Alumno:	Teo Alarcón Moya
Asignatura:	Herramientas HTML y CSS
Actividad:	P1. Desarrollo de una web
Repositorio:	https://github.com/Moyat89/Separacion
URL:	https://separacion.netlify.app/

1. Boilerplate

El *boilerplate* utilizado está basado en Parcel tal y como se proponía en el ejercicio. Para crear este *boilerplate*, como ya tenía instalado npm y node.js en mi ordenador (de las actividades previas a esta PAC), creé un repositorio local para el proyecto (PAC1), accedí a este a través de la consola virtual de Windows (cd…) para generar el *package.json* (npm init --yes) que iba a servir como base del proyecto.

Una vez generado, edité el *package.json* para añadir algunos detalles tales como el nombre del proyecto (separación_pac1), la versión (1.0.0), fuente donde se iba a ubicar el index.html en mi repositorio local (src/index.html) y autor entre otras cosas. También borré el campo autogenerado main, ya que podía interferir con *Parcel*.

Además, añadí soporte a navegadores antiguos propiedad con la línea <<browserslist": "> 0.5%, last 20 years>>, cuyos valores me parecían adecuados para capturar los requisitos del enunciado (navegadores que se utilizan > 0.5% mundialmente, todas las versiones lanzadas en los últimos 20 años sin especificar si están muertos o no).

Seguidamente instalé el mo*dule bundler Parcel*, con la consola virtual en la raíz de mi proyecto ejecutando npm install --save-dev parcel. Así mi proyecto ya estaba listo para ser compilado por *Parcel* y lo que me quedaba era trabajar en el *html, css y javascript* que lo iban a componer.

Como la tarea que nos ocupa es crear un sitio web de una serie, dentro de la carpeta del proyecto creé una carpeta src con un archivo html que serviría como punto de partida (index.html), una carpeta js con un archivo script.js para crear algún tipo de funcionalidad con *javascript* más adelante y una carpeta *css* con el archivo styles.scss (no css, ya que quería utilizar sass como preprocesador css y .scss es uno de los formatos que soporta).

Tras popular un básico *Hola Mundo* con un estilo básico definido en el archivo styles, en la carpeta raíz ejecuté npx parcel src/index.html para que *Parcel* generase el servidor de desarrollo con los archivos que había creado. A partir de ahí desarrollé la web con normalidad en los ficheros html, css y js con la diferencia de que los cambios los podía ver en cambio real en el servidor de desarrollo.

2.Dependencias

Nombre	Tipo de Procesador	Comando de instalación
postcss-responsive-type	Postcss	npm i postcss-responsive-type
posthtml-include	Posthtml	npm i -D posthtml-include
sass	Precss	npm install -g sass

PostCSS Responsive-type

Se ha utilizado con el fin de ahorrar líneas de código a la hora de hacer la web con un diseño responsive. Añade la propiedad responsive a los atributos css font-size, line-height y letter-spacing. Al usar esta propiedad los tamaños varían dentro de un rango definido dependiendo de la resolución en la que la web se visualice.

Para utilizarlo, en el archivo o archivos .css (o .scss en mi caso) donde definimos las reglas css que se van a aplicar en nuestra página, vamos al atributo (de los listados en el párrafo anterior) que queremos que sea responsive y le damos dos tamaños con la estructura siguiente: responsive(tamaño 1, tamaño 2). Estos dos tamaños serán el mínimo y el máximo con el que las propiedades fluirán dependiendo de las resoluciones que marquemos como mínimas y máximas para que fluyan.

Para configurar estas resoluciones que utilizará como base para que el tamaño fluya, se crea un archivo .postcssrc en la raíz de proyecto. Dentro de este archivo, en plugins se añade este (postcss-responsive) junto a sus resoluciones de la siguiente forma:

```
{
    "plugins": {
     "postcss-responsive": {
       "minWidth": 480,
      "maxWidth": 1280
     }
    }
  }
```

Si no se definen en este archivo las resoluciones, también se pueden definir a la hora de dar valor a las propiedades de la siguiente forma: propiedad: responsive (tamaño 1, tamaño 2, resolución mínima, resolución máxima).

Posthtml-include

Una barra de navegación es indispensable para que los usuarios puedan moverse a través de los contenidos de nuestra página web. El proceso de implementarla en nuestros proyectos a veces es tedioso, ya que en cada archivo html hay que escribir y replicar el código que la compone.

Para ahorrarme este proceso, he optado por utilizar posthtml-include y tener la barra de navegación en un fichero aparte (navgen.html) para poder referenciarla después en cada archivo html que la utilizaba. Para ello he utilizado las líneas de código propias de posthtml-include <include src="navgen.html"></include> donde debería estar y así he ahorrado el tiempo y el código que supondría el tener que replicar el menú de navegación en todas las páginas.

Sass

También he decidido utilizar sass para experimentar con un procesador precss y así hacer más cómodo el desarrollo del código css. Sass tiene muchísimas utilidades que ayudan a escribir un código css más fluido y de manera más intuitiva (ver https://sass-lang.com/guide/) pero las funcionalidades que yo he utilizado para adentrarme en su práctica son dos: la creación de variables y la de *anidamiento/nesting*. De la primera función, he creado una variable $color-theme con el valor rgb(130, 187, 135) para almacenar el color primario de la web en una variable y no tener que definirlo en todos los elementos que lo van a utilizar.

La segunda función, el nesting, la he utilizado en general para poder anidar los elementos css y sus propiedades como si se tratasen de un lenguaje de programación, sin tener que especificar individualmente los elementos de una manera granular. Por ejemplo:

```
.nav-scroll-gen {

    ul {
        background-color: rgba(2, 5, 7);
        list-style: none;
        background-position: right;
        display: grid;
        grid-template-columns: 1fr 1fr 1fr 1fr;
    }

    li {
        border: 1px black solid;
        padding: 0px;
    }

    a {
        text-decoration: none;
        color: $color-theme;
        text-transform: uppercase;
        margin: 0px;
        width: 100%;
        height: 100%;
```

```
        line-height: 5vh;
        display: block;
        text-align: center;
    }

    a:hover {
        background-color: white;
        color: black;

    }

}
```

Como se puede observar, no ha hecho falta definir uno por uno los elementos de .nav-scroll-gen a los que quiero cambiar un valor, si no que los he anidado todos en .nav-scroll-gen y los he modificado.

Para utilizar sass de una manera óptima que se adecuase al entorno de desarrollo de *Parcel,* he ejecutado el comando sass --watch input.scss output.css (siendo input.scss mi archivo styles y output.css el que he utilizado como referencia del estilo en el html) para poder ir modificando el scss y que se reflejarán en el output directamente (es decir, que sass los transformase en tiempo real).

3.Git

Para albergar el proyecto en un repositorio remoto y así poder vincularlo con el entorno de producción una vez que estuviese finalizado, he creado un repositorio vacío en GitHub con el nombre de Separacion (título de la web del proyecto). Una vez hecho esto, a través de la consola virtual de Windows, en la carpeta raíz de mi proyecto, he ejecutado git init para empezar a trackear el repositorio local.

Como Parcel crear muchos archivos que no son relevantes para el entorno de producción, seguidamente he creado un archivo .gitignore (que permite que git no trackee los archivos definidos en él) con las siguientes líneas:

/node_modules

.parcel-cache

.cache

/dist

Así me he asegurado de ignorar estos archivos (a veces pesado) a la hora de subir mis cambios al repositorio remoto.

A partir de aquí, he añadido todos los archivos restantes mediante el comando “git add .” , he hecho un commit (git commit -m”mensaje”) para trackear los cambios, he añadido como origen remoto al git local el repositorio remoto en GitHub que he mencionado al principio (git remote add origin https://github.com/Moyat89/Separacion.git) y he hecho un push request a la rama main de todo lo que habia añadido (“git push -u origin main”). Así todo el trabajo que había hecho en el repositorio local se había registrado en el remoto.

Cada vez que he añadido y modificado de una manera sustancial, lo he ido subiendo a GitHub para trackear los cambios y asegurarme de no perderlos.

4.Adecuación a la temática y estructura de la práctica

Para esta práctica he decidido construir una fan web de la serie Severance (de Apple TV+), ya que cuando la vi a principios de este año me impactó mucho su propuesta y me pareció adecuada para dedicarle este proyecto.

La web se compone de los siguientes elementos / archivos:

index.html: Sirve como página de entrada de la web y tiene como nombre Portada. El html tiene vinculado un script declarado en un archivo js aparte que se ejecuta al cargar la página y “sustituye” (ver checkIn.js) el contenido de la página por un botón. Una vez el usuario pulsa el botón, se ejecuta un script (ver butonEnter.js) que muestra el contenido de la portada: un menú de navegación con los diferentes html que componen el sitio y un cuadro de texto dando la bienvenida.

categoria.html: Esta página sirve como enlace a otras páginas de contenido sobre la serie. Se compone del mismo menú de navegación superior que sirve para navegar por el resto de páginas principales de la web, y también dispone de un menú izquierdo (superior en móviles) que nos permite movernos por los diferentes contenidos (detalles, making of, capítulos, curiosidades).

detalles.html: Página de detalles con la misma estructura que categoría. El contenido del cuadro de texto se compone de una ficha técnica de la serie, una sinopsis y un tráiler.

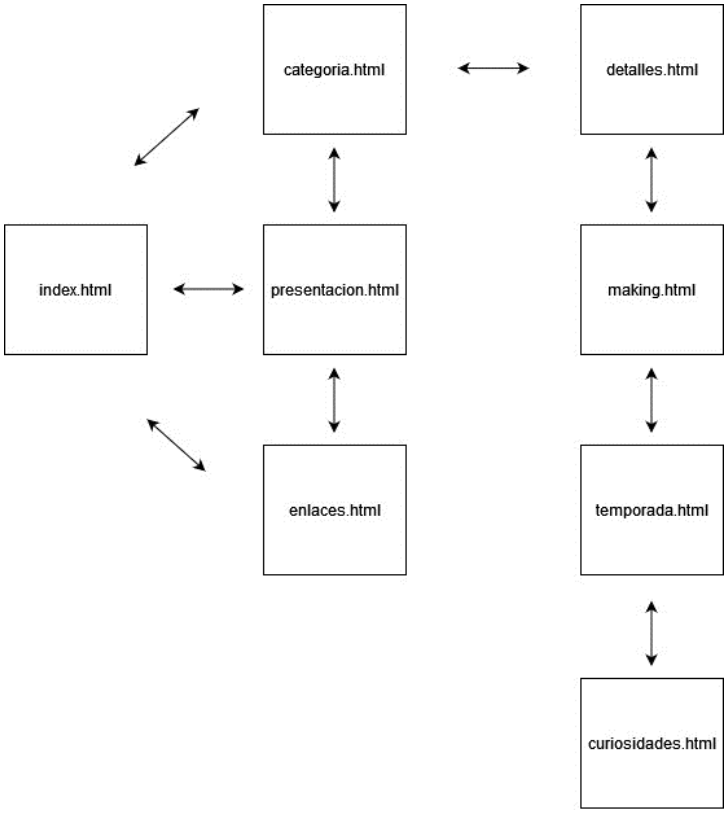
making.html: Segunda página de detalles con la misma estructura que categoría. Entrevistas a parte del equipo de la serie con anécdotas sobre la grabación. La página se compone de un artículo largo con estas entrevistas y una imagen que sirve de entrada para esta.

temporada.html: Tercera página de detalles con la misma estructura que categoría. El contenido es una tabla con la información básica de cada episodio y una sinopsis. Una imagen sirve de entrada para esta.

curiosidades.html: Cuarta y última página de detalles con la misma estructura que categoría. En esta, se detallan varias curiosidades sobre la serie que la gente no suele conocer. Tiene imágenes para ilustrarlas.

presentación.html: Página que presenta y explica la estructura y contenido de la web. Dispone de un menú de navegación superior para poder moverse entre las otras páginas del sitio.

enlaces.html: Página que recoge todos los recursos externos utilizados, acreditando a sus autores y mostrando enlaces directos a esos contenidos. Dispone de un menú de navegación superior para poder moverse entre las otras páginas del sitio.



styles.scss: Archivo que sirve como el principal con las reglas css para aplicar en los html.

checkIn.js: Archivo javascript que contiene la definición de una función y su ejecución: checkin(). La primera función se ejecuta automáticamente en el index.html y sirve para que el usuario, al ingresar en el sitio web, se encuentre solamente con un botón, ya que oculta el resto de elementos de la página.

ButonEnter.js: La otra función, entrar(), está vinculada al botón y se ejecuta al pulsar el botón. Actúa a la inversa que checkin(): esconde el botón y hace visible el contenido de la web.

5.Diseño

La web ha sido diseñada y desarrollada utilizando la filosofía *mobile first*. Es decir, el diseño ha sido concebido inicialmente para su visualización en móviles y tras este se han añadido una serie de *media queries* para adaptar algunos de sus elementos y tamaños a la visualización en dispositivos medianos y dispositivos grandes.

Las diferencias más grandes entre las diferentes visualizaciones residen en el tamaño de la pantalla que ocupa el contenido (en dispositivos pequeños ocupan toda la pantalla), el menú de navegación de categorías (estando en un lado en dispositivos grandes y estando bajo el menú de navegación general en los otros) y el tamaño de los paddings y fuentes de los párrafos e imágenes (se ha adaptado utilizando postcss responsive-type en casi todas las ocasiones y en los *media query* se ha definido lo mínimo).

Como fondo de la web se ha utilizado una imagen promocional de la serie, la cual está pensada que ocupe toda la pantalla. Para los colores, se han utilizado variaciones de los colores que se pueden ver en ese fondo promocional. Además, se ha utilizado la fuente de *Google Share Tech* Mono para dotar de un aire “retro” a la web como la serie.

En la página de Categoría, el menú de navegación que sirve para movernos a través de las páginas de detalles utiliza las imágenes de los protagonistas de la serie como fondo para hacerlo más atractivo visualmente.

La composición de los selectores html de todas las páginas es bastante sencillo y parecido. Dentro del body encontramos encontramos un div (.content-cover o .content-gen dependiendo de si es la portada o el contenido) que nos sirve para definir en css los espaciados y propiedades generales de los selectores anidados en este (todos). Encontramos el menú de navegación (include navgen.html) compuesto por una lista con display grid y cuatro columnas (portada, categoría, presentación, enlaces). Al menú le sigue un header con dos selectores (h1,h2) que contienen el logo y el lema de la web.

Seguidamente, encontramos la clase main que se compone normalmente de un article con los diferentes selectores html para contener el texto del contenido de la web. Dentro de este article, también podemos encontrar alguna figure con imagen o algún iframe con un tráiler dependiendo de la página.

En cuanto a accesibilidad y semántica, todos los selectores pertinentes han utilizado propiedades para hacer accesible el sitio web. Por ejemplo, las referencias tienen el idioma de origen explicito (si es diferente del español), las abreviaciones también disponen de propiedad title con el valor de que quiere decir la abreviatura. Lo mismo ocurre con las imágenes mediante el atributo alt y con demases selectores html que permiten hacer su contenido más accesible.

6.Publicación a Internet

Se ha utilizado *Netlify* para crear el entorno de producción y hacer accesible la web a través de internet. Después de crear la cuenta, simplemente he vinculado mi repositorio Github con el servicio y he establecido como directorio del paquete dist/ (el "output" del build) y el comando npm run build para que se construya. Al actualizar *GitHub* con nuevos commits, *Netlify* volvía a construir el sitio con Parcel mediante npm run build gracias a las dependencias e instrucciones definidas en el *package.json.*

7.Recursos Externos

Artículos originales

- Ficha técnica,sinopsis y detalles sobre episodios: *Severance (serie de televisión). (2023, 9 de octubre). Wikipedia, La enciclopedia libre. Fecha de consulta: 14:13*.
- Artículo de la sección Making of: *Roxana Hadadi, Jackson McHenry, Kathryn VanArendonk (15 de Abril de 2022). The Diamond Desk, Surveillance Shots, and 7 Other Stories About Making Severance. Vulture.*
- Artículo de la sección Curiosidades: *Severance: Anécdotas (2022). Sensacine Mexico.*

Materiales audiovisuales

- Imagen fondo principal de la página: *Recursos para la prensa de Severance. (2022). Apple TV+*.
- Imágenes utilizadas como "iconos" para las diferentes páginas de detalles:
    - Imagen de Mark: *Looper. (2022). Copyright Apple TV+.*
    - Imagen de Helly: *Severance Wiki. (2022). Copyright Apple TV+.*
    - Imagen de Dylan: *Reel Mockery. (2022). Copyright Apple TV+.*
    - Imagen de irving: *Severance Wiki. (2022). Copyright Apple TV+.*
- Imagen página Making of: *Cinemanía. (2022). Copyright Apple TV+.*
- Imagen página Capitulos: *Taylor Holmes. (2022). Copyright Apple TV+.*
- Imágenes utilizadas en Curiosidades:
    - Imagen de Ben Stiller: *Mike Marsland / WireImagem para Netflix.*
    - Imagen de Patricia Arquette: *El Mundo. (2020).*
    - Imagen del Edificio de Bell Labs *MBisanz talkBell_Labs_Holmdel,_The_Oval2.png: *derivative work: MBisanz talkBell_Labs_Holmdel,_The_Oval.jpg: Lee Beaumont - Bell_Labs_Holmdel,_The_Oval2.png, CC BY-SA 2.0.*
- Imagen de Presentación: *Revista Velvet (2020).*
- Icono de la web: *Computadora iconos creados por Freepik - Flaticon*

Severance, creada por Dan Erickson, es una propiedad intelectual de Apple.