**Documentación**

Programa: **Simulación y procesamiento paralelo con Web Workers en JavaScript**  
Autor: **Beatriz Jiménez Jiménez**

**Índice de contenidos**

1. Documentación de usuario final  
   1.1. Guía de usuario  
   1.2. FAQ (Preguntas frecuentes)  
   1.3. Solución de problemas comunes
2. Documentación técnica para desarrolladores  
   2.1. Introducción al software  
   2.2. Arquitectura del sistema  
   2.3. Estructura del código  
   2.4. Especificaciones de API  
   2.5. Dependencias y configuraciones
3. Guías de instalación y despliegue  
   3.1. Requisitos previos  
   3.2. Instrucciones de instalación  
   3.3. Despliegue  
   3.4. Pruebas y verificación
4. Mantenimiento y soporte  
   4.1. Manejo de errores  
   4.2. Registro de cambios (changelog)  
   4.3. Soporte y contacto

**1. Documentación de usuario final**

**1.1. Guía de usuario**

El software contiene múltiples ejemplos prácticos que muestran cómo realizar operaciones matemáticas intensivas o cualquier tarea que requiera de gran procesamiento de datos y utilizar Web Workers para procesamiento paralelo.

**Requisitos del Sistema**

* **Navegador web**: Compatible con Chrome, Firefox, Safari o Edge.
* **Servidor web o localhost:** En este caso usamos Apache.
* **Conexión a internet**: Para cargar y guardar datos.

**Pasos básicos:**

1. Seleccione el archivo HTML del ejemplo que desea ejecutar.
2. Ábralo en un navegador web.
3. Observe los resultados en la consola del navegador.

**Orden recomendado de ejecución:**

* **Cálculo intensivo sin paralelismo:** 001-sin multiproceso.html
* **Medición de tiempo de cálculo:** 002-con date.html
* **Exploración del hardware:** 003-navigator.html
* **Introducción a Web Workers:** 004-worker.html y 005-worker.html
* **Procesamiento paralelo avanzado:** 006-worker.html, 007-worker.html, 008-worker.html, y 009-workersemaforo.html

**1.2. FAQ (Preguntas frecuentes)**

**P: ¿Qué navegador necesito?**  
R: Se requiere un navegador moderno con soporte para JavaScript ES6 y Web Workers, como Chrome, Firefox, Edge o Safari

**P: ¿Cómo sé si mi sistema soporta procesamiento paralelo?**  
R: El script detecta los núcleos disponibles mediante navigator.hardwareConcurrency y muestra la información en la consola.

**P: ¿Qué pasa si no encuentro los archivos .js asociados?**  
R: Asegúrese de que todos los archivos .html y .js estén en el mismo directorio.

**1.3. Solución de problemas comunes**

* **Capacidad de procesamiento muy lenta para cálculos avanzados en un solo hilo:**  
  **Causa:** El ordenador no tiene capacidad suficiente para hacer ese cálculo y mostrar el resultado  
  **Solución:** Reducir la cantidad de datos a procesar o cortar el proceso para evitar una sobre carga del pc
* **Error al crear un Worker:**  
  **Causa:** El archivo .js correspondiente no se encuentra.  
  **Solución:** Verifique que el archivo está en el mismo directorio que el .html.
* **Mensajes no se muestran en consola:**  
  **Causa:** Las líneas de console.log pueden estar comentadas.  
  **Solución:** Descomente las líneas de depuración para activar la salida en consola.

**2. Documentación técnica para desarrolladores**

**2.1. Introducción al software**

Este software es un conjunto de ejemplos educativos que muestran cómo realizar cálculos matemáticos intensivos en JavaScript y optimizar el rendimiento mediante el uso de Web Workers. Los ejemplos están diseñados para ilustrar conceptos clave relacionados con la ejecución en un solo hilo y en paralelo, destacando las diferencias en tiempos de ejecución y eficiencia al aprovechar múltiples núcleos del procesador.

Objetivos principales:

1. Demostrar el impacto de tareas intensivas en la ejecución del hilo principal:  
   Se muestra cómo un cálculo intensivo puede bloquear la interfaz de usuario cuando se ejecuta en el hilo principal.
2. Introducir el procesamiento paralelo con Web Workers:  
   Los ejemplos explican cómo los Workers permiten delegar tareas computacionalmente pesadas a hilos secundarios, evitando que la aplicación principal se detenga.
3. Medición de tiempos de ejecución:  
   Con herramientas simples como el objeto Date, el software mide el tiempo necesario para completar tareas intensivas.
4. Uso de múltiples Workers:  
   Se incluyen ejemplos avanzados que demuestran la creación y coordinación de varios Workers para dividir tareas intensivas entre núcleos del procesador, mejorando significativamente el rendimiento.

Funcionalidades principales:

* Cálculos intensivos: Ejemplo de un bucle que realiza millones de operaciones matemáticas repetitivas.
* Medición de rendimiento: Registro de tiempos de ejecución para evaluar la eficiencia de cada enfoque.
* Procesamiento paralelo: Implementación de Web Workers para manejar tareas en paralelo.
* Detección de hardware: Uso de navigator.hardwareConcurrency para identificar la cantidad de núcleos disponibles y optimizar la asignación de tareas.
* Coordinación avanzada: Uso de semáforos para sincronizar la finalización de múltiples Workers.

Casos de uso:

Este software es ideal para:

* Educación: Aprender los conceptos básicos y avanzados de procesamiento paralelo en aplicaciones web.
* Optimización: Comprender cómo dividir tareas en múltiples hilos para aplicaciones intensivas en cálculos.
* Demostraciones: Mostrar las capacidades de los Web Workers para evitar bloqueos en aplicaciones interactivas.

**2.2. Arquitectura del sistema**

El sistema incluye:

* Archivos HTML con scripts integrados para ejecutar cálculos y coordinar Workers.
* Archivos .js para la lógica de los Workers.
* Comunicación entre el hilo principal y los Workers mediante postMessage y onmessage.

**2.3. Estructura del código**

Copiar código

001-sin multiproceso.html

002-con date.html

003-navigator.html

004-worker.html

└─ 004-worker.js

005-worker.html

└─ 005-worker.js

006-worker.html

└─ 006-worker.js

007-worker.html

└─ 007-worker.js

008-worker.html

└─ 008-worker.js

009-workersemaforo.html

└─ 009-worker.js

**2.4. Especificaciones de API**

* **postMessage(data):** Envía datos del hilo principal al Worker o viceversa.
* **onmessage(event):** Maneja los datos recibidos en el Worker o hilo principal.
* **navigator.hardwareConcurrency:** Proporciona el número de núcleos del procesador.

**2.5. Dependencias y configuraciones**

* Navegador con soporte para ES6 y Web Workers.
* Archivos organizados en el mismo directorio para referencias correctas.

**3. Guías de instalación y despliegue**

**3.1. Requisitos previos**

* Navegador web.
* Acceso a un ordenador con múltiples núcleos para pruebas avanzadas.

**3.2. Instrucciones de instalación**

1. Descargue y descomprima los archivos en caso de que estén comprimidos
2. Asegúrese de que todos los archivos .html y .js estén en la misma carpeta, dentro de un servidor web o servidor local
3. Abra cualquier archivo .html en el navegador.

**3.3. Despliegue**

Si no tiene un servidor necesita hacer previamente esta instalación, puede ser un servidor web o localhost, por ejemplo usando Apache.

Los ejemplos se ejecutan directamente en el navegador.

**3.4. Pruebas y verificación**

* **Pruebas iniciales:** Ejecute 003-navigator.html para verificar los núcleos disponibles.
* **Pruebas de carga:** Use 007-worker.html o 009-workersemaforo.html para evaluar el rendimiento paralelo.

**4. Mantenimiento y soporte**

**4.1. Manejo de errores**

* **Problema:** Los mensajes no se muestran en consola.  
  **Solución:** Verifique que las líneas console.log estén activas.

**4.2. Registro de cambios (changelog)**

* **Versión 1.0:** Implementación de cálculos básicos y pruebas con Workers.
* **Versión 2.0:** Agregado de múltiples Workers y coordinación mediante semáforos.

**4.3. Soporte y contacto**

Para soporte técnico, contacte a: **[correo o enlace de contacto]**.