1. O que é Node.js e por que é popular no desenvolvimento web?

Node.js é um ambiente de execução de JavaScript no lado do servidor. Ele permite que desenvolvedores usem JavaScript para construir aplicações back-end. É popular porque é rápido, assíncrono, baseado em eventos e utiliza a mesma linguagem no front e no back (JavaScript), facilitando o desenvolvimento full stack.

2. Qual é a diferença entre o Node.js e outras tecnologias de servidor, como o Apache?

Node.js é baseado em eventos e orientado a não bloquear processos (event-driven e non-blocking I/O), enquanto o Apache trabalha de forma tradicional, criando uma nova thread para cada requisição. Isso faz o Node.js ser mais leve e eficiente em aplicações em tempo real e com muitas conexões simultâneas.

3. Como você inicia um projeto Node.js usando o npm?

Basta abrir o terminal na pasta do projeto e digitar: npm init -y

4. O que é o Express.js e qual é o seu papel no desenvolvimento web com Node.js?

Express.js é um framework web minimalista e flexível para Node.js. Ele facilita a criação de servidores HTTP e APIs, fornecendo ferramentas para gerenciar rotas, requisições, respostas, middlewares e muito mais.

5. Explique o conceito de middleware no contexto do Express.js.

Middlewares são funções executadas durante o ciclo de requisição/resposta de um servidor Express. Eles podem:

- Modificar a requisição ou resposta
- Finalizar a requisição
- Passar o controle para o próximo middleware

Exemplo: `app.use(express.json())` é um middleware que interpreta o corpo JSON de uma requisição.

6. Como você roteia solicitações HTTP em um aplicativo Express?

Usando métodos como `app.get()`, `app.post()`, `app.put()`, `app.delete()`, onde o primeiro parâmetro é a rota e o segundo é a função que trata a requisição.

Exemplo:

```
app.get('/games', (req, res) => {
  res.json(listaGames);
});
```

7. O que é o middleware de análise de corpo (body-parser) e por que é útil em um aplicativo Express?

O `body-parser` (ou o próprio `express.json()`) interpreta o corpo das requisições HTTP (geralmente em JSON) e disponibiliza os dados no `req.body`. É essencial para processar formulários e requisições POST com dados.

8. Quais são os principais métodos HTTP e como eles são usados em rotas Express?

- **GET** – Recupera dados (ex: lista de jogos)
- **POST** – Envia dados (ex: adiciona novo jogo)
- **PUT** – Atualiza dados existentes
- **DELETE** – Remove dados

No Express, usamos assim:

```
app.post('/novogame', (req, res) => { ... });
```

9. Como você lida com erros em um aplicativo Express?

Erros podem ser tratados com blocos `try/catch`, verificações de validação, ou middlewares de erro, como:

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Algo deu errado!');
});
```

10. O que é uma API RESTful e como o Express pode ser usado para criar uma?

Uma API RESTful segue os princípios do REST (stateless, recursos identificados por URLs, uso de métodos HTTP). Com Express, podemos criar endpoints organizados que seguem esses padrões:

app.get('/games', ...);

app.post('/games', ...);

app.put('/games/:id', ...);

app.delete('/games/:id', ...);