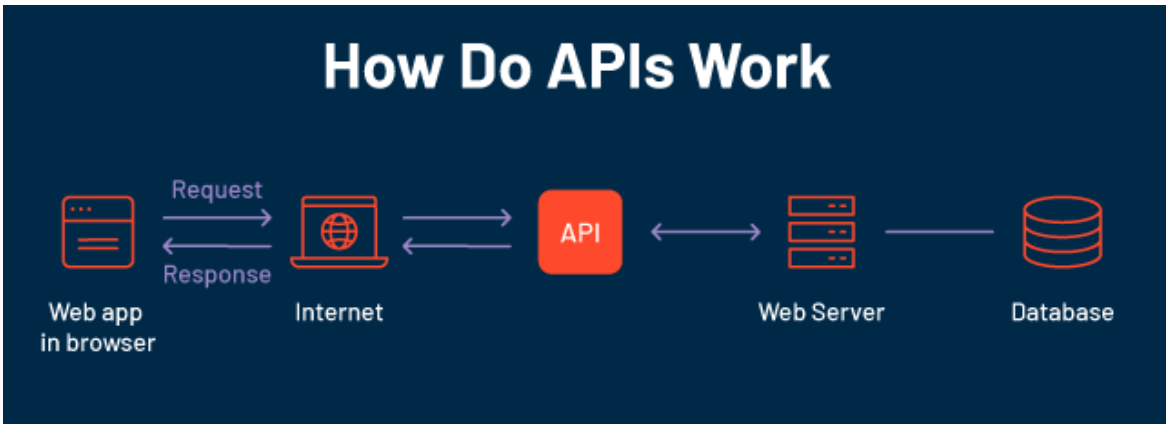
API JS ASYNC AWAIT FETCH



Una API (Application Programming Interface, por sus siglas en inglés) es un conjunto de reglas y protocolos que permiten la comunicación entre diferentes software o componentes de software. Básicamente, una API define cómo deben interactuar dos aplicaciones para intercambiar información o llevar a cabo ciertas acciones.

Una API actúa como una capa de abstracción que oculta la complejidad interna de un sistema y expone solo las funcionalidades necesarias para que otros programas puedan utilizarlo de manera segura y eficiente. Proporciona un conjunto de métodos, funciones y reglas predefinidas que permiten a los desarrolladores acceder y utilizar los servicios o datos de una aplicación o plataforma.

Las API se utilizan ampliamente en el desarrollo de software para facilitar la integración entre diferentes aplicaciones, sistemas o servicios. Por ejemplo, muchas redes sociales, servicios de pago en línea y servicios de mapas proporcionan API públicas que permiten a los desarrolladores crear aplicaciones que utilicen sus funciones o datos. De esta manera, los desarrolladores pueden aprovechar las capacidades de otras aplicaciones o servicios sin necesidad de conocer todos los detalles internos de implementación.

Iniciemos con nuestro scaffolding del html:

index.html | script.js | style.css

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link rel="stylesheet" href="css/style.css" />
    <title>Manejo de una API</title>
  </head>
  <body>
    <div class="contenedor" id="contenedor"></div>

    <div class="paginacion">
      <button id="btnAnterior">Anterior</button>
      <button id="btnSiguiente">Siguiente</button>
    </div>
    <script src="js/script.js"></script>
  </body>
</html>
```

1. **<link rel="preconnect" href="https://fonts.googleapis.com">**: Esta línea indica al navegador que establezca una conexión previa con el servidor ubicado en "https://fonts.googleapis.com". Esto significa que el navegador iniciará una conexión de red antes de que realmente necesite solicitar recursos de ese servidor en particular. Al establecer esta conexión previa, se reduce el tiempo de respuesta y se mejora el rendimiento cuando se solicitan fuentes de Google Fonts.

2. **<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>**: Esta línea es similar a la anterior, pero se refiere a otro servidor específico de Google Fonts ubicado en "https://fonts.gstatic.com". El atributo **crossorigin** indica que la solicitud de recursos a este servidor se realizará desde una ubicación diferente (en este caso, el dominio donde se encuentra el código HTML). Esto es necesario para evitar problemas de seguridad relacionados con las políticas de intercambio de recursos entre dominios (CORS, por sus siglas en inglés).

En resumen, estas líneas de código se utilizan para mejorar el rendimiento al establecer conexiones previas con los servidores de Google Fonts, permitiendo que la carga de fuentes se realice de manera más eficiente en el navegador del usuario.

Ahora nuestro Css:

```css
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;700&display=swap');

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

body{
    background: #e0e0e0;
    color: #1d1d1d;
}

.contenedor {
    width: 90%;
    max-width: 1000px;
    margin: 40px auto 100px auto;

    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 40px;
}

.contenedor .pelicula {
    text-align: center;
}

.contenedor .pelicula .titulo {
    font-size: 16px;
    font-weight: 600;
}

.contenedor .pelicula .poster {
    width: 100%;
    margin-bottom: 10px;
    border-radius: 15px;
}
```

```css
.paginacion {
    position: fixed;
    bottom: 0;
    background: #100a1f;
    width: 100%;
    display: flex;
    justify-content: center;
    gap: 20px;
    padding: 10px;
}

.paginacion button {
    cursor: pointer;
    border: none;
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
    height: 50px;
    width: 200px;
    background: #241744;
    color: #fff;
    border-radius: 100px;
    font-family: 'Montserrat', sans-serif;
    font-weight: 600;
    transition: .3s ease all;
}

.paginacion button:hover {
    background: #137c32;
}
```

@hdtoledo

Y por último nuestro JS:

```js
let pagina = 1;
const btnAnterior = document.getElementById('btnAnterior');
const btnSiguiente = document.getElementById('btnSiguiente');

btnSiguiente.addEventListener('click', () => {
    if(pagina < 1000){
        pagina += 1;
        cargarPeliculas();
    }
});

btnAnterior.addEventListener('click', () => {
    if(pagina > 1){
        pagina -= 1;
        cargarPeliculas();
    }
});
```

```js
const cargarPeliculas = async() => {
    try {
        const respuesta = await fetch(`https://api.themoviedb.org/3/movie/popular?api_key=192e0b9821564f26f52949758ea3c473&language=es-MX&page=${pagina}`);

        console.log(respuesta);

        // Si la respuesta es correcta
        if(respuesta.status === 200){
            const datos = await respuesta.json();

            let peliculas = '';
            datos.results.forEach(pelicula => {
                peliculas += `
                    <div class="pelicula">
                        <img class="poster" src="https://image.tmdb.org/t/p/w500/${pelicula.poster_path}">
                        <h3 class="titulo">${pelicula.title}</h3>
                    </div>
                `;
            });

            document.getElementById('contenedor').innerHTML = peliculas;

        } else if(respuesta.status === 401){
            console.log('Pusiste la llave mal');
        } else if(respuesta.status === 404){
            console.log('La pelicula que buscas no existe');
        } else {
            console.log('Hubo un error y no sabemos que paso');
        }

    } catch(error){
        console.log(error);
    }

}

cargarPeliculas();
```

@hdtoledo

1. **let pagina = 1;**: Esta línea declara una variable llamada **pagina** y le asigna el valor inicial de 1. Esta variable se utilizará para llevar un seguimiento de la página actual en la carga de películas.
2. **const btnAnterior = document.getElementById('btnAnterior');**: Esta línea busca un elemento HTML con el identificador **'btnAnterior'** y lo asigna a la constante **btnAnterior**. Se espera que este elemento sea un botón en el documento HTML.
3. **const btnSiguiente = document.getElementById('btnSiguiente');**: Similar a la línea anterior, esta línea busca un elemento HTML con el identificador **'btnSiguiente'** y lo asigna a la constante **btnSiguiente**. También se espera que este elemento sea un botón en el documento HTML.
4. **btnSiguiente.addEventListener('click', () => { ... });**: Esta línea agrega un "event listener" al botón **btnSiguiente**. Cuando se hace clic en este botón, se ejecuta la función de flecha proporcionada.
5. **if(pagina < 1000){ ... }**: Esta línea verifica si el valor de **pagina** es menor que 1000. Si es así, se ejecuta el código dentro del bloque de llaves.
6. **pagina += 1;**: Esta línea incrementa el valor de **pagina** en 1.
7. **cargarPeliculas();**: Esta línea llama a la función **cargarPeliculas()**. Esta función se encarga de cargar las películas correspondientes a la página actual.
8. **btnAnterior.addEventListener('click', () => { ... });**: Similar a la línea 4, esta línea agrega un "event listener" al botón **btnAnterior**. Cuando se hace clic en este botón, se ejecuta la función de flecha proporcionada.
9. **if(pagina > 1){ ... }**: Esta línea verifica si el valor de **pagina** es mayor que 1. Si es así, se ejecuta el código dentro del bloque de llaves.
10. **pagina -= 1;**: Esta línea decrementa el valor de **pagina** en 1.
11. **cargarPeliculas();**: Nuevamente, se llama a la función **cargarPeliculas()** para cargar las películas correspondientes a la página actual.
12. **const cargarPeliculas = async() => { ... }**: Esta línea declara una función asíncrona llamada **cargarPeliculas**. Esta función se encarga de realizar una solicitud a la API de The Movie Database (TMDB) para obtener películas populares en función de la página actual.
13. **const respuesta = await fetch(...);**: Esta línea utiliza la función **fetch** para hacer una solicitud a la API de TMDB. La URL de la solicitud incluye la página actual y una clave de API para autenticación. La respuesta de la solicitud se asigna a la constante **respuesta**.
14. **console.log(respuesta);**: Se imprime la respuesta de la solicitud en la consola.
15. **if(respuesta.status === 200){ ... }**: Esta línea verifica si el estado de la respuesta es 200, lo que indica que la solicitud fue exitosa. Si es así, se ejecuta el código dentro del bloque de llaves.
16. **const datos = await respuesta.json();**: Esta línea extrae los datos de la respuesta y los convierte en un objeto JavaScript utilizando el método **json()**. Los datos se asignan a la constante **datos**.
17. **let peliculas = '';**: Se declara una variable llamada **peliculas** y se le asigna una cadena vacía. Esta variable se utilizará para construir una cadena de HTML que representa las películas obtenidas.

18. **datos.results.forEach(pelicula => { ... })**: Se itera sobre el arreglo de películas (**results**) obtenido de los datos. Para cada película, se ejecuta el código dentro del bloque de llaves.

19. **peliculas +=** ...**;**: Se agrega una cadena de HTML que representa una película a la variable **peliculas**. La imagen del póster y el título se toman de cada objeto de película.

20. **document.getElementById('contenedor').innerHTML = peliculas;**: Se selecciona un elemento HTML con el identificador **'contenedor'** y se asigna la cadena de películas a su propiedad **innerHTML**. Esto inserta el contenido HTML generado dentro del elemento en el documento.

21. Las líneas 23 a 29 manejan diferentes escenarios en función del estado de la respuesta de la solicitud. Dependiendo del estado, se muestran mensajes específicos en la consola.

22. **console.log(error);**: Si se produce un error durante la ejecución de la función **cargarPeliculas**, se captura y se muestra en la consola.

23. **cargarPeliculas();**: Finalmente, se llama a la función **cargarPeliculas()** una vez para cargar las películas iniciales en la página.

En resumen, este código se encarga de controlar los botones de "anterior" y "siguiente" para cambiar la página de películas y muestra las películas correspondientes en el documento HTML utilizando la API de The Movie Database.