PRÁCTICA EVALUABLE DEL CURSO 1

Herramientas HTML y CSS

15 DE NOVIEMBRE DE 2024

ALEJANDO ABUÍN ÁLVAREZ

El presente documento tiene como objetivo llevar a cabo la explicación del proyecto web y se desarrollará según los siete puntos recomendados en la web de la PEC:

1. Creación del boilerplate basado en Parcel.
2. Gestión de dependencias: pre- o postprocesadores y dependencias adicionales.
3. Creación del repositorio Git
4. Adecuación a la temática y estructura de la práctica.
5. Diseño responsive, complejidad y estética
6. Semántica y accesibilidad
7. Publicación a internet

Antes de comenzar, dejaré constancia de los enlaces del repositorio y del enlace de la web:

- Repositorio: https://github.com/comolays/HerramientasUOC

- Web: https://loquacious-douhua-608636.netlify.app/

1. Creación del boilerplate basado en Parcel.

El **boilerplate** se ha definido con una estructura lo más organizada y simple posible, para facilitar la comprensión del mismo por alguien ajeno al proyecto. Todos los archivos referentes a la estructura (**HTML/IMG**) y estilo (**CCS**) se encuentran dentro de la carpeta "src". Fuera de esta, ya se encuentran el resto de los archivos necesarios para realizar el "workflow".

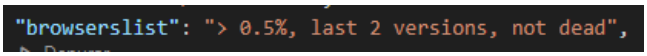
A continuación, se explicarán los dos entornos usados en este proyecto:

- De primera mano, el entorno de producción, bastante útil para ver la "experiencia final" de usuario y que todo cargase correctamente.

- En segundo lugar, el entorno de desarrollo se ha estado utilizando para ver al momento los cambios realizados. Esto ha proporcionado una gran ventaja para lograr los objetivos de una manera rápida, eficaz y corregir los fallos de manera óptima.

  Un reto encontrado fue la carga lenta de imágenes en este entorno, por lo cual se bajó temporalmente la resolución de las imágenes para facilitar la edición en vivo.

En cuanto a la compatibilidad con los navegadores, se ha definido la siguiente configuración de "browerlist":



Con este código, excluiremos los navegadores que ya no reciben actualizaciones o soportes de seguridad, por temas de seguridad del usuario y nuestro. A mayores, hacemos que el código sea compatible con las últimas dos versiones de cada navegador importante, lo que mejora la compatibilidad sin comprometer el rendimiento.

Por último, el código debe de ser compatible con todos los navegadores que tengan al menos el 0.5% de cuota de mercado. Esto hará que nos limitemos a navegadores populares y evitar dar soporte a navegadores poco usados.

Resumiendo, optimizaremos la aplicación para que sea posible usarla para la gran mayoría de usuarios. Al final, nuestro objetivo es ofrecer una experiencia de calidad a un amplio volumen de personas, priorizando tanto la funcionalidad como la seguridad.

En cuanto al apartado de scripts:

```
"scripts": {
  "start": "npm-run-all clean parcel:dev",
  "build": "npm-run-all clean parcel:build",
  "parcel:dev": "parcel",
  "parcel:build": "parcel build",
  "clean": "rimraf dist .parcel-cache"
```

Al ser un proyecto basado en parcel, utilizaremos los dos primeros scripts para ejecutar Parcel en modo de desarrollo (**start**) y producción (**build**), realizando en primer lugar una limpieza de archivos (script: **clean**). Esto último se hace para garantizar un entorno limpio, seguro, optimizado y evitar cargar archivos obsoletos/antiguos que puedan causar problemas.

A continuación, nos encontramos con palabras clave que ayudarán al posicionamiento de la web en búsquedas de usuarios.

Por último, Parcel tendrá como punto de entrada el archivo index.html, al tratarse de la web principal de nuestro sitio. A partir de este, comenzará a procesar y empaquetar todos los recursos necesarios para la web:

```
"source": "src/index.html",
```

2. Gestión de dependencias: pre- o postprocesadores y dependencias adicionales.

- **Dependencias principales utilizadas.**

En este proyecto he considerado usar las siguientes librerías CSS:

```
"dependencies": {
  "animate.css": "^4.1.1",
  "hover.css": "^2.3.2"
}
```

En primer lugar, quería un tipo de dependencias sencillas, al no haber trabajado nunca con ellas, pero que le pudiesen dar vida a la página y mejorar la experiencia de usuario.

- **Animate.css**: como indica su nombre, otorga animación a los elementos que se aplique. Por ejemplo, la he usado en bloques y en el menú de navegación para que de la impresión de que cuando el usuario entre en la página, esta parezca que se está construyendo delante de él.

- **Hover.css**: la he usado para darle dinamismo a los botones y efectos de zoom a alguna imagen y tarjeta, para que cuando el usuario haga **hover/focus** sobre estos elementos, permita una mejor visualización de las imágenes/tarjetas y enfatice los botones.

Ambas fueron instaladas mediante la terminal y, posteriormente en ella, utilizando el comando de **npm install**. Una vez instaladas ambas librerías, figurando ya en el archivo .json, las vincule con todos los archivos .html, para que estas pudiesen funcionar correctamente. Cada elemento .html del proyecto cuenta con las siguientes líneas:

```
<link rel="stylesheet" href="../node_modules/animate.css/animate.css">
<link rel="stylesheet" href="../node_modules/hover.css/css/hover.css">
```

Para ejecutar las animaciones que en ellas se encuentran, simplemente hay que llamar el efecto que se quiere usar. Este efecto es tipo clase y simplemente se ha de adjuntar al

elemento del html que se quiera aplicar. A continuación, ponemos de ejemplo un efecto de anímate.css:

```
<div class="gridmenu animate__animated animate__bounce">
```

Ambos paquetes tienen webs propias que permiten ver el efecto que aplicarán a tu código, por lo que esto facilita bastante el trabajo.

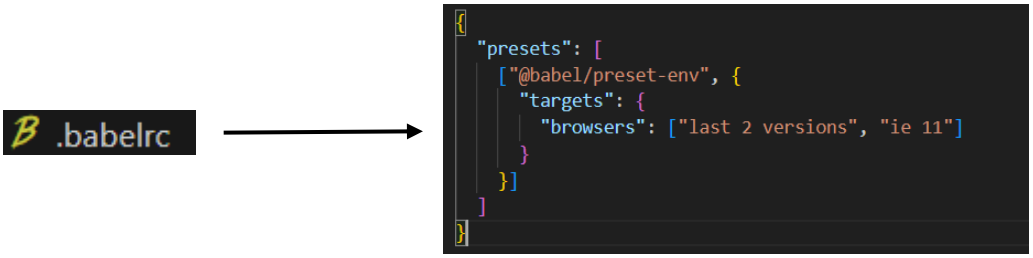
- **Postprocesador**:

En nuestro proyecto se ha solicitado incluir código de JavaScript, por ello, he decidido optar por el post-procesador Babel, el cual ya viene incluido con Parcel.

El principal motivo es que gracias a este post-procesador, nuestro código de JS moderno podrá ser compatible con navegadores antiguos, en caso de que alguien ejecute nuestro sitio web en alguno de ellos. Es decir, damos una mejor experiencia de usuario permitiendo que una mayor variedad de dispositivos y navegadores funcionen correctamente.

Fue necesario instalar su dependencia "preset-env", a través de **npm**, Este preset es una configuración de Babel que convierte automáticamente las características modernas de JavaScript según la compatibilidad del navegador, lo que facilita su correcta ejecución en entornos más antiguos.

Posteriormente, para su correcta configuración, tuve que crear un archivo ".babelrc" para la configuración del post-procesador, en el cual establecimos el preset, lo cual asegura que el código JavaScript se transpile según los navegadores objetivo que definimos:



Ha sido necesario incluir la línea “browsers” para forzar en este proyecto a cambiar el código de JS moderno a antiguo y ver el resultado del post-procesador. A continuación, adjuntamos una imagen del código usado, seguidas del resultado de la transpilación realizada por Babel:

```
//FUNCIÓN SIMPLE DE CONVERSIÓN DE MONEDA
function Euros_A_Dolares(euros){

  const dolar = 1.06;
  return 1.06*euros;
}

let euros = parseFloat(prompt("Introduce la cantidad de euros que quieras convertir:"));

console.log(euros + "€ Euros son en dólares: " + Euros_A_Dolares(euros).toFixed(2) + "$");
```

```
"use strict";

//FUNCIÓN SIMPLE DE CONVERSIÓN DE MONEDA
function Euros_A_Dolares(euros) {
  var dolar = 1.06;
  return 1.06 * euros;
}
var euros = parseFloat(prompt("Introduce la cantidad de euros que quieras convertir:"));
console.log(euros + "€ Euros son en dólares: " + Euros_A_Dolares(euros).toFixed(2) + "$");
```

Como vemos, las variables “const” y “let” han sido cambiadas por “var”.

3. Creación de repositorio GIT.

En primer lugar, creé la distribución del proyecto en el escritorio dentro de una carpeta, organizando todos los archivos de manera que consideré adecuada. Una vez me aseguré de que era la manera correcta de tener el proyecto organizado, elaboré el repositorio en GIT.

La creación del repositorio fue a través de la web de Github, al cual le di visualización pública. También una breve descripción (haciendo referencia a esta PEC), palabras clave relacionadas con la temática de la web y el enlace correspondiente para que los usuarios interesados tengan acceso directamente a ella.

Una vez creado el repositorio en GitHub, mediante la terminal realicé la vinculación de la carpeta local del escritorio con el repositorio. A partir de este momento, realizaba los comandos **git add**, **git commit** y **git push**.

Al ser la primera vez que utilizaba esta herramienta (Git y Github), probé varias formas de subir los archivos como manualmente y utilizando la aplicación de escritorio. Finalmente, la vinculación desde la terminal me pareció la opción más eficaz y sencilla para gestionar el proyecto.

Como nota informativa, he tenido también que usar el comando **git clone** ya que durante la realización de la práctica he tenido que cambiar de ordenador, por lo que esto me permitió recuperar de la nube de Github, el proyecto por donde lo había dejado.

4. Adecuación a la temática y estructura de la práctica.

La temática de la web fue seleccionada siguiendo las recomendaciones indicadas en la descripción del trabajo. Sin embargo, me motivó el hecho de que pertenezco a esta zona geográfica, por lo que podría mostrar de primera mano tanto sus sitios más representativos como gastronomía.

Durante todo el diseño de la web, quería que estuviesen reflejados el color azul y blanco, representativos de la bandera de la Comunidad Autónoma de Galicia. Además, son colores que combinan de forma armónica y favorecen la accesibilidad, facilitando una lectura clara para el usuario.

He intentado que la accesibilidad sea detallada y concisa, evitando párrafos largos, evitando desalentar la lectura del usuario. Me gustaría recordar, que todas las web tienen animación, para crear dinamismo en la web y crear una mejor experiencia de usuario.

Referente a la página de “Portada”, creí oportuno realizar 3 secciones para cada navegación, con una breve información, que se pedía que tuviera esta web.

En cuanto a la página de “Categoría”, opté por un sencillo diseño que permite una visualización total de todos los elementos de cada categoría, entre los cuales el usuario puede escoger de manera rápida cual es de su interés. También, cada tarjeta cuenta con una previsualización del sitio/comida, facilitándole una idea previa de lo que se va a encontrar en el sitio web específico del elemento seleccionado. Esta página tiene vinculado el archivo de JavaScript.

Me gustaría destacar el diseño por bloques en todas las páginas. Esta estructura permite organizar el contenido en secciones separadas, evitando que la información se vea sobrecargada en un solo bloque. Facilita la retención de la información y ayuda al usuario a crear un mapa mental más claro de lo que está viendo.

Por último, referente a la web de enlaces, nos encontraremos diversas tablas, útiles para clasificar cada recurso con su autor/ra, derechos y enlace. La elección de esta estructura frente a una lista, es debido a su maquetación por celdas mucho más clara. Esto ayuda a la relación entre cada recurso y su información asociada.

5. Diseño responsive, complejidad y estética

A lo largo del proyecto, se ha priorizado la implementación de un diseño responsive utilizando **CSS Grid** y **Flexbox.** Tanto el menú de navegación, como las secciones de cada una de las páginas (exceptuando la de enlaces) a mayores del pie de página cuentan con esta estructura

El objetivo principal de este enfoque es minimizar la cantidad de código necesario y llevar a cabo una optimización del código así como de rendimiento.

Cada diseño de bloque se encuentra dentro <article> o <section> al tratarse de contenido o carga informativa totalmente distinta. A consecuencia, cada uno de estos bloques tiene un diseño responsive propio utilizando las herramientas anteriormente citadas.

En cuanto a la página de detalles, tiene un responsive un poco distinto a los demás. Es en la única web donde se elimina contenido para poder ajustarlo al responsive de manera más adecuada. Este contenido eliminado hace referencia a las flechas que se pueden observar.

Por otro lado, en esta web se ha podido aprovechar el **HTML** de la página de Categoría , facilitando su desarrollo, permitiendo al usuario acceder a otros sitios similares del mismo grupo desde la misma página de detalles del sitio/comida.

En cuanto al diseño y estética, he hablado de él en el apartado anterior de manera más detallada. Si quisiese buscar un adjetivo para tratar de definir lo que buscaba sería: limpieza y facilitar la situación espacial del lector. Recomiendo leer el apartado anterior en donde le dediqué unas líneas más extensas para tratar de comprender mi objetivo.

6. Semántica y accesibilidad

En cuanto a la accesibilidad y semántica, el ajuste de tamaño de fuentes y elementos interactivos accesibles permite que personas con diferentes necesidades puedan navegar el sitio cómodamente desde distintos dispositivos.

A mayores podemos destacar, por ejemplo:

- El uso de un texto alternativo a las imágenes describiendo su contenido de manera concisa pero clara.

- El elemento “scope="row"” en los encabezados de fila y “scope=”col”” en los encabezados de columna. Esto facilita la navegación y accesibilidad de las tablas de la página de enlaces a personas con lectores de pantalla.

- El uso de etiquetas y <b> para enfatizar palabras importantes/claves.

- Se ha verificado que el sitio sea completamente navegable mediante el teclado. Esto permite que personas con discapacidades motoras que no pueden usar ratón logren desplazarse cómodamente por la página usando teclas como **Tab**, **Enter** y las flechas de dirección.

- Se ha tratad de seguir una estructura lógica de encabezados (<h1>, <h2>, etc.) lo que ayuda a las personas de lectores de pantalla a entender la organización del contenido.

- Los colores de fondo y de texto se han elegido con cuidado para asegurar un buen contraste, lo que mejora la legibilidad para personas con baja visión o daltonismo.

7. Publicación a internet.

Para la publicación en Internet del proyecto se ha usado la plataforma Netifly. Esta permite la publicación en línea de la web de una manera sencilla y rápida directamente desde un repositorio de Github. Primero realicé la vinculación de Netifly con mi repositorio, configuré la web y después, una vez que realizaba el **git push**, la página se reiniciaba con la información actualizada. Esta fue la razón por la que escogí esta plataforma.

En cuanto a la configuración del sitio web, una vez se vincula Netifly con el repositorio de Github, se escogió como comando de construcción “npm run parcel:build”. Es decir, se le indica a Netifly que compile el proyecto usando Parcel ante de publicarlo.

Por otro lado, tenemos el directorio público en la carpeta “dist”. Esto hace que Netifly tome los archivos generados en esta carpeta para crear la web online.

Por otro lado, mencionar la funcionalidad útil de Netifly, que en caso de que la web no pueda ser publicada correctamente en internet, informa del error. Además, esta indica donde se encuentra el error y una posible solución mediante su IA.