Taller Práctico: Ejecución de Scripts con NodeJS

Julian F. Latorre
Bootcamp Desarrollo Web Full Stack

1. Introducción

Este taller práctico te guiará a través de varios ejercicios para reforzar tus conocimientos sobre la ejecución de scripts con NodeJS. Cada ejercicio incluye un código base y instrucciones para completar la tarea.

2. Ejercicio 1: Hola Mundo Personalizado

Objetivo

Crear un script que salude al usuario con su nombre.

```
// Código base
const readline = require('readline').createInterface({
  input: process.stdin,
  output: process.stdout
});

// Tu código aquí
```

Instrucciones:

1. Usa `readline` para preguntar el nombre al usuario.
2. Imprime un saludo personalizado.
3. Cierra la interfaz de readline.

3. Ejercicio 2: Calculadora Simple

Objetivo

Implementar una calculadora que realice operaciones básicas.

```
// Código base
function sumar(a, b) {
  return a + b;
}

// Implementa las funciones restar, multiplicar y dividir

function calculadora(operacion, a, b) {
  // Tu código aquí
}

// Ejemplo de uso
console.log(calculadora('sumar', 5, 3));
```

Instrucciones:

1. Implementa las funciones `restar`, `multiplicar` y `dividir`.
2. Completa la función `calculadora` para que use la operación correcta.
3. Maneja el caso de división por cero.

4. Ejercicio 3: Lector de Archivos

Objetivo

Crear un script que lea el contenido de un archivo y lo muestre en consola.

```
// Código base
const fs = require('fs');

function leerArchivo(nombreArchivo) {
  // Tu código aquí
}

// Ejemplo de uso
leerArchivo('ejemplo.txt');
```

Instrucciones:

1. Usa el método `fs.readFile` para leer el archivo.
2. Maneja los errores si el archivo no existe.
3. Imprime el contenido del archivo en la consola.

5. Ejercicio 4: Servidor Web Simple

Objetivo

Crear un servidor web que responda con un mensaje personalizado.

```
// Código base
const http = require('http');

const servidor = http.createServer((req, res) => {
  // Tu código aquí
});

servidor.listen(3000, () => {
  console.log('Servidor corriendo en http://localhost:3000/');
});
```

Instrucciones:

1. Configura el servidor para responder con un mensaje HTML.
2. Incluye la fecha y hora actual en el mensaje.
3. Bonus: Haz que el servidor responda diferente según la ruta solicitada.

6. Ejercicio 5: Gestor de Tareas Asíncrono

Objetivo

Implementar un gestor de tareas que use operaciones asíncronas.

```
// Código base
const tareas = [
  { id: 1, descripcion: 'Hacer la compra', completada: false },{ id
    : 2, descripcion: 'Estudiar NodeJS', completada: true },
  { id: 3, descripcion: 'Hacer ejercicio', completada: false }
];

function obtenerTareas() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve(tareas);
    }, 1000);
  });
}

function marcarComoCompletada(id) {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const tarea = tareas.find(t => t.id === id);
      if (tarea) {
        tarea.completada = true;
        resolve(tarea);
      } else {
        reject(new Error('Tarea no encontrada'));
      }
    }, 500);
  });
}
```

```
async function gestionarTareas() {
  // Tu código aquí
}

gestionarTareas();
```

Instrucciones:

1. Implementa la función `gestionarTareas` usando async/await.
2. Obtén todas las tareas y muéstralas en consola.
3. Marca la tarea con id 1 como completada.
4. Muestra las tareas actualizadas.
5. Maneja los errores adecuadamente.

7. Ejercicio 6: Módulo Personalizado

Objetivo

Crear un módulo personalizado y utilizarlo en otro script.

Archivo: utilidades.js

```
// Tu código aquí
// Exporta funciones para:
// 1. Generar un número aleatorio entre un mínimo y un máximo
// 2. Convertir una cadena a mayúsculas
// 3. Contar el número de vocales en una cadena
```

Archivo: index.js

```
const utilidades = require('./utilidades');

// Tu código aquí
// Usa las funciones del módulo utilidades
```

Instrucciones:

1. En `utilidades.js`, implementa y exporta las funciones requeridas.
2. En `index.js`, importa el módulo y usa todas las funciones.
3. Muestra los resultados en la consola.

8. Ejercicio 7: Manejador de Eventos

Objetivo

Crear un manejador de eventos personalizado.

```
const EventEmitter = require('events');

class GestorTareas extends EventEmitter {
  constructor() {
    super();
    this.tareas = [];
  }

  // Implementa métodos para:
  // 1. Agregar una tarea
  // 2. Marcar una tarea como completada
  // 3. Listar todas las tareas
}

const gestor = new GestorTareas();

// Configura los listeners de eventos aquí

// Tu código para usar el GestorTareas
```

Instrucciones:

1. Implementa los métodos en la clase `GestorTareas`.
2. Emite eventos apropiados en cada método.
3. Configura listeners para cada evento.
4. Usa el `GestorTareas` para agregar, completar y listar tareas.

9. Ejercicio 8: API REST Simple

Objetivo

Crear una API REST simple para gestionar una lista de tareas.

```
const http = require('http');

let tareas = [];

const servidor = http.createServer((req, res) => {
  if (req.method === 'GET' && req.url === '/tareas') {
    // Implementa: Listar todas las tareas
  } else if (req.method === 'POST' && req.url === '/tareas') {
    // Implementa: Agregar una nueva tarea
  } else if (req.method === 'PUT' && req.url.startsWith('/tareas/')
    ) {
    // Implementa: Actualizar una tarea existente
  } else if (req.method === 'DELETE' && req.url.startsWith('/tareas
    /')) {
    // Implementa: Eliminar una tarea
  } else {
    res.writeHead(404);
    res.end('Ruta no encontrada');
```

```
  }
});

servidor.listen(3000, () => {
  console.log('API corriendo en http://localhost:3000');
});
```

Instrucciones:

1. Implementa la lógica para cada tipo de solicitud (GET, POST, PUT, DELETE).
2. Usa `JSON.parse()` y `JSON.stringify()` para manejar los datos.
3. Asegúrate de manejar los errores y enviar las respuestas apropiadas.
4. Prueba la API usando herramientas como cURL o Postman.

10. Conclusión

Estos ejercicios te han proporcionado una experiencia práctica en varios aspectos de la ejecución de scripts con NodeJS. Recuerda que la práctica constante es clave para dominar estas habilidades. No dudes en experimentar y expandir estos ejercicios para profundizar tu comprensión.

Recursos adicionales:

- Documentación oficial de NodeJS: `https://nodejs.org/docs/`
- MDN Web Docs - JavaScript: `https://developer.mozilla.org/es/docs/Web/JavaScript`
- NodeJS Best Practices: `https://github.com/goldbergyoni/nodebestpractices`

¡Buena suerte en tu viaje de aprendizaje con NodeJS!