Evidencias modulos 13

```
JS main.js > ...
// Este archivo principal (main.js) demuestra el uso de módulos en JavaScript,
// incluyendo introducción, exportación/importación y carga dinámica.

// ======================================================================
// TEMA: 13.1 Módulos, introducción - 20/05/2024
// ======================================================================
// Este bloque introduce el concepto de módulos en JavaScript, que permiten
// dividir el código en archivos separados. Esto mejora la organización, la
// reutilización y ayuda a evitar colisiones de nombres de variables o funciones
// en el ámbito global.
//
// Aprendimos que para usar módulos, la etiqueta <script> en el HTML debe
// incluir el atributo `type="module"`. Esto activa el modo módulo, donde
// cada archivo es un módulo independiente con su propio ámbito (scope).
// Las variables y funciones declaradas dentro de un módulo no son accesibles
// desde fuera a menos que sean exportadas explícitamente.
//
// Para ver los resultados de este tema, se debe observar la consola del navegador (F12).

console.log("--- 13.1 Módulos, introducción ---");
console.log("Los módulos permiten organizar el código en archivos JavaScript separados.");
console.log("Cada módulo tiene su propio ámbito (scope) privado, evitando conflictos de nombres global
console.log("Las variables y funciones solo son accesibles desde fuera si son exportadas.");
console.log("Para que un script actúe como módulo, su etiqueta <script> en HTML debe tener `type='modu

// Ejemplo del ámbito de un módulo:
// Esta variable 'mensajeIntro' es local a este módulo (main.js).
// No se puede acceder a ella directamente desde otros módulos o scripts sin exportarla.
const mensajeIntro = "Esta variable 'mensajeIntro' es local al módulo main.js.";
console.log(mensajeIntro); // Se muestra porque estamos dentro del mismo módulo.

// Evidencia en el navegador:
// El mensaje "Los módulos permiten organizar..." y "Esta variable 'mensajeIntro'..."
// aparecerán en la consola del navegador.


// ======================================================================
```

```
// ======================================================================
// TEMA: 13.2 Exportación e importación - 20/05/2024
// ======================================================================
// Este bloque demuestra cómo compartir código entre módulos utilizando las
// palabras clave `export` e `import`.
//
// `export`: Se usa en un módulo para hacer que variables, funciones, clases, etc.,
// sean accesibles desde otros módulos. Puede haber exportaciones nombradas
// (varias por archivo) o una exportación por defecto (solo una por archivo).
//
// `import`: Se usa en un módulo para acceder a los elementos que han sido
// exportados por otro módulo.
//
// Para este ejemplo, estamos importando elementos desde `moduleA.js`.
//
// Para ver los resultados, observa la consola y el elemento `exportacion-importacion-output`
// en el HTML.

console.log("\n--- 13.2 Exportación e importación ---");

// ----------------------------------------------------------------------
// Importaciones nombradas:
// Se usan llaves {} para importar elementos específicos por su nombre exacto.
// Los nombres deben coincidir con los exportados en `moduleA.js`.
// ----------------------------------------------------------------------
import { nombreModuloA, saludarDesdeModuloA, contadorModuloA } from './moduleA.js';

// Usando la constante 'nombreModuloA' importada
console.log(`[Importado Nombrado] Constante: ${nombreModuloA}`);
document.getElementById('exportacion-importacion-output').innerHTML += `<p>Constante importada: ${nombre

// Usando la función 'saludarDesdeModuloA' importada
const saludoDesdeModulo = saludarDesdeModuloA("Invitado");
```

Lín. 158, col. 93 Espacios: 4 UTF-8 CRLF

```
const miProducto = new MiClaseProducto("Teclado Mecánico", 85);
console.log(`[Importado por Defecto] Clase: ${miProducto.getInfo()}`);
document.getElementById('exportacion-importacion-output').innerHTML += `<p>Clase importada (por de

// Evidencia en el navegador:
// Los mensajes de consola y los párrafos añadidos al div 'exportacion-importacion-output'
// mostrarán los valores y resultados de las importaciones.


// ======================================================================
// TEMA: 13.3 Importaciones dinámicas - 20/05/2024
// ======================================================================
// Este bloque explora las importaciones dinámicas, una característica avanzada
// que permite cargar módulos en tiempo de ejecución. Esto significa que un módulo
// no se carga ni se parsea hasta que realmente se necesita, lo que es útil
// para optimizar el rendimiento de la aplicación (code splitting, lazy loading).
//
// Aprendimos que `import()` (como una función, no la declaración de import al inicio)
// devuelve una Promesa. Esta Promesa se resuelve con un objeto módulo que contiene
// todas las exportaciones del módulo cargado.
// Se usa principalmente para funcionalidades condicionales o que no son críticas
// en la carga inicial.
//
// Para este ejemplo, un módulo (`dynamicModule.js`) se cargará solo cuando el
// usuario haga clic en un botón.
//
// Para ver los resultados, haz clic en el botón en el HTML y observa la consola
// y el elemento `importaciones-dinamicas-output`.

console.log("\n--- 13.3 Importaciones dinámicas ---");

const cargarModuloBtn = document.getElementById('cargarModuloBtn');
const dynamicOutput = document.getElementById('importaciones-dinamicas-output');

// Se añade un 'EventListener' al botón para manejar el clic del usuario.
cargarModuloBtn.addEventListener('click', async () => {
```

Lín. 158, col. 93 Espacios: 4 UT

```
main.js > ...
cargarModuloBtn.addEventListener('click', async () => {
        dynamicOutput.innerHTML = `Módulo dinámico cargado exitosamente!<br>`;
        dynamicOutput.innerHTML += `Mensaje desde módulo dinámico: <strong>${moduloDinamico.mensajeDinam
        dynamicOutput.innerHTML += `Resultado de sumar (5, 3) desde módulo dinámico: <strong>${moduloDin

        // Si el módulo dinámico tiene una exportación por defecto
        if (moduloDinamico.default) {
            const instanciaDinamica = new moduloDinamico.default("Gadget Secreto");
            dynamicOutput.innerHTML += `Instancia de clase dinámica: <strong>${instanciaDinamica.obtener
        }

        console.log("Módulo dinámico cargado y ejecutado. Contenido:", moduloDinamico);

    } catch (error) {
        // Si hay un error durante la carga del módulo (ej. archivo no encontrado, error de sintaxis)
        dynamicOutput.innerHTML = `<span style="color: red;">Error al cargar el módulo dinámico: ${error
        console.error("Hubo un error al cargar el módulo dinámico:", error);
    }
});

// Evidencia en el navegador:
// Inicialmente, no hay nada en el div 'importaciones-dinamicas-output'.
// Al hacer clic en el botón, el contenido del div se actualizará con los resultados
// de la carga y ejecución del módulo dinámico. También se mostrarán mensajes en la consola.
```