

Asincronía en JavaScript y APIs del Navegador





Repasso: Asincronía en JavaScript

Explicación:

Los callbacks son funciones que se pasan como argumento a otras funciones y se ejecutan después de que la función principal haya completado su operación. Son una forma básica de manejar la asincronía en JavaScript

Promesas (Promise)

Explicación:

Las promesas son un objeto que representa la eventual finalización (o falla) de una operación asíncrona y su valor resultante.

```
function obtenerDatos() {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => {  
            let exito = true; // Cambia a `false` para  
            simular un error  
            if (exito) {  
                resolve("Datos obtenidos con éxito");  
            } else {  
                reject("Error al obtener los datos");  
            }  
        }, 2000);  
    });  
}
```

```
obtenerDatos()  
.then(resultado => console.log(resultado))  
.catch(error => console.error(error));
```



```
async function obtenerDatos() {  
  try {  
    let respuesta = await  
    obtenerDatosSimulados();  
    console.log(respuesta);  
  } catch (error) {  
    console.error(error);  
  }  
}  
  
function obtenerDatosSimulados() {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      resolve("Datos obtenidos con await");  
    }, 2000);  
  });  
}  
  
obtenerDatos();
```

async y await

Las palabras clave `async` y `await` permiten escribir código asincrónico que parece síncrono, haciendo el código más fácil de entender y mantener.



APIs del Navegador



Tipos de Solicitudes HTTP

GET

Obtener datos.

POST

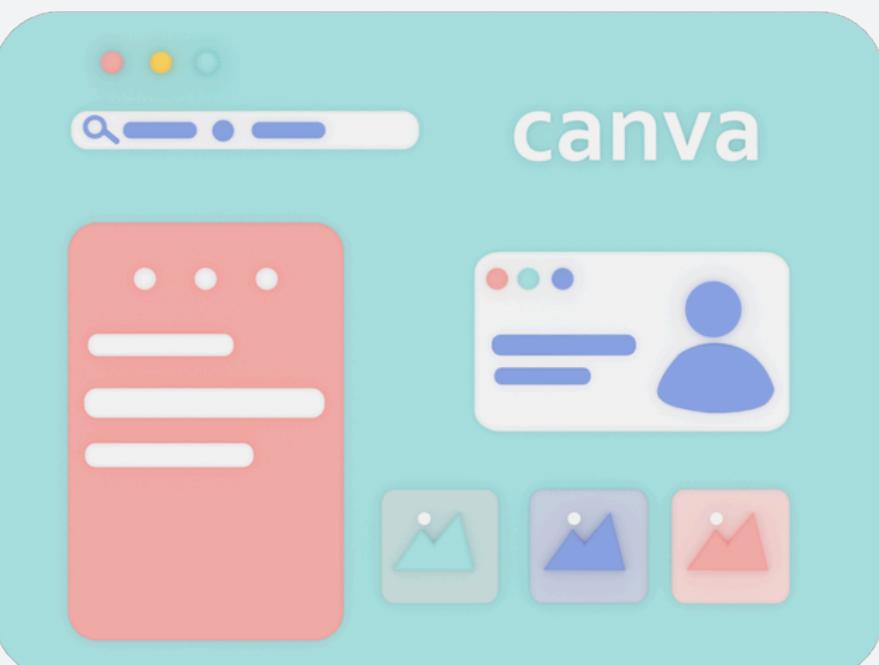
Enviar datos para crear un nuevo recurso.

PUT

Actualizar un recurso existente.

DELETE

Eliminar un recurso



```
fetch('https://jsonplaceholder.typicode.co
m/posts', {
  method: 'GET'
}
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:',
error));
```



Ejemplo Práctico: Consumiendo una API

Crear una pequeña aplicación que realice una solicitud GET para obtener datos y los muestre en la consola.

```
async function obtenerPublicaciones() {  
    try {  
        let respuesta = await  
fetch('https://jsonplaceholder.typicode.com/posts');  
        let datos = await respuesta.json();  
        console.log(datos);  
    } catch (error) {  
        console.error("Error al obtener las publicaciones:",  
error);  
    }  
}  
  
obtenerPublicaciones();
```

Manejo de Errores



Es importante manejar los errores para asegurarse de que la aplicación responda adecuadamente si algo sale mal. Los errores pueden ocurrir debido a problemas de red, URLs incorrectas, o respuestas inesperadas del servidor.

```
async function obtenerPublicaciones() {  
  try {  
    let respuesta = await fetch('https://jsonplaceholder.typicode.com/posts');  
    if (!respuesta.ok) throw new Error('Error en la solicitud');  
    let datos = await respuesta.json();  
    console.log(datos);  
  } catch (error) {  
    console.error("Error al obtener las publicaciones:", error);  
  }  
}  
  
obtenerPublicaciones();
```

Crear una Mini Galería de Imágenes Favoritas

Objetivo

Aplicar los conceptos de APIs, localStorage, y asincronismo en JavaScript para construir una mini galería de imágenes donde los estudiantes puedan buscar, visualizar y guardar imágenes como favoritas.

Duración estimada: 20 a 60 minutos

NOTA: Un punto por cada requerimiento.



Requerimientos



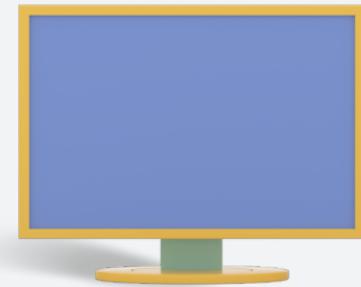
BUSCAR IMÁGENES DESDE UNA API:

- Utiliza la API pública de Unsplash o la API de Pixabay para realizar búsquedas de imágenes.
- Los estudiantes deben crear un campo de búsqueda donde el usuario pueda ingresar un término (por ejemplo, "cats").
- Cuando el usuario presione el botón de búsqueda, se debe realizar una solicitud GET a la API y mostrar las imágenes resultantes en la pantalla.



VISUALIZAR LAS IMÁGENES

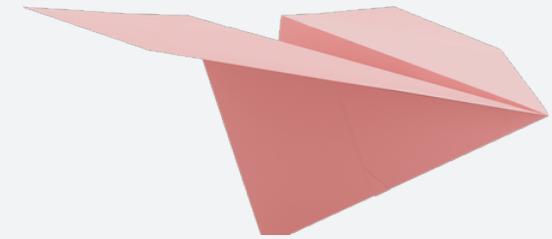
- Muestra un máximo de 10 imágenes en una cuadrícula.
- Cada imagen debe incluir un botón o un ícono de "favorito" que permita al usuario guardar la imagen en sus favoritos.



GUARDAR IMÁGENES EN LOCALSTORAGE

- Al hacer clic en el botón o ícono de "favorito", la información de la imagen (por ejemplo, la URL de la imagen) debe guardarse en localStorage.
- Los estudiantes deben asegurarse de que los favoritos persistan incluso si se recarga la página.

Requerimientos



MOSTRAR IMÁGENES FAVORITAS

- Crea una sección en la página donde se muestren las imágenes guardadas como favoritas.
- Esta sección debe cargarse automáticamente con las imágenes almacenadas en localStorage cuando la página se abre.

ELIMINAR FAVORITOS

Permite que el usuario elimine imágenes de sus favoritos. Cuando el usuario elimine una imagen de los favoritos, esta debe desaparecer de la sección de favoritos y también eliminarse de localStorage.

MANEJO DE ERRORES

Si la API devuelve un error (por ejemplo, si no hay resultados), los estudiantes deben mostrar un mensaje de error adecuado en la interfaz.

Reto de programación

Crea una calculadora básica que permita realizar operaciones aritméticas simples (suma, resta, multiplicación y división) entre dos números. Debes implementar esta calculadora utilizando funciones en JavaScript.

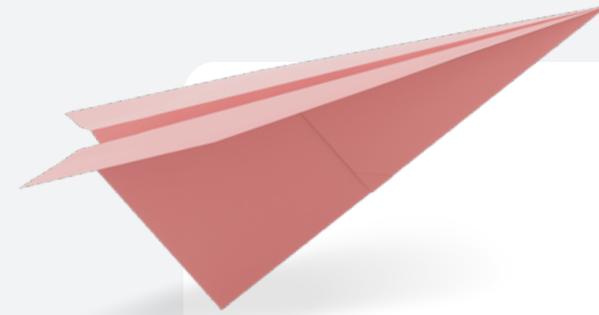
Duración estimada: 20 a 60 minutos

NOTA: Cantidad de punto maxima son 5.



Requisitos:

1. **Solicitar al usuario que ingrese dos números a través del método prompt.**
2. **Solicitar al usuario que ingrese la operación que desea realizar (+, -, *, /).**
3. **Implementa funciones para cada operación (por ejemplo, sumar(a, b), restar(a, b), multiplicar(a, b), dividir(a, b)).**
4. **Utiliza una estructura de control (if-else o switch) para determinar qué operación se debe realizar en función del operador ingresado por el usuario.**
5. **Muestra el resultado de la operación al usuario usando alert o console.log.**
6. **Maneja los casos especiales:**
 - **Si el usuario intenta dividir por cero, muestra un mensaje de error.**
 - **Si el usuario ingresa una operación no válida, muestra un mensaje de advertencia.**



**¿Tienes alguna
pregunta?**

