Consumo de API 2.0

En esta ocasión consumiremos la API de RestCountries, es una API que proporciona información sobre países de todo el mundo, como su nombre, capital, moneda, idioma, bandera, entre otros datos interesantes. Los desarrolladores pueden utilizar esta API en sus aplicaciones para mostrar información sobre países en sus páginas web o aplicaciones móviles. Además, RestCountries.com es una excelente herramienta para aquellos que deseen aprender más sobre la geografía y los datos de los países.

Iniciaremos con el HTML:

```html
<body>
    <div class="container">
        <h1>Guia de Paises</h1>
        <div class="container-busqueda">
            <input type="text" id="nombre-pais" placeholder="Ingresa el nombre de un pais">
            <button class="btnBusqueda">Buscar</button>
        </div>
        <div class="resultado"></div>
    </div>
    <script src="js/script.js"></script>
</body>
```

Agregamos los estilos CSS:

```css
@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&display=swap");

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
}

html {
  font-size: 62.5%;
}

body {
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background-color: blueviolet;
}

.container {
  width: 500px;
  min-height: 150px;
  padding: 20px;
  flex-direction: column;
  background-color: white;
  border-radius: 20px;
  box-shadow: 0 0 40px rgba(0, 0, 0, 0.6);
}
```

```css
h1 {
  color: blueviolet;
  text-transform: uppercase;
  text-shadow: 1px 1px black;
  margin-bottom: 1.5rem;
}

.container-busqueda {
  display: grid;
  grid-template-columns: 9fr 3fr;
  gap: 2rem;
}

input {
  padding: 5px 10px;
  font-weight: 500;
  border: none;
  outline: none;
  border-bottom: 2px solid blueviolet;
}

button {
  background-color: blueviolet;
  border: none;
  color: white;
  border-radius: 2rem;
  cursor: pointer;
}

.resultado {
  margin-top: 10px;
}
```

@hdtoledo

```css
.resultado .flag-img {
  display: block;
  width: 45%;
  min-width: 7.5em;
  margin: 1.8em auto 1.2em auto;
}

.resultado h2 {
  font-weight: 600;
  color: #222a43;
  text-align: center;
  text-transform: uppercase;
  letter-spacing: 2px;
  margin-bottom: 1.8em;
}

.resultado .data-wrapper {
  margin-bottom: 1em;
  letter-spacing: 0.3px;
}
.resultado h4 {
  display: inline;
  font-weight: 500;
  color: #222a43;
}

.resultado span {
  color: #5d6274;
}

.resultado h3 {
  margin-top: 10px;
  text-align: center;
  font-size: 1.2em;
  font-weight: 400;
  color: #ff465a;
}
```

```css
.resultado .loading {
  margin-top: 10px;
  margin-bottom: -10px;
  color: black;
  font-size: 15px;
  text-align: center;
}
```

@hdtoledo

A continuación, vamos con el código JS, realizaremos una búsqueda en la API Rest Countries para obtener información sobre un país en particular y mostrar los resultados.

Primero inicializaremos las variables:

```
const inputEl = document.querySelector("#nombre-pais");
const searchBtnEl = document.querySelector(".btnBusqueda");
const result = document.querySelector(".resultado");
```

A continuación, realizaremos una función asíncrona, esto significa que puede esperar a que se complete una tarea antes de continuar ejecutando el resto del código. En este caso, la tarea que se espera es la respuesta de la API Rest Countries.

```
async function getResults() {

    let countryName = inputEl.value;
    try {
        result.innerHTML = `<h2 class="loading">Cargando Resultados ... </h2>`;
        let fetchUrl = `https://restcountries.com/v3.1/name/${countryName}?fullText=true`;
        let data = await fetch(fetchUrl).then((res) => res.json());

        result.innerHTML = `
    <img src="${data[0].flags.svg}" class="flag-img">
    <h2>${data[0].name.common}</h2>
    <div class="wrapper">
        <div class="data-wrapper">
            <h4>Capital:</h4>
            <span>${data[0].capital[0]}</span>
        </div>
    </div>
    <div class="wrapper">
        <div class="data-wrapper">
            <h4>Continente:</h4>
            <span>${data[0].continents[0]}</span>
        </div>
    </div>
     <div class="wrapper">
        <div class="data-wrapper">
            <h4>Poblacion:</h4>
            <span>${data[0].population}</span>
        </div>
    </div>
    <div class="wrapper">
        <div class="data-wrapper">
            <h4>Moneda:</h4>
            <span>${data[0].currencies[Object.keys(data[0].currencies)].name
            } - ${Object.keys(data[0].currencies)[0]}</span>
        </div>
    </div>
     <div class="wrapper">
        <div class="data-wrapper">
            <h4>Idioma:</h4>
            <span>${Object.values(data[0].languages)
                .toString()
                .split(",")
                .join(", ")}</span>
        </div>
    </div>
        `;
    }

    catch (error) {
        if (countryName.length == 0) {
            result.innerHTML = `<h3>El campo no puede estar vacio !</h3>`;
        } else {
            result.innerHTML = `<h3>Por favor ingrese un pais correcto.</h3>`;
        }
    }
}
```

@hdtoledo

La función **getResults()** es una función asíncrona que se utiliza para buscar información de un país utilizando una API y mostrar los resultados en una página web.

La función comienza declarando una variable llamada **countryName** que almacena el valor que se ingresa en un campo de entrada HTML con el id nombre-pais. Este valor se utilizará más adelante para hacer la consulta a la API.

A continuación, se utiliza una declaración **try...catch** para manejar los errores que puedan surgir al realizar la consulta a la API. Dentro del bloque try, se actualiza el contenido de un elemento HTML con la clase resultado para mostrar un mensaje de "Cargando resultados..." mientras se realiza la consulta.

Luego, se construye la URL de la API concatenando el valor del campo de entrada **countryName** en la URL base de la API. El **fetch** se utiliza para hacer una solicitud a la API con la URL creada y se convierte la respuesta en un objeto JSON utilizando el método json().

Finalmente, se actualiza el contenido del elemento HTML con la clase resultado para mostrar la información del país utilizando los datos obtenidos de la API. Se utilizan varias plantillas de cadena para mostrar la bandera, el nombre, la capital, el continente, la población, la moneda y el idioma del país en elementos HTML.

Si se produce un error al hacer la consulta, la función **catch** maneja los errores y actualiza el contenido del elemento HTML con la clase resultado para mostrar un mensaje de error personalizado, dependiendo del tipo de error que se haya producido. Si el campo de entrada está vacío, se muestra un mensaje que indica que el campo no puede estar vacío. Si el nombre del país no se puede encontrar en la API, se muestra un mensaje que indica que se debe ingresar un país válido.

```
searchBtnEl.addEventListener("click", getResults);
```

Por ultimo habilitamos el botón mediante el método de addEventListener para que al dar click se ejecuten la función.