PEC01-Herramientas HTML CSS

Daniel Marcos Cuesta

WEB - https://rad-fox-84769a.netlify.app/index.html

Github - https://github.com/dmarcosc/vanila-project

Documentación

La web construida trata de un manga japonés y su autor, el proyecto parte de lo trabajado en el primer módulo de la asignatura.

Consta de una portada simple, que permite navegar a las principales páginas y tiene una reproducción de audio que completa la experiencia que el diseño y la temática buscan inculcar. La barra de navegación nos acompañará durante todas las páginas de la web. La página “work” lista en cards las distintas entregas de la obra haciendo una llamada a una API REST y mostrando distintos campos de las entregas, estos cards son clickables y nos redirigen a una vista “detail” donde podremos ver más información y recursos de la entrega. La página “autor” nos cuenta algo más sobre la obra y el autor de manera simple, y la página sources es simplemente una lista de enlaces a las fuentes de los recursos utilizados.

Como norma general se ha trabajado con el navegador Firefox developer recomendado, trabajando la maquetación con filosofía mobile-first, en cuanto a los comportamientos responsive, se han utilizado media-queries y generalmente estas cambian la flex-direction de los layout para poder aprovechar bien el espacio, así como tamaños de recursos y fuentes. En cuanto a diseño se ha intentado mantener una personalidad a lo largo de la web y distintas páginas, utilizando un color principal y otro a modo de hightlight, los enlaces y partes interactuables de la web cuentan con distintas formas de dar feedback al usuario, ya sea en hover para desktop o cambiando color o tamaño para móvil.

La web cumple con los requisitos de navegación y páginas requeridas, utiliza CSS HTML y javascript vanilla, las dos páginas que contienen lógica JS son la de work y la de detalle, en un primer momentop se pensó utilizar js modules y hacer varios ficheros pero debido a la simpleza del código, finalmente se utilizó un único fichero app.js que en ambas vistas realiza una llamada get utilizando axios, que es el paquete extra añadido de npm para el trabajo, de cara a simplificar esta llamada a la API, que en este caso es sencilla, pero en un primer lugar se había planteado otra api que requeria de un token de acceso previo y aquí axios tenía quizá aún más sentido, de todas formas nos ayuda a la hora de hacer esta llamada pues un simple fetch daba problemas de CORS.

Como apunte, aquí se recibe un objeto json que recorremos con una función para mostrar todos los cards con los campos deseados, en caso de que el campo no exista se ha cuidado de no mostrar un null sino no mostrar el campo o mostrar un contenido con sentido, la llamada se ha envuelto en un try catch tratando los casos de error comunes que, en este caso es un status 429 pues la web no acepta muchas peticiones muy seguidas, también se utiliza async await para realizar la llamada de forma asíncrona.

El id en el caso de la llamada al detalle se pasa por la url, y, aunque lo ideal sería hacer una llamada getbyId la respuesta de la API no tiene un funcionamiento correcto, por lo que la solución ha sido volver a hacer un getAll y filtrar el resultado por id.

En un primer momento se ha creado un mock con la respuesta del getAll para trabajar antes de añadir axios, este mock se podría añadir como respuesta en caso de error y por eso se ha mantenido en el proyecto.

Como curiosidad, he intentado añadir un paquete de reproducción de audio para la portada como howler, audio-player o web-audio-player, pero al final hacían más complicado algo que se puede hacer de forma muy sencilla con html y js básico.