UniSENAI

Módulo 1: Introdução ao Node.js

Módulo 1: Introdução ao Node.js

1.1 O que é Node.js?

Conceito

Node.js é uma plataforma de execução de JavaScript baseada no motor V8 do Google, projetada para criar aplicações de alto desempenho e escaláveis. Com o Node.js, é possível executar códigos JavaScript fora do navegador, o que o torna ideal para aplicações do lado do servidor.

Principais Características

- **Arquitetura Single-threaded:** Processa várias requisições simultaneamente usando o modelo non-blocking.
- **Programação Assíncrona:** Utiliza callbacks, promises e async/await para lidar com operações de I/O.
- **Eficiente:** Ideal para aplicações em tempo real, como chats e streaming.
- **Grande Ecossistema:** A plataforma utiliza o npm (Node Package Manager), que possui milhares de bibliotecas.
- **Motor V8:** Responsável por converter código JavaScript em código máquina de forma eficiente.

Importância e Aplicações

- Desenvolvimento de APIs REST e GraphQL.
- Servidores web e aplicações em tempo real (WebSockets).
- Sistemas de fila e processamento em segundo plano.
- Microserviços escaláveis e flexíveis.

Exemplo de Uso

```
console.log('Node.js está funcionando!');
```

Execute com o comando:

```
node arquivo.js
```

1.2 Arquitetura e Modelo de Execução

Como o Node.js funciona?

O Node.js utiliza um loop de eventos (Event Loop) para gerenciar tarefas assíncronas. A plataforma é baseada no conceito de callbacks e promessas, o que permite que uma aplicação processe várias tarefas sem bloquear o fluxo.

Componentes Chave

- **Event Loop:** Mecanismo que gerencia eventos e callbacks.
- **Call Stack:** Pilha de execução para códigos sincronos.
- **Thread Pool:** Gerencia tarefas de I/O, como operações de leitura e escrita em disco.
- **Callback Queue:** Fila onde os eventos aguardam para serem executados no loop principal.
- **Motor V8:** Executa o código JavaScript e converte-o em código máquina.

Definições Importantes

- **Evento:** Uma ação que ocorre no sistema, como uma conexão de cliente, leitura de arquivo ou timeout.
- **Callback:** Uma função passada como argumento para outra função, executada quando uma tarefa é concluída.
- **Thread:** Unidade básica de execução dentro de um processo. No Node.js, threads são usadas para tarefas de I/O no Thread Pool.
- **Queue:** Fila onde tarefas pendentes aguardam sua vez de serem processadas pelo Event Loop.

Single-threaded vs Multi-threaded

- **Single-threaded:** O Node.js utiliza um único thread principal para processar eventos.
- **Multi-threaded:** Processos podem usar múltiplos threads para executar tarefas paralelamente, como ocorre em algumas APIs específicas do Node.js, como `worker_threads`.

Diagrama Simplificado

1. Requisições chegam ao Event Loop.
2. Tarefas leves são processadas diretamente.
3. Operações demoradas são delegadas ao Thread Pool.

Exemplo de Execução Assíncrona

```
setTimeout(() => {
  console.log('Essa mensagem aparece depois de 2 segundos');
}, 2000);

console.log('Essa mensagem aparece primeiro');
```

Saída:

```
Essa mensagem aparece primeiro
Essa mensagem aparece depois de 2 segundos
```

1.3 Configurando o Ambiente de Desenvolvimento

Instalação do Node.js

1. Acesse o site oficial: Node.js.
2. Baixe a versão recomendada (LTS).
3. Verifique a instalação:

```
node -v
npm -v
```

Configuração do Editor de Código

- **Recomendado:** VS Code.
- Instale a extensão "Node.js Extension Pack".

Criando seu Primeiro Projeto

1. Crie uma pasta e inicialize o projeto:

```
mkdir meu-projeto
cd meu-projeto
npm init -y
```

2. Adicione um arquivo `index.js` com o seguinte conteúdo:

```
console.log('Meu primeiro projeto Node.js!');
```

3. Execute o projeto:

```
node index.js
```

1.4 Gerenciamento de Pacotes com npm

O que é npm?

O npm (Node Package Manager) é o gerenciador de pacotes oficial do Node.js. Ele permite adicionar bibliotecas externas ao seu projeto, facilitando o desenvolvimento.

Comandos Principais

- **Instalar uma biblioteca:**

```
npm install nome-da-biblioteca
```

- **Instalar como dependência de desenvolvimento:**

```
npm install nome-da-biblioteca --save-dev
```

- **Listar dependências:**

```
npm list
```

- **Remover uma biblioteca:**

```
npm uninstall nome-da-biblioteca
```

O Arquivo `package.json`

O `package.json` é o arquivo de configuração do seu projeto Node.js, contendo informações como:

- Nome e versão do projeto.
- Dependências e dependências de desenvolvimento.
- Scripts customizados para executar tarefas automatizadas.

Exemplo Prático

1. Instale o pacote `chalk` para colorir saídas no terminal:

```
npm install chalk
```

2. Modifique o `index.js`:

```
const chalk = require('chalk');
console.log(chalk.green('Texto em verde!'));
```

3. Execute o arquivo:

```
node index.js
```

Explorando o Ecossistema npm

O npm possui um vasto repositório de pacotes. Alguns pacotes populares incluem:

- `express`: Framework para criação de servidores.
- `axios`: Cliente HTTP para realizar requisições a APIs.
- `dotenv`: Gerenciamento de variáveis de ambiente.

Lista de Exercícios

Questões Teóricas

1. Explique o que é Node.js e suas principais características.
2. Qual é a diferença entre o modelo single-threaded e o multi-threaded?
3. O que é o Event Loop no Node.js?
4. Liste três aplicações onde o Node.js é amplamente utilizado.
5. Como o npm facilita o desenvolvimento de aplicações?
6. Quais os passos para criar um projeto Node.js?
7. Cite dois exemplos de pacotes populares no npm e suas utilidades.
8. O que é o arquivo `package.json` e qual sua importância?
9. Explique o conceito de callback no Node.js.
10. Qual é o papel do motor V8 no Node.js?

Questões Práticas

1. Instale o Node.js e verifique a versão instalada.
2. Crie um projeto Node.js e exiba a mensagem "Bem-vindo ao Node.js!".
3. Escreva um código que exiba uma mensagem com um atraso de 3 segundos.
4. Instale o pacote `axios` e use-o para fazer uma requisição GET para uma API.
5. Use o pacote `chalk` para exibir uma mensagem de erro em vermelho no terminal.
6. Crie um script que leia um arquivo `texto.txt` e exiba o conteúdo no console.
7. Escreva uma função que use callbacks para somar dois números.
8. Adicione uma dependência de desenvolvimento ao seu projeto e verifique-a no arquivo `package.json`.
9. Crie um arquivo JSON com informações de um usuário e leia-o usando Node.js.
10. Desenvolva um script que exiba a data e hora atuais formatadas no terminal.

Instruções para a Entrega das Atividades

1. **Elaboração e Envio do Arquivo**
   - Responda todas as questões de forma clara e objetiva.
   - Gere um arquivo no formato **.PDF** contendo as respostas de cada questão.
   - Envie o arquivo para os e-mails dos professores responsáveis.
2. **Validação da Atividade**
   - Após o envio do arquivo, procure o(s) professor(es) para realizar a validação da atividade.
   - **Não inicie a próxima atividade sem antes validar a anterior com o professor.**
3. **Forma de Validação**
   - **Explicação Verbal**: Explique cada resposta verbalmente ao(s) professor(es).
   - **Perguntas e Respostas**: Esteja preparado para responder aos questionamentos do(s) professor(es) sobre o conteúdo das respostas.
   - **Orientação**: Receba orientações sobre a apresentação do(s) tema(s).