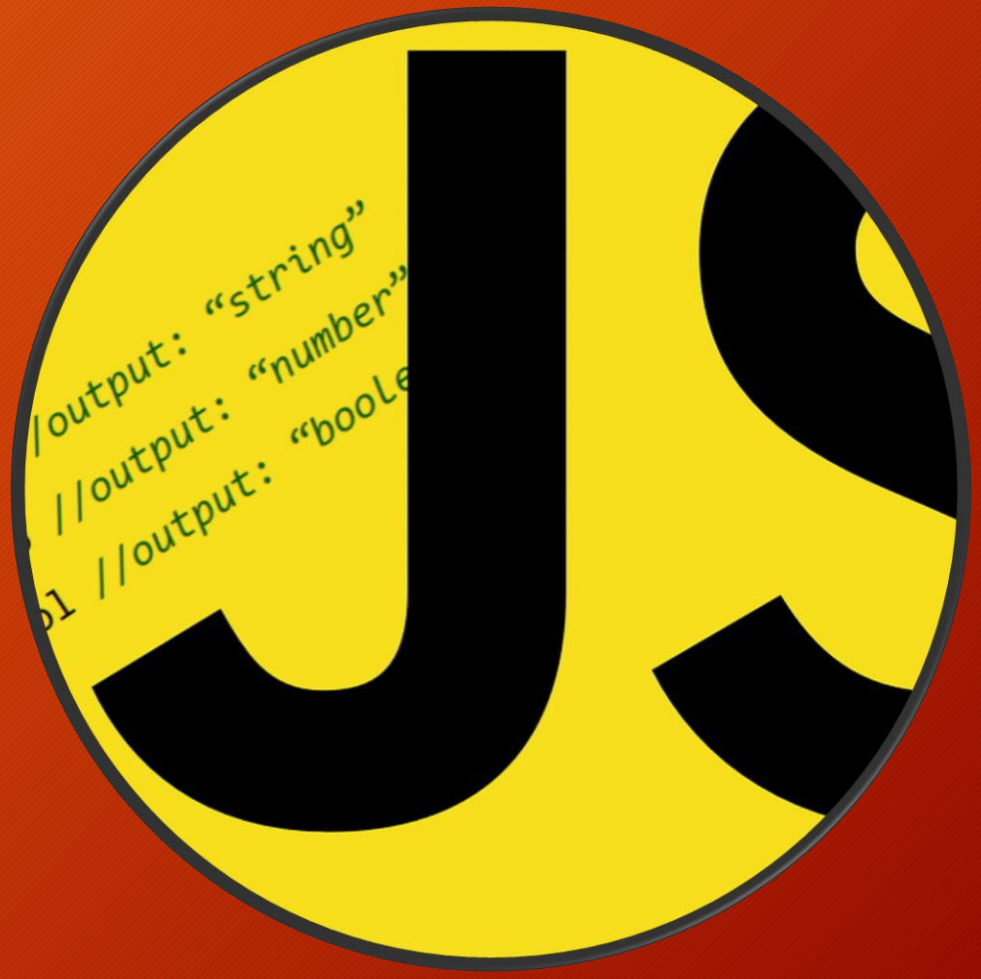
Estrategias de Escalabilidad en JavaScript.

Jorge Luis Lugo González LG242867
Gabriel Mario Hernández Rosales HR242882

Agenda

- Uso de eventos y asincronismo en JavaScript
- Diferencias entre ejecución síncrona y asíncrona
- Impacto del rendimiento en aplicaciones web dinámicas
- Bibliografía

Uso de eventos y asincronismo en JavaScript



Definición de eventos en JavaScript y su manejo mediante "event listeners"

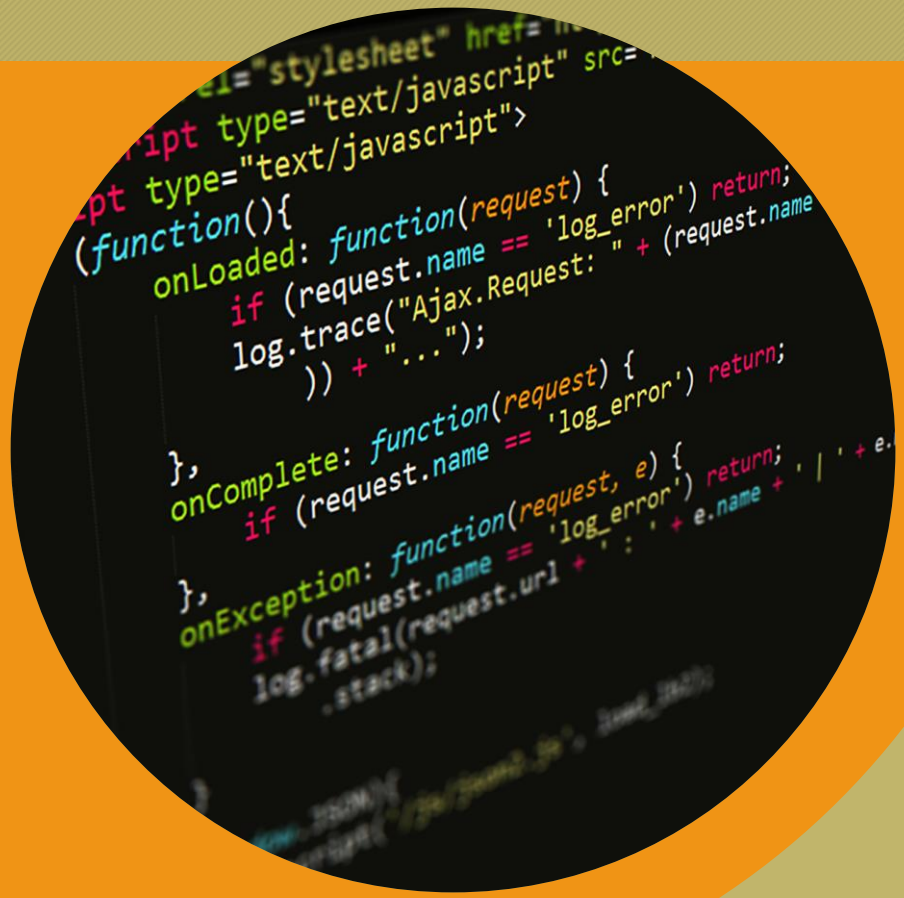
- Los eventos en JavaScript representan acciones o sucesos que ocurren en el navegador, como clics del usuario, carga de páginas o interacciones con formularios. Estos eventos se gestionan mediante "event listeners", funciones que se ejecutan en respuesta a un evento específico.

Técnicas de asincronismo:

- ***Callbacks*:** Funciones que se pasan como argumentos a otras funciones y se ejecutan después de que la operación principal ha concluido.
- ***Promises*:** Objetos que representan la eventual finalización (o falla) de una operación asíncrona y su valor resultante.
- ***Async/Await*:** Sintaxis que permite escribir código asíncrono de manera más legible y estructurada, facilitando el manejo de promesas.

Ejemplos y ventajas para la escalabilidad

Implementar estas técnicas permite que las aplicaciones manejen múltiples operaciones simultáneamente sin bloquear el hilo principal, mejorando la capacidad de respuesta y escalabilidad.



Diferencias entre ejecución síncrona y asíncrona

Navegar por las sesiones de Preguntas y respuestas

- **Ejecución síncrona**: Las operaciones se realizan de manera secuencial; cada instrucción espera a que la anterior termine antes de ejecutarse. Esto puede llevar a bloqueos si una operación tarda demasiado en completarse.
- **Ejecución asíncrona**: Permite que el programa continúe ejecutando otras tareas mientras una operación de larga duración se procesa en segundo plano, evitando bloqueos y mejorando la eficiencia.

Comparativa con ejemplos de código: La ejecución síncrona puede ilustrarse con funciones que bloquean el flujo hasta completarse, mientras que la asíncrona utiliza callbacks, promesas o async/await para manejar operaciones no bloqueantes.

Impacto en el rendimiento de aplicaciones web dinámicas

Los puntos que generan dicho impacto son

- **Requerimientos de rendimiento en aplicaciones modernas:** Las aplicaciones web actuales demandan respuestas rápidas y eficientes, especialmente cuando manejan datos dinámicos y en tiempo real. Mejora de la escalabilidad.
- **tiempos de respuesta:** El uso de técnicas asíncronas permite que las aplicaciones procesen múltiples solicitudes concurrentemente, reduciendo tiempos de espera y mejorando la experiencia del usuario.
- **Buenas prácticas para optimización y prevención de cuellos de botella:** Implementar correctamente el manejo de eventos y operaciones asíncronas, junto con la optimización de recursos y la gestión adecuada de memoria, es crucial para mantener un rendimiento óptimo.

Bibliografía

- *E. A. Smith, JavaScript: The Definitive Guide, 7th ed., O'Reilly Media, 2020.*
- *D. Flanagan, JavaScript: The Good Parts, O'Reilly Media, 2008.*
- *M. Cantelon, M. Harter, T. Holowaychuk, y N. Rajlich, Node.js in Action, Manning Publications, 2014.*

Fuentes de la UDB.

- *https://rd.udb.edu.sv/items/b2b8e8f9-3d15-4b5e-9b23-4f23efb2c559*

Gracias