### **Parte 1: Perguntas Teóricas**

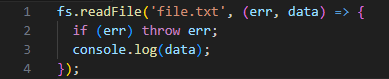
**Node.js:**

**Explique o que é o Event Loop no Node.js e como ele funciona.**

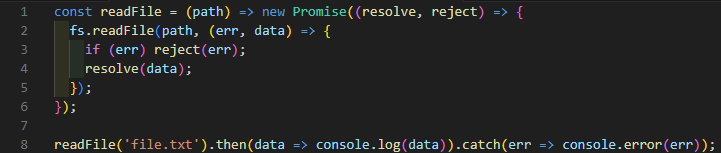
O Event Loop é um dos componentes principais do Node.js que permite a execução não bloqueante de operações de I/O. Ele monitora a fila de tarefas e executa callbacks ou funções de forma assíncrona. O Event Loop segue um ciclo contínuo de verificação de tarefas pendentes, execução de callbacks prontos, e repetição do ciclo.

**Qual a diferença entre callbacks, Promises e async/await no Node.js?**

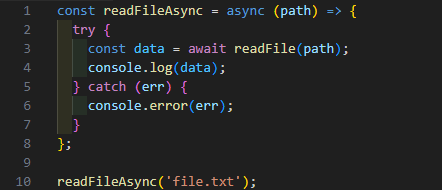
Callbacks: Funções passadas como argumento para outra função e executadas após a conclusão de uma operação assíncrona.



Promises: Objetos que representam a eventual conclusão ou falha de uma operação assíncrona.

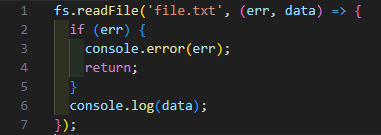


async/await: Sintaxe para escrever código assíncrono de forma mais síncrona e legível.



**Como você lidaria com erros no Node.js? Dê exemplos de como tratar erros de maneira eficaz.**

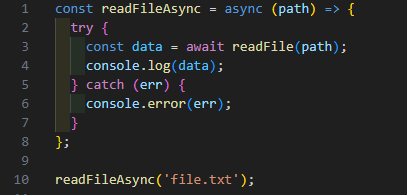
Callbacks: Passar o erro como primeiro argumento no callback.



Promises: Usar .catch() para capturar erros.

****

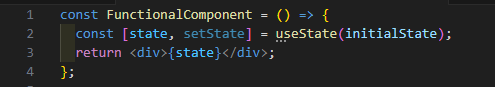
async/await: Usar try/catch para capturar erros.



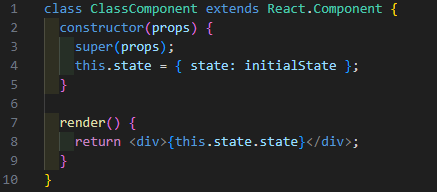
**React.js:**

**Qual a diferença entre componentes funcionais e componentes de classe no React?**

Componentes funcionais: São funções JavaScript que retornam elementos React. Usam hooks para gerenciar estado e ciclo de vida.

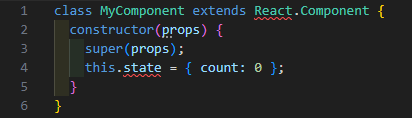
****

Componentes de classe: São classes ES6 que estendem React.compoment. Possuem métodos de ciclo de vida e this.state para gerenciar estado.

****

**Explique o conceito de "state" e "props" no React. Como eles diferem?**

State: Representa dados internos que um componente controla e pode alterar



Props: Representa dados passados para um componente, que são imutáveis no contexto do componente.



**O que são hooks no React? Dê exemplos de alguns hooks mais usados e suas finalidades.**

useState: Gerencia estado em componentes funcionais

****

useEffect: Efeitos colaterais, como fetching de dados ou manipulação de DOM

****

useContext: Consome contexto sem precisar de um componente consumidor.

****

**Bancos de Dados:**

**Qual a diferença entre bancos de dados relacionais e não relacionais? Dê exemplos de quando usar cada um.**

Relacionais: Organizam dados em tabelas com relações definidas. Usam SQL. Ex: MySQL, PostgreSQL.

* Quando usar: Dados estruturados, integridade referencial.

Não relacionais: Armazenam dados em formatos flexíveis como documentos, grafos, ou pares chave-valor. Ex: MongoDB, Redis.

* Quando usar: Dados não estruturados, escalabilidade horizontal.

**Explique o conceito de "normalização" em bancos de dados relacionais.**

Processo de organizar dados para reduzir redundância e melhorar a integridade. Divide grandes tabelas em tabelas menores e define relacionamentos.

* Exemplo:
  + 1NF: Eliminar grupos repetitivos.
  + 2NF: Remover dependências parciais.
  + 3NF: Remover dependências transitivas.

**Como você lidaria com uma situação em que precisa escalar horizontalmente um banco de dados?**

Distribuir carga de um banco de dados em múltiplos servidores.

* Técnicas:
  + Sharding: Dividir dados em pedaços menores distribuídos em diferentes servidores.
  + Replication: Copiar dados para múltiplos servidores para redundância e balanceamento de carga.

**Git:**

**Explique o que é o Git e por que ele é usado.**

Git é um sistema de controle de versão distribuído que permite a rastreabilidade de alterações no código, colaboração em equipe e gerenciamento de diferentes versões de um projeto.

**O que é um "branch" no Git? Como você gerencia os branches em um projeto grande?**

Um branch é uma linha paralela de desenvolvimento que permite trabalhar em funcionalidades separadamente.

* Gerenciamento de branches:
  + Criar: git branch feature-branch
  + Mudar: git checkout feature-branch
  + Merge: git merge feature-branch
  + Deletar: git branch -d feature-branch

**O que são "merge conflicts" e como você resolve um conflito de merge no Git?**

Conflitos de merge ocorrem quando há alterações conflitantes no mesmo trecho de código em diferentes branches.

* Resolução:
  + Manual: Editar os arquivos conflitantes e remover marcas de conflito.
  + Ferramentas: Usar ferramentas de merge como KDiff3, Meld.

