Eros Ruiz Ramirez

1. **Explica brevemente qué es Node.js y en qué se diferencia de otras tecnologías de servidor.**
   Node.js es un entorno de JavaScript del lado del servidor que permite ejecutar código JavaScript en el servidor en lugar de en el navegador, es muy eficaz en el funcionamiento asíncrono orientado a eventos.
   A diferencia de otras tecnologías de servidor, Node.js está diseñado para ser altamente eficiente y escalable, esto se debe a que utiliza un enfoque especial que no bloquea el servidor mientras espera que algunas tareas se completen, puede hacer varias cosas al mismo tiempo sin ralentizarse.
2. **Menciona tres ventajas principales de usar Node.js en comparación con otros entornos de servidor.**
   1. Eficiencia y rendimiento: Node.js utiliza un modelo de E/S no bloqueante que permite manejar un gran número de conexiones simultáneas de manera eficiente, lo que lo hace ideal para aplicaciones en tiempo real y con alta concurrencia.
   2. Compartir código entre el lado del cliente y el servidor: Al usar JavaScript tanto en el lado del cliente como en el servidor, se puede compartir lógica de programación y validaciones de datos, lo que facilita el desarrollo y el mantenimiento.
   3. Amplio ecosistema de paquetes: Node.js cuenta con un administrador de paquetes llamado npm que proporciona acceso a una amplia variedad de módulos y bibliotecas de código abierto que facilitan el desarrollo de aplicaciones.
3. **¿Qué es npm? ¿Para qué se utiliza en el desarrollo de aplicaciones Node.js?**
   npm (Node Package Manager) es el administrador de paquetes predeterminado para Node.js. Se utiliza en el desarrollo de aplicaciones Node.js para buscar, instalar y administrar módulos y bibliotecas de terceros. npm simplifica la gestión de dependencias, lo que facilita la integración de funcionalidades adicionales en una aplicación Node.js y el trabajo en proyectos colaborativos.
4. **Describe el concepto de "callback" en Node.js y su relevancia en la programación asincrónica.**
   Es una función que se pasa como argumento a otra función y se ejecuta después de que se complete una operación asincrónica, como una lectura de archivo o una solicitud HTTP.
   Los callbacks son esenciales para la programación asincrónica en Node.js, ya que permiten que el código continúe ejecutándose mientras se espera que se complete una operación de E/S. Esto evita bloquear el hilo principal de ejecución y mejora la capacidad de respuesta de las aplicaciones en situaciones de alta concurrencia.
5. **¿Qué son los módulos en Node.js? Proporciona un ejemplo de cómo puedes importar y utilizar un módulo en una aplicación.**
   Son unidades de código reutilizables que encapsulan funcionalidades específicas.
   Para importar y utilizar un módulo en una aplicación se debe usar la función “require”, en este caso voy a poner de ejemplo el módulo “fs” que sirve para trabajar con archivos.

```
const fs = require("fs")

fs.readFile("datos.txt", "utf8", (error, datos) => {
   if (error) {
      console.error(`Error al leer el archivo: ${error}`)
   } else {
      console.log(`Contenido del archivo: ${datos} `)
   }
})
```