**Trabajo Practico N°1**

**Actividad**

Responder las siguientes preguntas sobre node

1. ¿Qué es Node.js y en qué se diferencia de JavaScript en el navegador?
2. ¿Qué son las versiones LTS?
3. Explica la diferencia entre un proceso síncrono y uno asíncrono
4. ¿Cómo se maneja la asincronía en Node.js? Explica las diferencias entre callbacks, Promises y async/await.
5. ¿Qué es el objeto global en Node.js y en qué se diferencia de window en el navegador?
6. ¿Node.js es multithreading?
7. ¿Qué es el Event Loop en Node.js y cuál es su papel en la ejecución de código asíncrono?
8. Mencionar tres ventajas de Node.js
9. menciona tres desventajas de Node.js.
10. ¿Podrías nombrar algunas bibliotecas que los desarrolladores utilizan frecuentemente con Node.js?

**Respuesta**

1. Node.js es un entorno de ejecución de JavaScript del lado del servidor. A diferencia de JavaScript en el navegador, que se ejecuta en un entorno cliente, Node.js utiliza el motor V8 de Google, lo cual lo hace más rápido y eficiente. Este entorno permite la ejecución de servicios asíncronos y es ideal para desarrollar aplicaciones en tiempo real, como chats o servidores que manejan múltiples conexiones simultáneas de manera escalable.
2. Las versiones LTS (Long-Term Support) de Node.js son versiones con soporte a largo plazo que reciben actualizaciones críticas de seguridad, corrección de errores y mejoras menores durante un período extendido, generalmente de 30 meses. Estas versiones están diseñadas para ser estables y confiables, por lo que son recomendadas para proyectos en producción. Las versiones que no son LTS suelen recibir soporte por un tiempo más corto y se utilizan principalmente para experimentar con nuevas características.
3. Un proceso síncrono ejecuta tareas de forma secuencial, bloqueando el flujo hasta que cada operación termina. En cambio, un proceso asíncrono permite que tareas que toman tiempo se ejecuten en segundo plano, mejorando el rendimiento al no bloquear otras operaciones.
4. Node.js maneja la asincronía con callbacks, Promises y async/await.

* Callbacks: Funciones que se ejecutan al completar una tarea asíncrona. Pueden generar callback hell si se anidan demasiado.
* Promises: Mejoran la legibilidad del código al manejar asincronía con métodos como .then() y .catch().
* async/await: Sintaxis moderna que permite escribir código asíncrono como si fuera síncrono, haciendo que sea más claro y fácil de leer.

1. El objeto global en Node.js es global, mientras que en el navegador es window. A diferencia de window, global no se usa comúnmente para definir variables globales. Además, Node.js proporciona otros objetos globales como process y module, que no existen en el navegador.
2. No, Node.js no es multithreading de forma nativa. Usa un solo hilo para ejecutar JavaScript, pero puede manejar operaciones asíncronas y tareas en segundo plano mediante su modelo de I/O no bloqueante y el Event Loop.
3. El Event Loop en Node.js es el mecanismo que permite manejar operaciones asíncronas. Mantiene un solo hilo de ejecución y se encarga de procesar las tareas pendientes (como callbacks y promesas) en un ciclo continuo, lo que evita el bloqueo del hilo principal y mejora el rendimiento.

* Alto rendimiento gracias al motor V8 de Google.
* Modelo de I/O no bloqueante, ideal para aplicaciones en tiempo real.
* Gran ecosistema de bibliotecas y paquetes disponibles a través de npm.
* No es ideal para aplicaciones con mucho procesamiento en CPU.
* La asincronía puede complicar la estructura del código.
* No tiene un buen manejo de operaciones bloqueantes, lo que puede afectar el rendimiento en algunos casos.
* Express: Framework minimalista para crear aplicaciones web.
* Axios: Cliente HTTP para hacer solicitudes.
* Mongoose: Biblioteca para trabajar con MongoDB.