Node JS.1 Teoria y Hola Mundo

Nombre	Ismael	Curso	
Apellidos	Gonzalez Tempa	Fecha	

Preguntas de Teoria NODE Js

1. ¿Cuál es el propósito principal de Node.js?

Permitir la creación de aplicaciones de red altamente escalables y eficientes, especialmente aquellas que manejan muchas conexiones simultáneas.

2. ¿En qué año fue creado Node.js y quién fue su creador?

Fue creado en **2009** por **Ryan Dahl**.

3. ¿Qué empresa apadrina la evolución de Node.js y tiene a su creador en plantilla?

La empresa es **Joyent**, la cual contrató a Ryan Dahl para liderar el proyecto. Actualmente, el proyecto es gestionado por la OpenJS Foundation.

4. ¿Cuál es el modelo de evaluación de ejecución de Node.js?

Se basa en un modelo de **E/S (entrada/salida) sin bloqueo** y orientado a eventos.

5. ¿Por qué Node.js se diseñó con un enfoque de único hilo de ejecución?

Para evitar el alto consumo de memoria y recursos que implica crear un hilo nuevo por cada conexión, lo que permite manejar miles de conexiones concurrentes con muy poca carga de sistema.

6. ¿Qué inconveniente tiene el enfoque de único hilo de ejecución en Node.js y cómo se puede abordar?

El mayor inconveniente es que las tareas pesadas de CPU pueden bloquear el hilo principal. Se aborda mediante el uso de **procesos hijos (child processes)** o, en versiones modernas, con **Worker Threads**.

7. ¿Cuál es el motor de ejecución utilizado por Node.js y quién lo desarrolló originalmente?

Utiliza el motor **V8**, desarrollado originalmente por **Google** para el navegador Chrome.

8. ¿Qué es V8 y cuál es su relación con Node.js?

Es un motor de código abierto que compila JavaScript directamente a código de máquina. En Node.js, es el componente que interpreta y ejecuta el código JavaScript.

9. ¿Cómo maneja Node.js eventos asíncronos y qué librería utiliza para ello?

Los maneja mediante una cola de eventos y un bucle de eventos (event loop). Utiliza la librería **libuv** para gestionar esta asincronía.

10. ¿Cuál es el propósito de libuv en el contexto de Node.js?

Es una librería escrita en C que proporciona soporte para operaciones asíncronas basadas en el bucle de eventos y el manejo de hilos en segundo plano para el sistema de archivos, DNS, etc.

11. ¿Qué tipos de módulos básicos incorpora Node.js en su propio binario?

Incorpora módulos de bajo nivel como `http`, `https`, `fs` (file system), `path`, `os`, `crypto` y `stream`.

12. ¿Cómo se pueden utilizar módulos desarrollados por terceros en Node.js?

Se pueden utilizar mediante la instrucción `require()` (CommonJS) o `import` (ES Modules), tras haber sido instalados en el proyecto.

13. ¿Qué función cumple el Node Package Manager (npm) en la gestión de módulos en Node.js?

Es el gestor de paquetes que facilita la descarga, instalación, actualización y gestión de dependencias de librerías creadas por la comunidad.

14. ¿Qué posibilidades ofrece Node.js en términos de desarrollo homogéneo entre cliente y servidor?

Ofrece la posibilidad de utilizar **JavaScript tanto en el frontend como en el backend**, permitiendo compartir código, validaciones y lógica de negocio entre ambos.

15. ¿Cómo se registra Node.js con el sistema operativo y qué sucede cada vez que un cliente establece una conexión?

Node.js le indica al SO que quiere escuchar en un puerto específico. Cada vez que un cliente se conecta, se emite un evento que se añade a la cola para ser procesado sin detener la escucha de nuevas conexiones.

16. ¿Cómo se activa el bucle de gestión de eventos en Node.js y cuándo se termina?

Se activa automáticamente al ejecutar el comando `node` con un archivo. Se termina cuando ya no quedan más callbacks por ejecutar o eventos programados.

17. ¿Qué importancia tiene el bucle de eventos en Node.js y cómo difiere de otros servidores dirigidos por eventos mencionados?

Es el corazón de Node.js. A diferencia de otros servidores que crean hilos por conexión, Node.js utiliza el bucle para despachar todas las tareas, lo que lo hace mucho más ligero.

18. ¿Qué significa que Node.js es asíncrono y cómo se refleja en su arquitectura orientada a eventos?

Significa que las operaciones de I/O no detienen la ejecución del programa; el programa continúa y, cuando la operación termina, se ejecuta una función de respuesta (callback).

19. ¿Cuáles son algunos ejemplos de otros entornos de ejecución similares a Node.js mencionados en el texto?

Ejemplos comunes incluyen **Deno** y **Bun** (entornos modernos creados para mejorar aspectos de Node.js), o implementaciones como EventMachine (Ruby) y Twisted (Python).

20. ¿Qué es el entorno REPL en Node.js y para qué se utiliza?

Significa *Read-Eval-Print-Loop*. Es una consola interactiva donde puedes escribir código JavaScript y obtener resultados inmediatos; se usa para pruebas rápidas y experimentación.

21. ¿Cómo aborda Node.js la concurrencia y por qué se destaca su capacidad para manejar operaciones de entrada/salida concurrentes?

No usa hilos del SO para cada tarea, sino que delega las operaciones de I/O al sistema y continúa procesando otras tareas, destacando por su bajísimo consumo de memoria en tareas masivas.

22. ¿Cuál es el papel de V8 en la ejecución de JavaScript en Node.js?

Transforma el código JavaScript de alto nivel en instrucciones optimizadas que el procesador de la computadora puede entender rápidamente.

23. ¿Cómo se compone el cuerpo de operaciones de base de Node.js y en qué lenguajes está escrito?

Se compone de una API de JavaScript que interactúa con un núcleo de bajo nivel escrito principalmente en **C++** y **C**.

24. ¿Cuáles son algunos ejemplos de módulos básicos incorporados en el binario de Node.js?

`fs` para archivos, `http` para servidores web, `events` para manejar emisores de eventos y `util` para funciones de utilidad.

25. ¿Cómo facilita Node.js el desarrollo homogéneo entre el cliente y el servidor, según el texto?

Al usar el mismo lenguaje y estructuras de datos (JSON) en ambos lados, se elimina la "barrera mental" y técnica de cambiar de lenguaje entre el navegador y el servidor.