

# Comparación entre versiones no modulares y modulares

*Jazmin Viveros Sarmiento*

# 1. Introducción

Los módulos de JavaScript permiten separar la lógica del programa en archivos independientes, facilitando la organización, reutilización, y mantenimiento del código. Mientras que en versiones no modulares el JS suele estar incrustado directamente en el HTML, los módulos permiten mantener responsabilidades separadas.

## Comparación de Estructura:

### 1. Ubicación del Código JavaScript

- Versión No Modular:

En la versión no modular, el código JavaScript generalmente está directamente incrustado en el archivo HTML. Esto se hace mediante la etiqueta `<script>`, que puede ser colocada en la cabecera (`<head>`) o al final del cuerpo (`<body>`) del documento HTML. La ventaja de esta técnica es que es sencilla y directa, pero puede volverse difícil de gestionar en proyectos grandes.

- Versión Modular:

En la versión modular, el código JavaScript se separa en archivos independientes, que son cargados en el HTML mediante la etiqueta `<script>` con el atributo `type="module"`. Este enfoque permite organizar el código en diferentes módulos que se pueden importar y exportar.

Cada archivo JS puede tener su propio conjunto de funcionalidades, y puede ser importado en otros archivos JS para reutilización de funciones y lógica. Este enfoque es más adecuado para proyectos grandes o escalables.



## 2. Declaración de Scripts

- Versión No Modular:
  - Los scripts se incluyen sin especificar que son módulos. Se usan etiquetas `<script>` normales que permiten ejecutar el código directamente en la página.
  - Esta forma no permite la modularización del código ni la reutilización de funciones entre distintos archivos.
- Versión Modular:
  - Los scripts deben ser declarados con el atributo `type="module"`, lo que indica que este código está estructurado como un módulo y puede importar y exportar funcionalidades. Esto también habilita características como la carga diferida de los módulos.
  - La importación y exportación de módulos facilita la organización y reutilización del código.

## 3. Reutilización de Funciones

- Versión No Modular:
  - En la versión no modular, la reutilización de funciones dentro de un proyecto es limitada. Si necesitas utilizar una función que has definido en un archivo, tienes que copiar y pegar el código en otros lugares, lo que incrementa el riesgo de errores y hace más difícil mantener el proyecto a medida que crece.
  - Además, la carga de un archivo JavaScript no es controlada de manera eficiente, lo que puede llevar a problemas de rendimiento.
- Versión Modular:
  - La principal ventaja de usar módulos es la facilidad para reutilizar funciones y lógica entre diferentes archivos de forma eficiente. Usando `export` e `import`, puedes definir funciones y luego importarlas en otros archivos JS.
  - Esto permite que el código sea más limpio, más fácil de mantener y mucho más organizado. Puedes reutilizar partes del código sin tener que duplicarlo.

#### 4. Tiempo de Carga del Navegador

- Versión No Modular:
  - La carga del código JavaScript en una versión no modular es generalmente más rápida en comparación con la modular, ya que el código se carga directamente desde el HTML sin la necesidad de buscar archivos adicionales. Sin embargo, esto puede volverse un problema cuando el proyecto se vuelve más grande.
  - La falta de organización puede hacer que el código se cargue todo junto, lo que podría afectar la eficiencia y el tiempo de carga cuando el archivo JavaScript crece significativamente.

#### 5. Escalabilidad


- Versión No Modular:
  - En proyectos pequeños, la versión no modular puede ser adecuada, ya que no es necesario organizar el código de manera compleja. Sin embargo, a medida que el proyecto crece, la estructura plana del archivo HTML y el código JS hace que se vuelva más difícil de mantener y escalar.
  - A medida que el número de funcionalidades aumenta, la cantidad de código en un solo archivo puede dificultar la comprensión y depuración.
- Versión Modular:
  - La modularización del código permite que el proyecto sea altamente escalable. Los archivos JS pueden dividirse en funciones más pequeñas y reutilizables, y el uso de módulos permite que el código se divida en piezas más manejables.
  - Esto facilita agregar nuevas funcionalidades sin sobrecargar el archivo principal, y permite una mejor organización y una estructura más clara, lo que es crucial para proyectos grandes y complejos.



## 6. Mantenimiento y Depuración

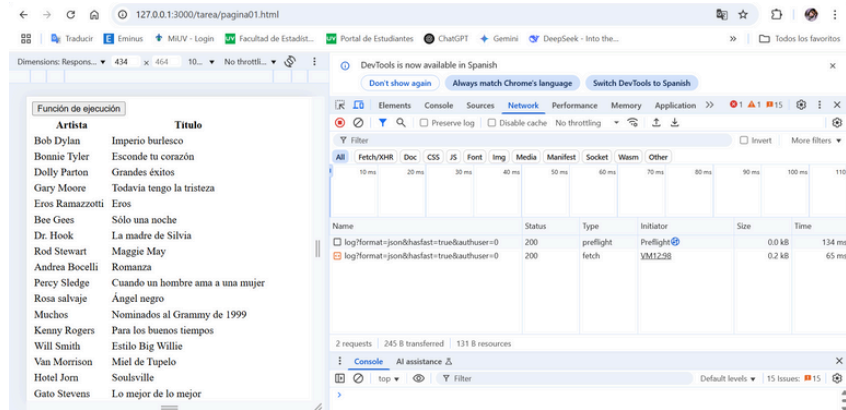
- Versión No Modular:
  - Dado que el código está todo incluido en un solo archivo, la depuración y mantenimiento de un proyecto grande se vuelve más difícil. Los errores no siempre son claros, y cualquier cambio realizado en un área puede afectar a muchas otras partes del código.
  - Las versiones no modulares también tienen dificultades para mantener una buena separación de responsabilidades entre diferentes partes del código.
- Versión Modular:
  - La principal ventaja de usar módulos es que cada archivo puede tener una responsabilidad clara y definida. Esto mejora enormemente la capacidad para debuggear y encontrar errores, ya que el código está más organizado y segmentado.
  - El mantenimiento también es más sencillo, ya que puedes modificar un módulo sin afectar al resto del código, siempre que mantengas las interfaces (importaciones y exportaciones) consistentes.

## 7. Diferencias en ejecución

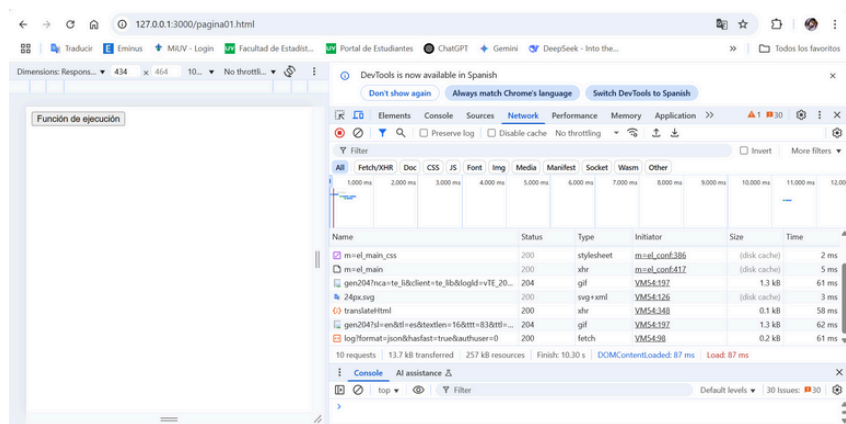
- Tiempo de carga inicial: En módulos puede haber un micro retraso por la lectura externa, pero el navegador puede cachearlo eficientemente.
  - Lectura del DOM: En módulos es más común usar `DOMContentLoaded` para asegurar que el DOM esté disponible.
  - Debugging y mantenimiento: En módulos es más sencillo ya que se pueden ver errores por archivo específico.
- 

# Capturas de pantalla

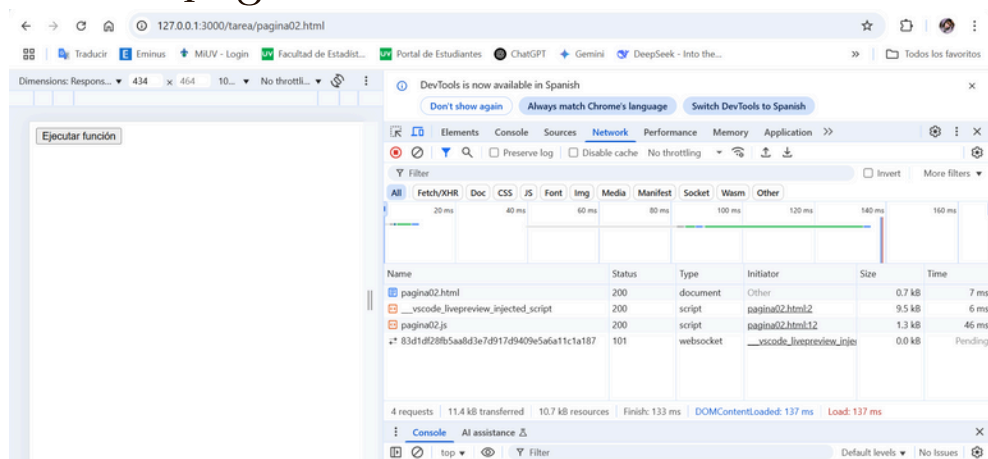
## Modular pagina01:



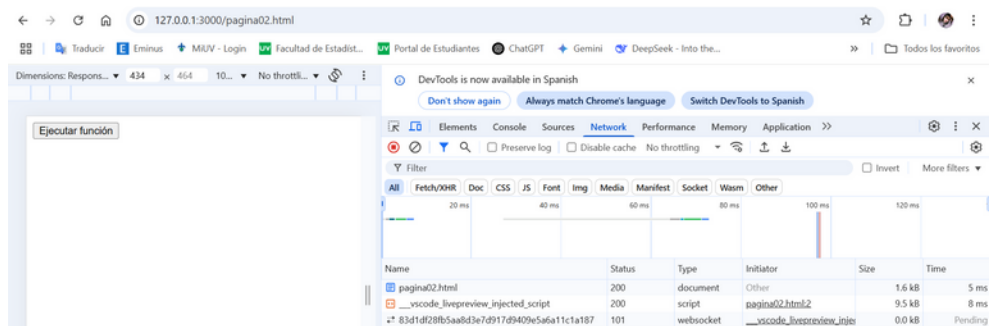
## No modular:



## Modular pagina02:



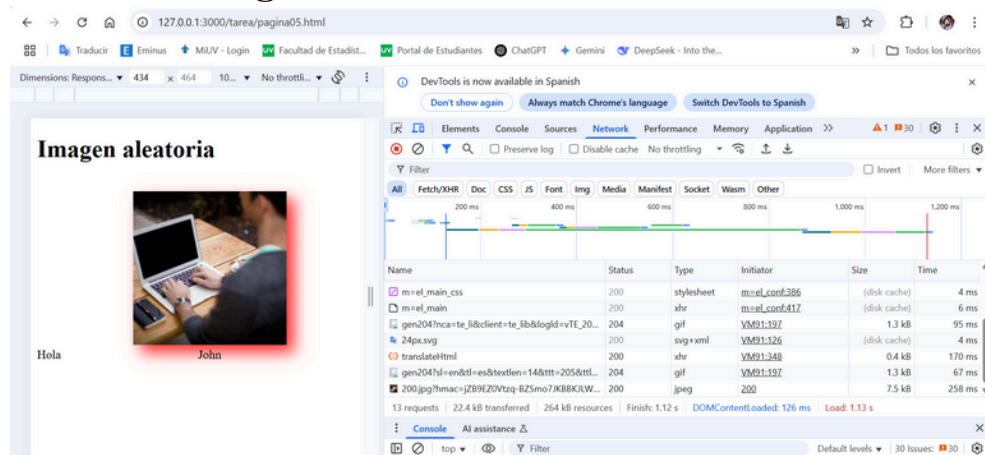
## No modular:



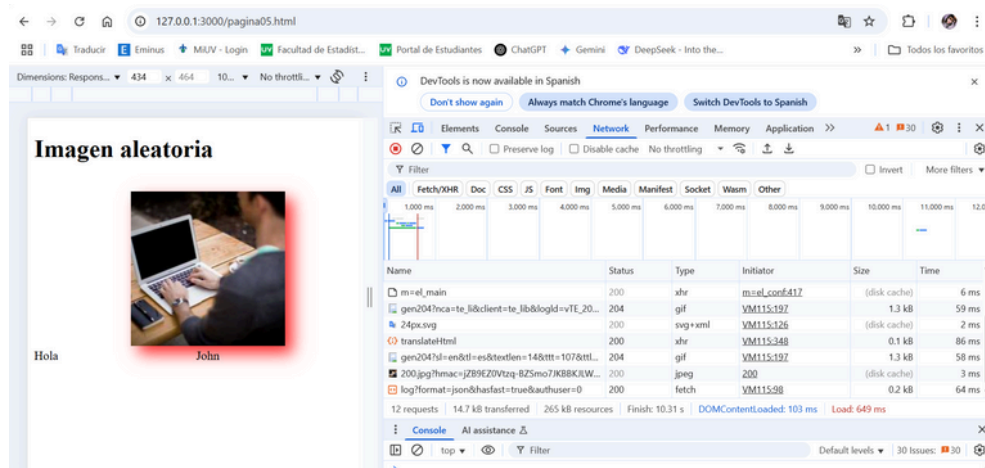


# Capturas de pantalla

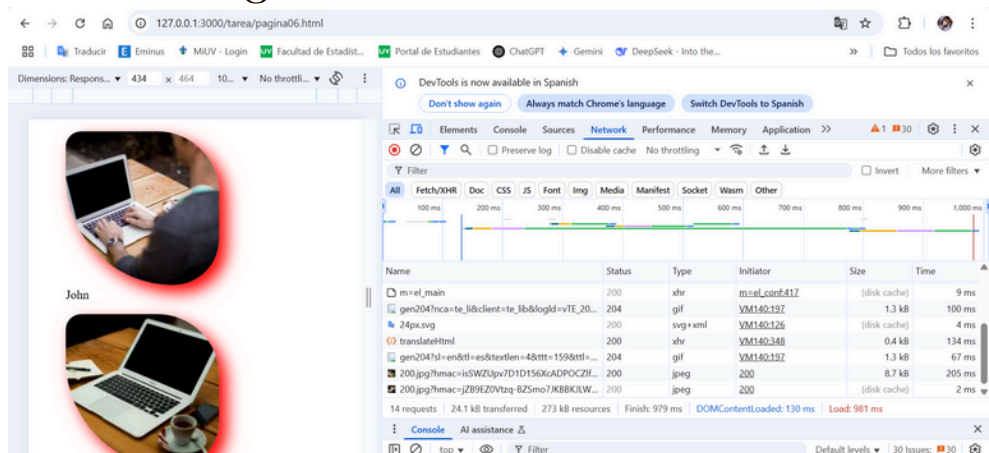
## Modular pagina05:



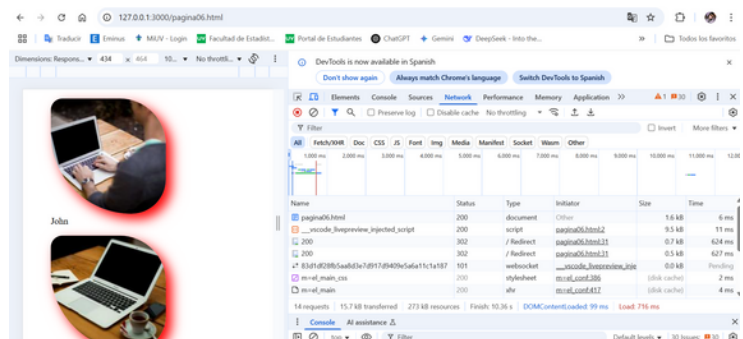
## No modular:



## Modular pagina06:

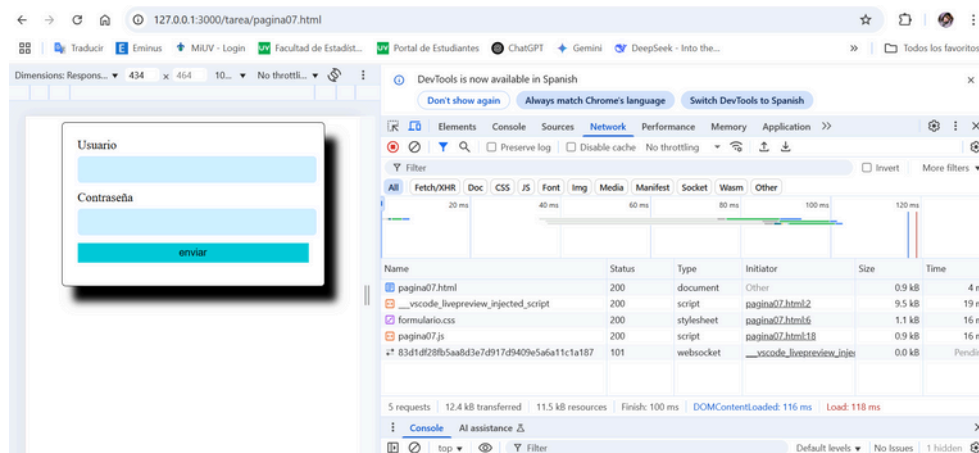


## No modular:

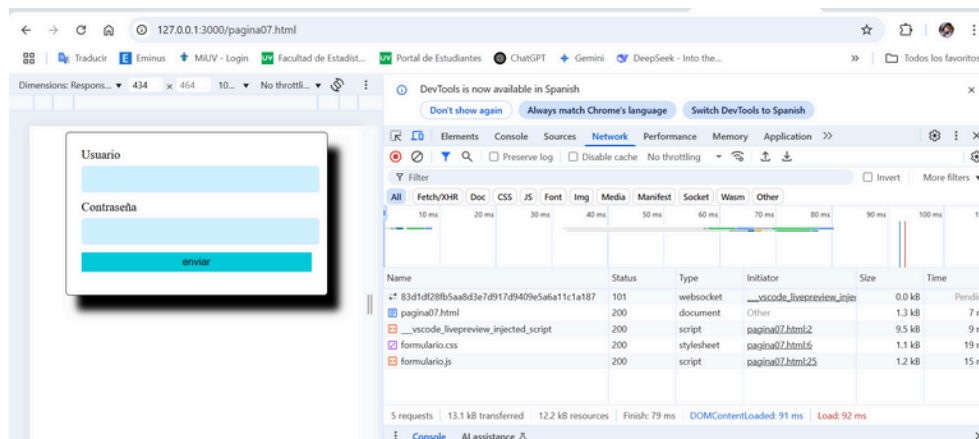


# Capturas de pantalla

## Modular pagina07:



## No modular:



## Conclusion:

El uso de módulos en JavaScript es esencial para mantener proyectos organizados, especialmente a medida que escalan. Aunque la diferencia en tiempo de carga puede ser mínima, las ventajas a nivel de mantenimiento, claridad y reusabilidad son significativas.