC, que ya viene con una estructura de carpetas ya definida y con las herramientas que voy a necesitar.

Voy a usar como editor de código VsCode ya que es el que uso normalmente.

La estructura de trabajo es la siguiente.

1. **src**: Carpeta desde la que voy a hacer mi práctica
   1.1. **image**: Esta carpeta la he creado y es dónde voy a poner todas las imágenes que necesito en la práctica.
   1.2. **Styles:** Aquí tengo mi hoja de estilos *main.scss*. Es la hoja de estilos principal, en ella no pongo ningún estilo, sólo importo los parciales que si son los ficheros en dónde escribo código.
   He creado dos parciales: *_curriculum.scss* que es dónde escribo todos los estilos que necesito y *_variables.scss* en dónde están las variables, mixins y funciones que utilizo en *_curriculum.scss.*
   1.3. **Index.html:** Fichero html dónde voy a crear mi página.
2. **dist**: Carpeta en dónde está el proyecto compilado y es la carpeta que se pone en producción. Se crea cuando se ejecuta el script *build*.

3- Estilo

He leído principalmente dos guías de estilo.

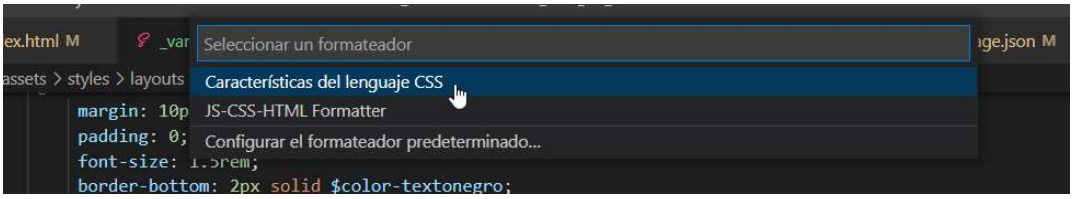
1- Guía de estilo de google: https://google.github.io/styleguide/htmlcssguide.html
2- Guía de estilo de @modo: https://codeguide.co

Ambas son muy parecidas.

Voy a seguir las siguientes recomendaciones del estilo según @modo.

1. Usar minúsculas para los códigos Hx.
2. Espacio después de la declaración.
3. No usar nombres para los colores.
4. Nueva línea después de cada punto y coma.
5. Siempre poner unidades cuando el valor sea cero.
6. Dejar el código bien indexado.
7. Ordenar las declaraciones. Para ello me descargaré *stylelint-config-recess-order.*

Para dejar indexado perfectamente el código utilizo un plugin de VsCode



El cual me ayuda bastante a tener el código ordenado.

He puesto las siguientes reglas en el StyleLint.

1. "selector-class-pattern": "^.[a-z]([a-z0-9-]+)?(__([a-z0-9]+-?)+)?(--([a-z0-9]+-?)+){0,2}$"
   Como voy a usar la metodología BEM he puesto esta regla para que me obligue a nombrar los elementos de un bloque según su metodología
2. "color-named": "never": No es aconsejable poner los colores por el nombre.
3. "color-no-invalid-hex": true: Los nombres lo voy a poner en hexadecimal, esta regla comprueba que exista ese código como color.
4. "declaration-block-semicolon-newline-after": "always" Es buena práctica que las declaraciones estén en distintas líneas. Como las declaraciones acaban en ; la regla te obliga a añadir una línea nueva después del ;.
5. "number-leading-zero": "never": No dejar poner un cero en los decimales entre 1 y -1.
6. Para cumplir el orden de declaraciones según la guía de estilo de @modo he instalado stylelint-config-recess-order con npm.
7. "color-hex-case": "lower": Para no dejar poner minúsculas enlos Hx.

```
{
    "extends": ["stylelint-config-recommended-scss", "stylelint-config-recess-order"],
    "rules": {
        "selector-class-pattern": "^.[a-z]([a-z0-9-]+)?(__([a-z0-9]+-?)+)?(--([a-z0-9]+-?)+){0,2}$",
        "color-named": "never",
        "color-no-invalid-hex": true,
        "declaration-block-semicolon-newline-after": "always",
        "number-leading-zero": "never",
        "color-hex-case": "lower"
    }
}
```

3- Metodología

La metodología que he decidido usar es la BEM. Para esta práctica tan pequeña he pensado que es la más adecuada y también es la que menos complicada de entender me ha resultado.

La idea de usar SMACSS para dividir las hojas de estilos en 5 categorías para esa práctica no lo veo muy útil. Bajo mi punto de vista esa metodología es para proyectos más grandes.

Usando OOCSS seguramente acabaría escribiendo la hoja de estilos sin ningún orden, no me ha gustado

Para mí BEM está entre de OOCSS Y SMACSS. Un término medio.

La idea de cómo se nombre las clases con BEM es lo que más me ha interesado.

Esta metodología me obliga a dividir mi página en elementos individuales llamados bloques.

Estos elementos los puedo reutilizar si fuese necesario. (Programación modular).

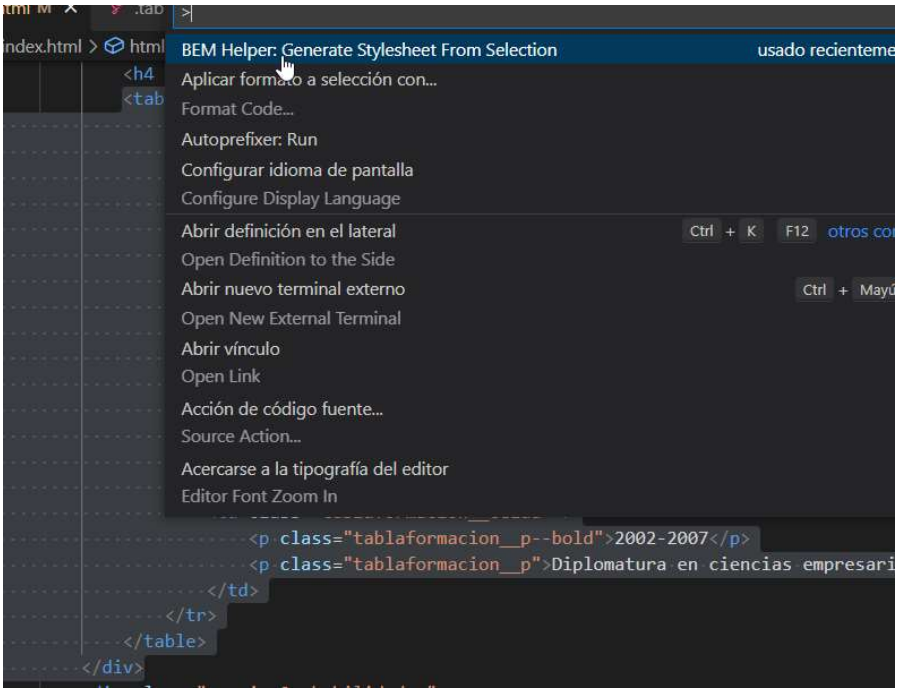
La forma de nombrar las clases es muy clara, aunque a veces sean muy largas.

Cada bloque tiene sus elementos y sus modificadores. Hay que usar siempre clases.

Para ayudarme a nombrar las clases como se describe en la metodología he instalado el plug-in

BEM HELPER.

Selecciono un bloque de mi HTML:



Y me da como resultado: (eligiendo sass). Anidado.

```
.tablaformacion {
  &__celda {
  }
  &__columna {
  }
  &__p {
    &--bold {
    }
  }
}
```

4- Desarrollo

Para el desarrollo de la página he empezado con los estilos para visualizar la página en un móvil (mobile first). La he dividido en dos secciones. La primera es dónde pongo la foto, referencias, hobbies y contacto y en la segunda es dónde pongo el perfil, experiencia, formación y habilidades. Lo he hecho así para que en el móvil se vean las secciones una debajo de la otra y en el escritorio una al lado de la otra usando un grid.

Para los estilos de la sección he usado una función pasando como parámetros el color de fondo y el color de texto ahorrándome escribir lo mismo dos veces y cambiando sólo los colores.

```
@mixin cajasseccion1($color, $colorfondo) {
    padding-top: 10px;
    padding-bottom: 25px;
    padding-left: 25px;
    color: $color;
    background-color: $colorfondo;
}
```

Para todas las listas también he usado una función porque son todas iguales y solo cambia el margen izquierdo que es el que paso como parámetro.

```
@mixin lista($margenleft) {
    display: flex;
    flex-direction: column;
    margin-left: $margenleft;
    font-size: 1.2rem;
    text-align: left;
    list-style: none;
    @media screen and (max-width: 480px) {
        padding: 10px;
        margin-left: 0PX;
        font-size: 1rem;
    }
}
```

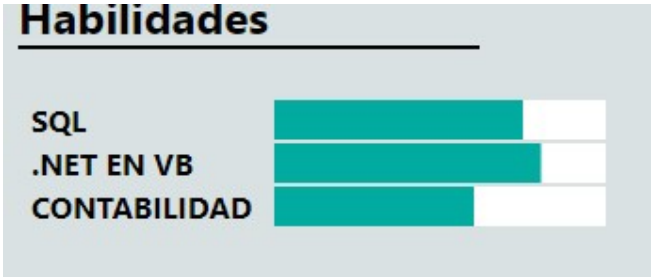
En habilidades he usado una función en dónde paso como parámetro el color del bakcground usando estas variables.

```
$backgroundsql: linear-gradient(90deg, $color-verde 0%, $color-verde 25%, $color-verde 75%, $color-textoblanco 75%);

 $backgroundconta: linear-gradient(90deg, $color-verde 0%, $color-verde 25%, $color-verde 60%, $color-textoblanco 60%);

 $backgroundnet: linear-gradient(90deg, $color-verde 0%, $color-verde 62%, $color-verde 80%, $color-textoblanco 81%);
```

Según que variable pase como parámetro obtendré las diferentes gráficas

```
@mixin habilidades($background) {
    margin-left: 10px;
    background: $background;
 }
```

Es para obtener este resultado



También he utilizado varios mixin con la única finalidad de no duplicar código.

Las variables, mixins y funciones, están en un único _parcial.

En _curriculum es dónde he escrito todos los estilos llamando a los mixin y funciones cuando las he necesitado.

Para escribir los estilos he seguido el mismo orden que en el HTML, tal como lo he hecho se ve claramente los módulos en la que he dividido la página.

Para los iconos de mi página he utilizado la librería de fontawesome.

Para ello he instalado el paquete desde la consola :

```
S C:\MASTER\HTML_CSS_II\uoc_html_css_II> npm install --save @fortawesome/fontawesome-free
```

Luego he tenido que darme de alta en su página para que me den un código para ponerlo en mi página y poder usar la librería.

```
<script src="https://kit.fontawesome.com/522df78ad8.js" crossorigin="anonymous"></script>
```

Resultado

5- GIT y Publicación

1. Git: https://github.com/Jparellada79/uoc_html_css_II-.git
2. Netlify: https://dapper-toffee-89e88a.netlify.app

.