Práctica guiada Color Flipper [Opcional]

Ahora que ya sabemos trabajar con **HTML**, **CSS**, **JavaScript**, hemos visto cómo generar un proyecto con **Vite** y el uso de los **imports** y **exports** vamos a crear nuestra primera aplicación web haciendo uso de todas estas tecnologías.

En esta ocasión vamos a generar un **Color Flipper**, una aplicación que nos permitirá modificar el color de nuestro fondo de la aplicación seleccionando entre una gama de colores predefinidos por nosotros.

Para ello vamos a volver a crear una nueva aplicación con **Vite**.

```
npm create vite@latest

✔ Project name: color-flipper
✔ Select a framework: › vanilla
✔ Select a variant: › javascript
```

Accedemos a la carpeta del proyecto, instalamos las dependencias y arrancamos el servidor de desarrollo:

```
cd color-flipper
npm install
npm run dev
```

De esta manera ya tendremos nuestra plantilla por defecto generada por Vite con **JavaScript vanilla** en el enlace que nos indica la consola:

```
VITE v3.0.9  ready in 713 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
```

Manteniendo Ctrl y haciendo click podemos acceder al link.

En este punto limpiaremos nuestro proyecto:

- Eliminaremos el fichero **javascript.svg** que contiene el logo de JS.
- Limpiaremos el fichero **style.css** dejándolo vacío para insertar nuestros propios estilos. Solo vamos a respetar los estilos anidados en **:root** para tener nuestras variables con las fuentes, los tamaños por defecto y el fondo.
- Eliminaremos el fichero **counter.js**.
- En el fichero **main.js** solo dejaremos la línea de importación de **style.css**

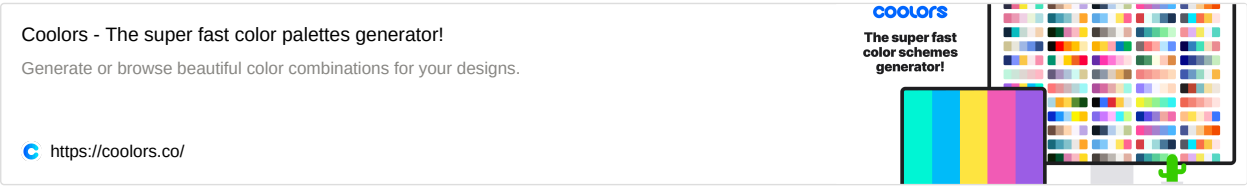
Ahora que tenemos el proyecto con solo la estructura desde 0 podemos empezar a trabajar.

Vamos a modificar nuestro fichero **index.html** para modificar el título por el de nuestra aplicación:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Color Flipper</title>
  </head>
  <body>
    <div id="app"></div>
    <script type="module" src="/main.js"></script>
  </body>
</html>
```

A partir de aquí lo que vamos a hacer es modificar nuestro fichero HTML para crear la estructura del proyecto y el fichero **main.js** para darle toda la funcionalidad necesaria.

Para crear un Color Flipper lo primero que tenemos que hacer es tener registrados una serie de colores para aplicarlos al fondo de nuestra aplicación, así que vamos a visitar la web **Coolors** para recopilar unos 4 o 5 colores.



En este ejemplo hemos recopilado los siguientes:

- Raisin Black: #28262C
- Blue Bell: #998FC7
- Lavender Blue: #D4C2FC
- Magnolia: #F9F5FF
- Resolution Blue: #14248A

Podremos utilizar cualquier color que queramos.

Una vez recopilada esta información lo que tenemos que tener en cuenta es que en nuestro HTML vamos a tener un fondo y dos bloques. Un bloque dirá qué color tenemos seleccionado actualmente, otro de los bloques nos permitirá seleccionar el color a través de un select y el fondo reflejará el color seleccionado.

Para ello tendremos que generar una estructura de datos que nos permita seleccionar dichos datos, como lo puede ser un objeto. Así que vamos a ir a nuestro archivo **main.js** y definir estos colores:

```
import './style.css'

const COLOR_PALETTE = {
  '#28262C': 'Raisin Black',
  '#998FC7': 'Blue Bell',
  '#D4C2FC': 'Lavender Blue',
  '#F9F5FF': 'Magnolia',
  '#14248A': 'Resolution Blue'
}
```

En este caso hemos utilizado el código hexadecimal como clave y el nombre del color como valor, de esta forma tenemos un objeto con los diferentes colores que vamos a utilizar.

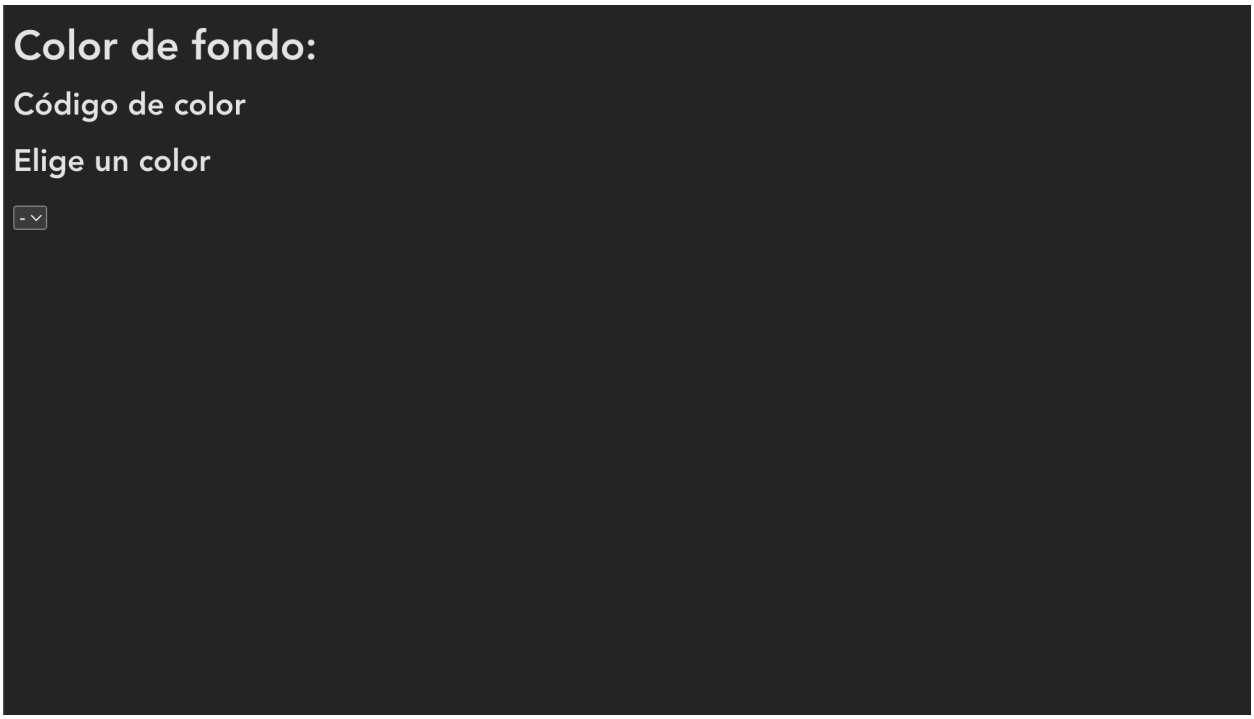
El siguiente paso será crear el esqueleto de nuestro HTML con los bloques que hemos mencionado anteriormente, para ello modificaremos dicho fichero de la siguiente forma:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Color Flipper</title>
```

```
  </head>
  <body>
    <div id="app">
      <div class="block">
        <h1>Color de fondo:</h1>
        <h2>Código de color</h2>
      </div>
      <div class="block">
        <h2>Elige un color</h2>
        <select id="color-picker">
          <option value="@fff">-</option>
        </select>
      </div>
    </div>
    <script type="module" src="/main.js"></script>
  </body>
</html>
```

Hemos creado un div con la clase block que nos dirá el color de fondo (en un futuro insertaremos el código de color dinámicamente) y otro bloque con un select que nos permitirá seleccionarlo (de momento hemos puesto solo el valor del color blanco y sin texto).

En este punto nuestra aplicación se verá tal que así:



Vamos a darle un poco de estilo a nuestra página:

```
:root {
  font-family: Inter, Avenir, Helvetica, Arial, sans-serif;
  font-size: 16px;
  line-height: 24px;
  font-weight: 400;
  background-color: #fff;
```

```
  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  -webkit-text-size-adjust: 100%;
}
* {
  box-sizing: border-box;
}

html,body, #app {
  margin: 0;
  min-height: 100vh;
}

.block {
  background-color: #242424;
  color: rgba(255, 255, 255, 0.87);
  padding: 1rem;
  width: 100%;
  text-align: center;
}

#app{
  display: flex;
  align-items: center;
  flex-direction: column;
  justify-content: center;
  gap: 1rem;
  padding: 1rem;
}

#color-picker {
  min-width: 12.5rem;
  padding: 0.5rem;
  font-size: 1rem;
}
```

Lo único que hemos hecho es definir los tamaños de toda la vista general, repartir de manera flexible los dos bloques dentro del contenedor app y definido unos tamaños y unos márgenes para que todo se vea mucho mejor y más centrado.

Ahora que tenemos nuestra parte visual y la estructura de HTML preparada para funcionar, vamos a implementar dicha funcionalidad en el fichero **main.js**.

Lo primero que vamos a generar es un montón de opciones para el select que incluyan tanto la clave como el valor de los campos de mi objeto **COLOR_PALETTE**.

Para ello haremos uso del método **Object.keys**, con el cual convertiremos nuestro objeto en un array y a través de un forEach generaremos tantos options necesitemos con la longitud del mismo asignándole como valor el color en si y como texto el valor de cada una de las claves:

```
import "./style.css";

const COLOR_PALETTE = {
  "#28262C": "Raisin Black",
  "#998FC7": "Blue Bell",
  "#D4C2FC": "Lavender Blue",
  "#F9F5FF": "Magnolia",
  "#14248A": "Resolution Blue",
};

 Object.keys(COLOR_PALETTE).forEach((color) => {
    const option = document.createElement("option");
    option.value = color;
    option.innerText = COLOR_PALETTE[color];
});
```

Con esto ya tendríamos todas las options de nuestro select. Así que vamos a inyectárselas al select que tenemos en HTML:

```
const colorPickerSelect = document.querySelector("#color-picker");

  Object.keys(COLOR_PALETTE).forEach((color) => {
    const option = document.createElement("option");
    option.value = color;
    option.innerText = COLOR_PALETTE[color];

    colorPickerSelect.append(option);
  })
```

De esta forma hemos recuperado nuestro elemento select con el id **color-picker** y le hemos añadido con el método **append** todas las opciones de **colorOptions**. Podéis probar a utilizar cualquier otro método diferente para acceder a las claves del objeto.

Vamos a optimizar un poco nuestro código y vamos a encapsular lo que hemos hecho en una función, ya que no va a ser lo único que va a ejecutar nuestro script y va a resultar un poco confuso si no llevamos un orden y una encapsulación correcta de funcionalidades:

```
const addOptionsToColorPicker = () => {
  const colorPickerSelect = document.querySelector("#color-picker");

  Object.keys(COLOR_PALETTE).forEach((color) => {
    const option = document.createElement("option");
```

```
    option.value = color;
    option.innerText = COLOR_PALETTE[color];

    colorPickerSelect.append(option);
  })
}

addOptionsToColorPicker()
```

De esta manera ya hemos “compartimentado” una de las funcionalidades que va a tener nuestra aplicación web por si en un futuro necesitamos hacer un módulo independiente con esta misma.

Vamos a comprobar nuestra aplicación para ver si efectivamente está funcionando.

Al haber realizado un bucle a la hora de generar las opciones, podemos aumentar en un futuro nuestro objeto tanto como queramos que nuestra aplicación funcionará exactamente igual.

Lo siguiente que vamos a implementar es el cambio del color de fondo y el texto reflejado una vez seleccionado en nuestro select. Para ello vamos a hacer uso de los escuchadores de eventos.

```
const addEventListenerToColorPicker = () => {
  const colorPickerSelect = document.querySelector("#color-picker");

  colorPickerSelect.addEventListener("change", (event) => {

    const newColor = event.target.value;
    //Almacenamos el código de color

    document.body.style.backgroundColor = newColor;
    //Le aplicamos el background color con el código de color seleccionado

  });
};

addOptionsToColorPicker();
addEventListenerToColorPicker();
```

Lo que acabamos de hacer es añadirle al **select** un escuchador de eventos que cada vez que cambie gracias al evento "**change**" seteará el valor del **target** del evento (es decir, el código hexadecimal en una constante) y lo va a setear mediante estilos en linea en el body del documento.

De esta forma, cada vez que se detecte un cambio en el select recuperará la opción elegida, almacenará el value de cada option y lo seteará como fondo:

Lo que nos faltaría sería setear en el **h2** de nuestro primer bloque el texto con el color seleccionado. Para ello vamos a ir un momento a nuestro HTML y asignarle un id a dicho h2 para poder recuperarlo de mejor manera en JavaScript:

```
<h2 id="color-name">-</h2>
```

También hemos sustituido el texto de ejemplo por un guión al igual que en el select para que tengamos una opción “nula”.

Vamos a nuestro script:

```
const addEventListenerToColorPicker = () => {
  const colorPickerSelect = document.querySelector("#color-picker");
  const colorName = document.querySelector("#color-name");
  colorPickerSelect.addEventListener("change", (event) => {
    const newColor = event.target.value;
    document.body.style.backgroundColor = newColor;
    //Le asignamos a nuestro h2 el valor de la clave que hay almacenada en newColor al igual que hemos hecho en la primera función
    colorName.innerText = COLOR_PALETTE[newColor];
  });
};

addOptionsToColorPicker();
addEventListenerToColorPicker();
```

Al haber recuperado el valor de la clave seleccionada podremos ver en nuestra aplicación que efectivamente nos está seteando el texto adecuado gracias al método **innerText.**

Como extra vamos a indicarle al seteador de texto que si es undefined, como es el caso de la opción dada por guión, que nos ponga un guión gracias a un ternario.

Y además podemos concatenar el código de color para que nos muestre el nombre y su código:

```
const colorNameText = `${COLOR_PALETTE[newColor]} | ${newColor}`
    colorName.innerText = COLOR_PALETTE[newColor] ? colorNameText : "-";
```

Ya habríamos terminado de completar nuestra funcionalidad. Ahora el reto es darle nuestro propio toque, modificar la forma de elegir los colores (por ejemplo, a través de un input de tipo color) y completar nuestra primera aplicación web creada con **Vite**, **HTML**, **CSS** y **JavaScript**.


Color de fondo:
Código de color
-
-