Taller Práctico: Dominando la Programación Asíncrona

Julian F. Latorre
Bootcamp Desarrollo Web Full Stack

Proyecto: Galería de Fotografías Dinámica

</> Introducción

¡Bienvenidos desarrolladores! En este taller práctico, construiremos una galería de fotografías dinámica utilizando programación asíncrona. Aprenderemos a manejar solicitudes HTTP, procesar datos y actualizar la interfaz de usuario de manera eficiente.

1. ☁ Ejercicio 1: Recuperar Imágenes desde una API

Utiliza la API de Unsplash para obtener imágenes de alta calidad. Crea una función asíncrona que recupere 5 imágenes aleatorias.

```
const UNSPLASH_API_KEY = 'TU_API_KEY_AQUI';
const UNSPLASH_API_URL = 'https://api.unsplash.com/photos/random';

async function obtenerImagenes(cantidad) {
  const imagenes = [];
  // Tu código aquí
  // Pista: Utiliza fetch y parámetros de consulta adecuados
  return imagenes;
}

// Prueba tu función
obtenerImagenes(5)
  .then(imagenes => console.log('Imágenes obtenidas:', imagenes))
  .catch(error => console.error('Error en la solicitud:', error));
```

2. 🖼 Ejercicio 2: Procesamiento de Metadatos

Procesa los metadatos de las imágenes (título, autor, fecha) y organízalos en un formato adecuado para su visualización.

```
function procesarMetadatos(datosImagen) {
  return new Promise((resolve, reject) => {
    // Tu código aquí
    // Pista: Extrae y formatea la información relevante
  });
}

// Prueba tu función
procesarMetadatos({
  title: "Paisaje urbano",
  author: "Juan Pérez",
  created_at: "2024-01-15T10:30:00Z"
})
  .then(metadata => console.log('Metadatos procesados:', metadata))
  .catch(error => console.error('Error en el procesamiento:', error
    ));
```

3. Ejercicio 3: Mostrar la Galería

Crea una función que combine las imágenes y sus metadatos, y los muestre en la página. Utiliza async/await para manejar las operaciones asíncronas.

```
async function montarGaleria() {
  try {
    const imagenes = await obtenerImagenes(5);
    const galeria = document.getElementById('galeria-dinamica');

    for (const imagen of imagenes) {
      const metadata = await procesarMetadatos(imagen);
      // Tu código aquí
      // Pista: Crea elementos HTML para cada imagen y sus datos
    }
  } catch (error) {
    console.error('Error al montar la galería:', error);
  }
}

// Inicia la galería
montarGaleria();
```

4. Ejercicio 4: Sistema de Comentarios

Implementa un sistema de comentarios asíncrono con manejo de errores y retroalimentación al usuario.

```
function enviarComentario(comentario, callback) {
  setTimeout(() => {
    const exito = Math.random() > 0.1; // 90% de éxito en el envío
    if (exito) {
      callback(null, 'Comentario guardado exitosamente');
    } else {
      callback(new Error('Error al guardar el comentario'));
    }
```

```
  }, 1000); // Simula latencia de red
}

// Usa la función así:
enviarComentario("  Excelente   fotografía!", (error, respuesta) => {
  if (error) {
    console.error('Error:', error.message);
  } else {
    console.log(respuesta);
  }
});
```

5. Ejercicio 5: Optimización de Imágenes

Crea una función que procese las imágenes de forma asíncrona, aplicando optimizaciones básicas como redimensionamiento y compresión.

```
function optimizarImagenAsync(imagen) {
  return new Promise((resolve) => {
    setTimeout(() => {
      // Simula el procesamiento de la imagen
      const imagenOptimizada = `[Optimizada]${imagen}`;
      resolve(imagenOptimizada);
    }, 1000);
  });
}

async function procesarGaleriaOptimizada(imagenes) {
  const imagenesOptimizadas = [];
  // Tu código aquí
  // Pista: Usa Promise.all para procesar todas las imágenes en
    paralelo
  return imagenesOptimizadas;
}

// Prueba la función
const imagenesMuestra = ['img1.jpg', 'img2.jpg', 'img3.jpg', 'img4.
    jpg'];
procesarGaleriaOptimizada(imagenesMuestra)
  .then(resultado => console.log('Galería optimizada:', resultado))
    ;
```

6. Proyecto Final: Galería Dinámica Completa

Combina todas las misiones anteriores para crear una Galería Dinámica completa. Incluye:

- Carga asíncrona de imágenes
- Procesamiento de metadatos

- Sistema de comentarios
- Optimización de imágenes

Consejo Práctico

Recuerda siempre manejar los errores adecuadamente y proporcionar retroalimentación al usuario durante las operaciones asíncronas.

7. Recursos Adicionales

- https://unsplash.com/developersDocumentación de la API de Unsplash
- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global$_{O}bjects/PromiseMDNWebDocs$: I

Conclusión

La programación asíncrona es fundamental en el desarrollo web moderno. Dominar estos conceptos te permitirá crear aplicaciones más eficientes y responsivas.