James O'Beirne
M4.258 - Herramientas HTML y CSS II aula 1
PEC 1
22 de Octubre de 2021

1. Documentación del proceso de desarrollo (10%)
2. Justificación de las decisiones tomadas en el desarrollo (10%)
3. Justificación de la elección de la metodología y/o guía de estilo y memoria de la aplicación de los criterios elegidos (20%)

La URL pública de la web**:** **https://jamie-obeirne-resume-cv-10-2021.netlify.app/**

Enlace al repositorio de Git**:** **https://github.com/jamieobeirne/cv-resume.git**

1 Documentación del proceso de desarrollo

El proceso de desarrollo comenzó en realidad con los 2 proyectos no evaluados ("Flujo de trabajo" y "Guías de estilo") en los que estuve expuesto a algunas de las metodologías de este PEC. He explorado este proceso con las metodologías en la pregunta 3 ("Justificación de la elección de la metodología").

Después de haber elegido la mitología ITCSS(Inverted Triangle CSS) y de haber creado los 7 archivos CSS pertinentes, instalé Stylelint y empecé a buscar reglas que soportaran esta metodología. Soy nuevo en Stylelint y tengo/tenía alguna confusión con él. Por ejemplo, me parece que para utilizar correctamente StyleLint, todas las características de estilo (o linters internos) relacionadas con Visual Studio Code deben estar desactivadas. En ese caso, ¿necesitamos completamente Stylelint o la configuración de VSC puede proporcionar suficiente apoyo? Por ejemplo, dejé muchas de las reglas de código de VSC en su lugar durante este proyecto (una sangría de 5 espacios, por ejemplo - aunque en realidad prefiero una sangría de 2 espacios).

Me parece que StyleLint es enormemente flexible. Por ejemplo, con la ayuda de un repositorio externo, https://github.com/KamiKillertO/stylelint-itcss, encontré un plugin específico que crea reglas adaptadas a ITCSS para apoyar el enfoque de ITCSS (no creo que VSC pueda adaptarse así). El plugin/reglas ITCSS no permite el uso de "!important" y el uso de "variables", lo cual está en línea con la filosofía/perspectiva ITCSS. Sin embargo, ambas reglas están permitidas en sus respectivos parciales

("important" en el 7º parcial de "Trump / Utilidades" y variables en el 1º archivo de "Ajustes") - para acomodar esto, añadí un comando ignore al documento stylelintrc.json. Como probablemente recordarás de nuestros intercambios por correo electrónico, tuve algunos problemas para conseguir que Stylelint leyera este plugin externo y las reglas. Volví a leer la documentación del plugin y comenté este problema en el foro de la UOC y descubrí que el problema estaba relacionado con el "espacio de nombres", que es un concepto de programación con el que no creo haberme encontrado antes. Afortunadamente, pudisteis ayudarme con un detalle de configuración y el proceso de Stylelint funcionó (este proceso de solución ocurrió asíncrono durante varios días). El código final en mi stylelintrc.json final se ve así:

```
{
"extends": "stylelint-config-recommended-scss",
"plugins": ["stylelint-itcss"],
"rules": {
        "itcss/no-at-important":true,
        "itcss/no-variable-declaration": true,
        "unit-allowed-list": ["em", "rem", "%", "px"]
},
"ignoreFiles": ["src/assets/styles/_settings.scss","src/assets/styles/_trumps.scss" ]
}
```

Sospecho que posiblemente podría haber incluido más reglas (por ejemplo, relacionadas con los índices de 2 espacios), pero lo mantuve al mínimo debido a las limitaciones de tiempo y energía. Quizás, si no hubiera tenido algunas dificultades con el plugin, me habría aventurado a descubrir más reglas.

Aparte de la configuración de Stylelint, el proceso de desarrollo fue sencillo. Veo mi curriculum vitae / CV, que fue diseñado por un servicio en línea, libre, https://resume.io/app/resumes) y reflejó el diseño visual utilizando las reglas de SASS en los diversos parciales de ITCSS que luego se importan en el archivo principal de SASS. Cuando digo que utilicé reglas SASS, probablemente debería aclarar que la mayoría de ellas son reglas "vanilla CSS" con la excepción de las variables. Esencialmente, las variables son realmente útiles (especialmente cuando se agrupan en el enfoque de organización de ITCSS, en

Settings). Sin embargo, me siento menos cómodo con otras características de SASS como las funciones y los mixins, que no utilicé. En general, he reflejado mi curriculum vitae / CV con CSS / HTML y en gran medida disfrutado del proceso. Siento que mi capacidad para controlar un diseño visual con buena precisión está mejorando.

Otro descubrimiento interesante para mí en mi proceso como desarrollador fue el uso de objetos CSS, siguiendo en cierto modo el enfoque OOCSS. En realidad, utilicé mi propia versión del enfoque OOCSS, donde me di cuenta de que era útil tener varios contenedores espaciadores preestablecidos para separar las diferentes secciones del diseño. No sé si este es el enfoque correcto, pero he creado tres objetos espaciadores: ".no-margin-padding", ".reduce-bottom-margin-padding", y ".block-spacer". Estos tres objetos tienen varios atributos de espaciado y los incluí en las declaraciones de clase en el documento HMTL cuando quería cambiar rápidamente el espaciado. Como mencioné, no sé si este es el uso correcto (o un uso OOCSS en absoluto) pero me permitió hacer cambios rápidos sin volver a escribir los márgenes y el relleno comunes una y otra vez. Por ejemplo, el ".block-spacer" me permitió crear fácilmente espacio al final de las secciones principales del diseño:

```
.block-spacer {
        margin-bottom: 3em;
}
```

Incluyo estos objetos/reglas en el parcial de Objetos siguiendo el enfoque del ITCSS.

Otra herramienta útil fue darme cuenta de lo fácil que son los iconos de FontAwesome. Ya había utilizado iconos de FA anteriormente, pero lo hacía insertando el elemento SVG completo (tiene varias líneas con una larga lista de números). Durante este PEC, aprendí a importar FontAwesome como una dependencia y luego usar una sola línea para insertar un icono en el HTML. Utilicé un gran número de iconos. Esta fue la principal dependencia que importé (a menos que la importación del plugin para los archivos ITCSS de terceros también pueda contarse como importación de una dependencia).

Una vez que estas características estaban en su lugar, me puse a crear la capacidad de respuesta de la página. No sé si tengo un enfoque razonable, pero una recomendación de estilo muy útil es el concepto de mantener los saltos de medios cerca o directamente sobre su declaración inicial en los archivos CSS/SASS. He seguido esta recomendación y he encontrado que es un sistema mucho más fácil, al menos cuando vuelvo a hacer cambios al final de un proyecto. La regla inicial y su correspondiente ruptura de medios se establecen juntos. También establecí saltos de medios en el uso de variables en el parcial de Configuración de la siguiente manera:

```
/*------------------------------------*\
3.Media breaks
\*------------------------------------*/

$tabletMaxWidth: 769px;
$phoneMaxWidth: 481px;
$smallScreenMaxWidth: 1200px;
```

Una vez más, esto es muy útil cuando se modifican posteriormente los responsivenss - cualquier cambio se puede hacer en un solo lugar.

Una vez finalizado todo esto, la única dificultad que tuve fue con Netlify, que no lanzaba mi página debido a que StyleLint no estaba configurado correctamente. Una vez resuelto (¡el último día del PEC!), Netlify se ejecutó correctamente y tuve una copia de trabajo de mi CV / currículum en línea, lo cual es útil ya que estoy buscando mi primer trabajo de desarrollo. En ese momento, la página web estaba funcionando correctamente y tuve la sensación de desarrollador feliz - cuando una programación se ejecuta correctamente se siente como ser uno de los hermanos Wright con un nuevo avión.

2 Justificación de las decisiones tomadas en el desarrollo

Las principales decisiones que tuve que tomar fueron las siguientes::

1. **La elección de ITCSS como sistema de organización de CSS.** Esta fue una decisión natural y fácil que tomé al principio de este proyecto, en gran parte debido a la presentación personal de Harry Robert de las ventajas de ITCSS en una charla en línea (Harry Roberts - Managing CSS Projects with ITCSS en YouTube). Destacó las dificultades del CSS "vainilla" y defendió firmemente la separación de las reglas CSS en 7 archivos diferentes ordenados por especificidad. (En la pregunta 3 hablo un poco más de la disposición de ITCSS).

   Elegí este enfoque al principio del proyecto, y el propio proceso de desarrollo no ha hecho más que confirmar y profundizar mi interés. En otras palabras, el uso de este sistema ha sido tan exitoso como esperaba. Supongo que podría decir que me he "convertido" al ITSS. En realidad, sólo he utilizado 4 de los 7 archivos (Settings para las variables de SASS, Elements para los elementos HTML desnudos, Objects para mi versión de los objetos OOCSS y Components para la mayoría de las reglas CSS). Sin embargo, en esta breve toma de contacto, me sentí más orientado y cómodo haciendo la maquetación y los cambios.

2. **Usando un plugin de terceros para crear 2 reglas de StyleLint que soporten ITCSS.** Para ser honesto, todavía estoy aprendiendo StyleLint y aún me estoy sintiendo cómodo con el gran número de reglas y opciones. Como resultado, utilicé un número mínimo de reglas (3), dos de las cuales soportan directamente ITCSS (sin !import y sin variables, excepto en la ubicación designada). En esa línea, tuve algunas dificultades para hacer funcionar correctamente el StyleLint, problema que persistió hasta el último día (cuando

me ayudaste con el "namespace"). Como resultado de todo esto, no exploré otras reglas de StyleLint como quizás podría haberlo hecho. Además, todavía tengo preguntas sobre cómo StylLint interactúa con la configuración de Visual Studio Code.

3. **Importación de FontAwesome como dependencia y uso de una serie de iconos, que necesitaban una pequeña cantidad de posicionamiento.** He utilizado Font Awesome en mi página de portfolio (https://jamieobeirne.github.io/Developer/index.html) junto con el proyecto que hicimos en la clase de HTML/CSS Herramientas I de la UOC (https://spam-cafe-lazy-loading.netlify.app/) , pero esta vez ha sido mucho más fácil debido a la importación de la librería como dependencia. De este modo, la huella HTML es mucho menor. Los iconos de FontAwesome son impresionantes y muy divertidos de usar.
4. **Uso de Flexbox para manejar el apilamiento de los dos contenedores principales (el verde y el blanco) relativos a las versiones responsivas de la página web.**
   Flexbox es una de mis herramientas favoritas para crear capacidad de respuesta - simplemente pones contenedores más pequeños en un contenedor grande con una configuración flex para pantalla grande y, más tarde, añades una regla "flex-flow:row" para pantallas más pequeñas. He trabajado un poco con CSS Grid, pero, por el momento, me siento mucho más cómodo con Flexbox.
5. **Posicionamiento y personalización de la imagen.** Este era un problema menor que implicaba el uso de padding / margin y un par de comandos de centrado como "align-items: center" y "justify-content:center". El viewport alrededor de la imagen necesitaba ser configurado para crear una forma ovalada.

6. **Elegir el diseño básico de la página web, que modelé a partir de mi currículum actual que utiliza una plantilla de resume.io.** Elegí este diseño para mi currículum hace varios meses en resume.io (un servicio gratuito) y todavía me gusta. Aquí está una copia en línea: https://resume.io/r/EqzGpdHaC Por lo tanto, fue un paso natural para manual "espejo" este diseño preexistente. Hice algunos cambios menores como añadir el icono de FontAwesome y, por supuesto, añadir algunos de los requisitos de la PEC como la capacidad de respuesta.

3 Justificación de la elección de la metodología y/o guía de estilo y memoria de la aplicación de los criterios

Aunque esta es la última pregunta de la documentación, en realidad, aquí es donde empezó este proyecto para mí: aprender sobre las metodologías de las guías de estilo y elegir una.

En primer lugar, trabajé en las dos actividades no evaluadas.

En relación a la actividad "WorkFlow" no evaluada, he trabajado con comandos de terminal, comandos npm dev/dist, Parcel, Netlify y Github en otras clases pero descubrí algunos puntos nuevos durante este PEC. Por ejemplo, desplegar un host local con mi portátil y luego pasar esa dirección I.P. a mi teléfono para que se mostrara la misma página en el teléfono, incluso mientras se realizaban actualizaciones, fue muy interesante (¡muy cool!). Otra buena lección que aprendí fue sobre la importación de FontAwesome como una dependencia, lo que reduce mucho la complejidad - en el pasado, había insertado los iconos de FontAwesome como elementos HTML completos, de la misma manera que se ven los iconos SVG en bruto. Mi método anterior es más engorroso.

En relación con la actividad de "Guías de estilo" no evaluadas, todo esto era un territorio nuevo para mí. Básicamente, leí los artículos que recomendaban, desde el de Nicholas C. Zayas hasta el de Harry Robert, pasando por el de Mark Otto y las distintas metodologías (OCSS, BEM, SMASS e ITCSS), y tomé notas.

En general, todos estos artículos y enfoques eran interesantes y tenían ideas positivas. En relación con el aprendizaje real y la adquisición de ideas como estudiante, me gusta pensar en lo que realmente voy a utilizar y recordar dentro de seis meses. (En este sentido, mi única crítica sería que quizás hay demasiadas ideas para poner en práctica e interactuar con ellas de forma profunda). Me puse a sintetizar las distintas ideas en un núcleo de reglas útiles y memorables. Esto es lo que se me ocurrió:

1 Utilice sangrías de 2 espacios, no tabulaciones; "tabulación suave".

2 Cree un índice de contenidos.

3 Sólo debe haber unos 80 caracteres en cada línea.

4 Utilice el espacio de forma cuidadosa y coherente (generalmente, más espacio entre sesiones completamente nuevas; un espacio como mínimo).

5 Evite el anidamiento con SASS.

6 Comente todo lo que no sea obvio.

7 Mantenga los comentarios cortos.

8 Con los nombres CSS, muestra lo que hace el atributo en el nombre. Esto me recuerda el consejo de Robert Martins en *Clean Code* sobre que las variables sean "expresivas".

9 Poner guiones en los nombres (con un guión). No camel case o underscore.

Aparte de estas perspectivas generales, había varias metodologías que tenían procesos e ideas específicas. Desde el enfoque B.E.M. (Block Element Modifier), me gustaba mucho la idea de que dentro de un bloque determinado, los descendientes CSS deberían tener el nombre de los padres. En otras

palabras, todas las cadenas CSS dentro de "parent1" deberían comenzar con esa cadena: "parent1-class-name", o "parent1-image-class". Así, las relaciones son muy visibles sin un bloque de código determinado. La idea de utilizar el "elemento" en el nombre es útil, pero no entendí muy bien qué es un "modificador". Otras ideas útiles de esta metodología fueron las siguientes:

1 Mantener las consultas de medios lo más cerca posible del elemento. Siempre había agrupado las consultas de medios por posición de ruptura (en otras palabras, todas las reglas de tamaño de tableta van juntas, todas las reglas de tamaño de teléfono van juntas). Este antiguo enfoque es quizás útil a la hora de escribir el código, pero hace mucho más difícil volver a editar un componente individual (en relación con la capacidad de respuesta). Poner la regla de medios justo debajo de la regla inicial es un mejor enfoque y mantiene las cosas bien organizadas.

2 Limite el uso de declaraciones abreviadas, como los márgenes o el relleno. En otras palabras, utilice la forma completa de la siguiente manera:

~~margin: 10px 5px 15px 5px~~
margin-top: 10px;
margin-right: 5px;
margin-bottom: 15px;
margin-left: 5px;

3 En relación con el HTML, con múltiples valores de clase, agrupe los nombres de clase relacionados entre paréntesis []. (de lo contrario, mantenga 2 espacios entre ellos). No sabía que era posible utilizar paréntesis como este dentro de las asignaciones de nombres de clase en un documento HTML.

4 Utilice siempre comillas dobles en los atributos HTML.

Desde la OOCSS, me costó un poco entender este enfoque. En general, creo que el enfoque consiste en identificar las reglas de declaración de uso común y, a continuación, crear cajas u objetos de reglas de estilo vacíos que puedan aplicarse globalmente cuando sea necesario. En otras palabras, se crean reglas universales y vacías que se aplican al contenido cuando es necesario. Espero que se entienda de

forma razonable porque, aunque no utilicé este enfoque en gran medida, sí que creé algunos conjuntos de reglas vacías (especialmente relacionadas con el espaciado vertical) que luego añadí en varios bloques HTML. Estos "objetos" se establecieron en el parcial "_object" de mis archivos SASS, siguiendo el protocolo ITCSS.

En relación con el enfoque SMACSS, he leído el tema pero no me he sentido atraído por este enfoque en este momento.

Finalmente, llegué al enfoque ITCSS (Inverted Triangle CSS) y, especialmente, a una charla en línea de Harry Roberts (que mencioné y referencié en la pregunta 2). Él me iluminó en una perspectiva que tal vez había experimentado pero nunca había reconocido: el hecho de que cada pieza de CSS tiene la capacidad de afectar o ser afectada por todo lo demás, debido a la naturaleza en cascada de CSS. Describe el CSS como, esencialmente, "un árbol de dependencias" con el que es tan difícil trabajar. Señaló el hecho de que cuando se "inspecciona" una página web (una herramienta para desarrolladores) a menudo hay muchas, muchas reglas CSS que afectan a un componente, junto con reglas "tachadas" que están siendo anuladas por otras reglas CSS. Esto es muy complicado.

El enfoque de la ITCSS consiste en eliminar o reducir esta complejidad, básicamente escribiendo el código en un orden determinado basado en la especificidad, con lo menos específico en la parte superior y lo más específico en la parte inferior. En otras palabras, las reglas CSS deben escribirse de lo genérico a lo explícito, o de lo lejano a lo local, o de la baja especificidad a la alta especificidad. Esta idea fue revolucionaria para mí. Inmediatamente me sentí obligado a utilizar este útil enfoque y creé los 7 archivos (parciales de SASS) en el archivo boilerplate de la UOC:

1 **Settings** para las variables SASS y la configuración global (por ejemplo, las reglas "*")

2 **Tools** para funciones y mixins (no lo he utilizado)

3 **Generic**. Esta capa utiliza una serie de comandos con los que no estoy muy familiarizado: restablecer y/o normalizar los estilos, definir el tamaño de las cajas, etc.

4 **Elements** para el HTML en bruto, como H1-H6

5 **Objects** siguiendo el sistema OOCSS

6 **Components** para la mayoría de las reglas CSS

7 **Trumps / Utilities** utilizar una especificidad muy alta, especialmente !importante

Siguiendo este enfoque se "domina el CSS", según Harry Roberts. Todo esto me pareció muy concreto, útil y aplicable.

Aunque este sistema de 7 archivos/parciales es un poco "exagerado" para nuestro pequeño proyecto PEC, lo encontré inmediatamente útil. Establecí todas las variables de SASS en el archivo de configuración y lo encontré mucho más rápido para hacer ajustes. Hice algunos pequeños "objetos" que establecí en el parcial de Objetos y luego los apliqué a la página web cuando fueron útiles. Declaré todos mis elementos HTML en bruto en el archivo Elements. Finalmente, la mayor parte de mi código es el parcial Componentes. Debido al hecho de que el archivo Elementos, Objetos y Componentes todos utilizaron variables SASS, y que las variables fueron agrupadas en el archivo de Configuración, fue mucho más fácil hacer pequeños y grandes ajustes.

Otros enfoques de estilo que solía utilizar eran mantener las pausas de medios con cada elemento, utilizar guiones en lugar de guiones bajos y guiones y, por último, crear pequeños objetos que pudiera adjuntar cuando fuera necesario para manipular un elemento sin escribir ningún código nuevo.

Confío en que estas guías de estilo y el enfoque de la ITSS se quedarán conmigo y formarán parte de la perspectiva de desarrollo permanente que me informa. Por esa razón, junto con otras, este PEC me resultó