Linguagem de Programação JavaScript

02 - Node.js no backend, introdução e elementos da linguagem JavaScrip

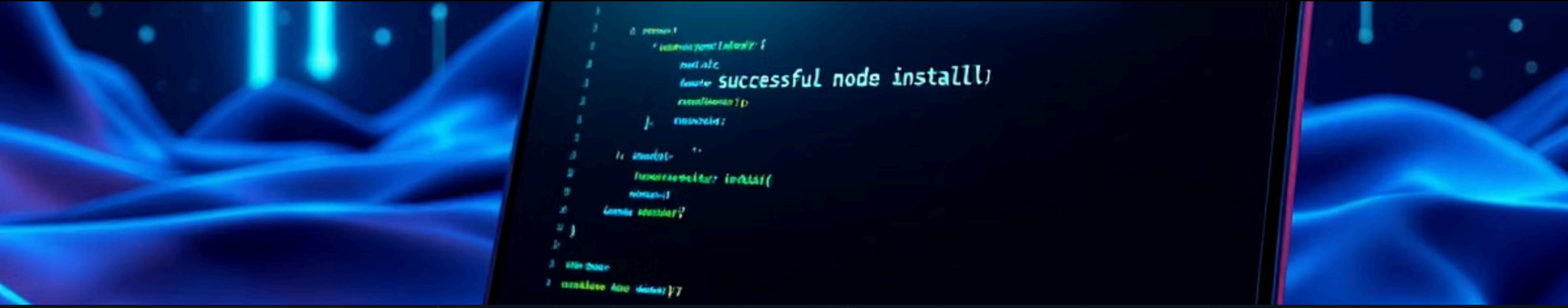
O que é Node.js?

Definição

Node.js é um ambiente de execução JavaScript que permite executar código JavaScript fora do navegador, sendo amplamente utilizado para desenvolvimento de servidores e aplicações backend.

Características Principais

- Execução assíncrona e orientada a eventos
- Alto desempenho para aplicações em tempo real
- Ecossistema robusto com milhares de bibliotecas



Instalação do Node.js

01

Baixar o Instalador

Acesse o site oficial node.js.org e selecione a versão LTS (Long Term Support), recomendada para garantir estabilidade e suporte prolongado.

02

Executar o Instalador

Abra o arquivo baixado e siga as instruções do assistente. Aceite os termos de uso e mantenha as configurações padrão para uma instalação adequada.

03

Verificar a Instalação

Abra o terminal do seu sistema operacional e execute o comando `node -v` para confirmar que a instalação foi realizada com sucesso.

Verificando a Instalação

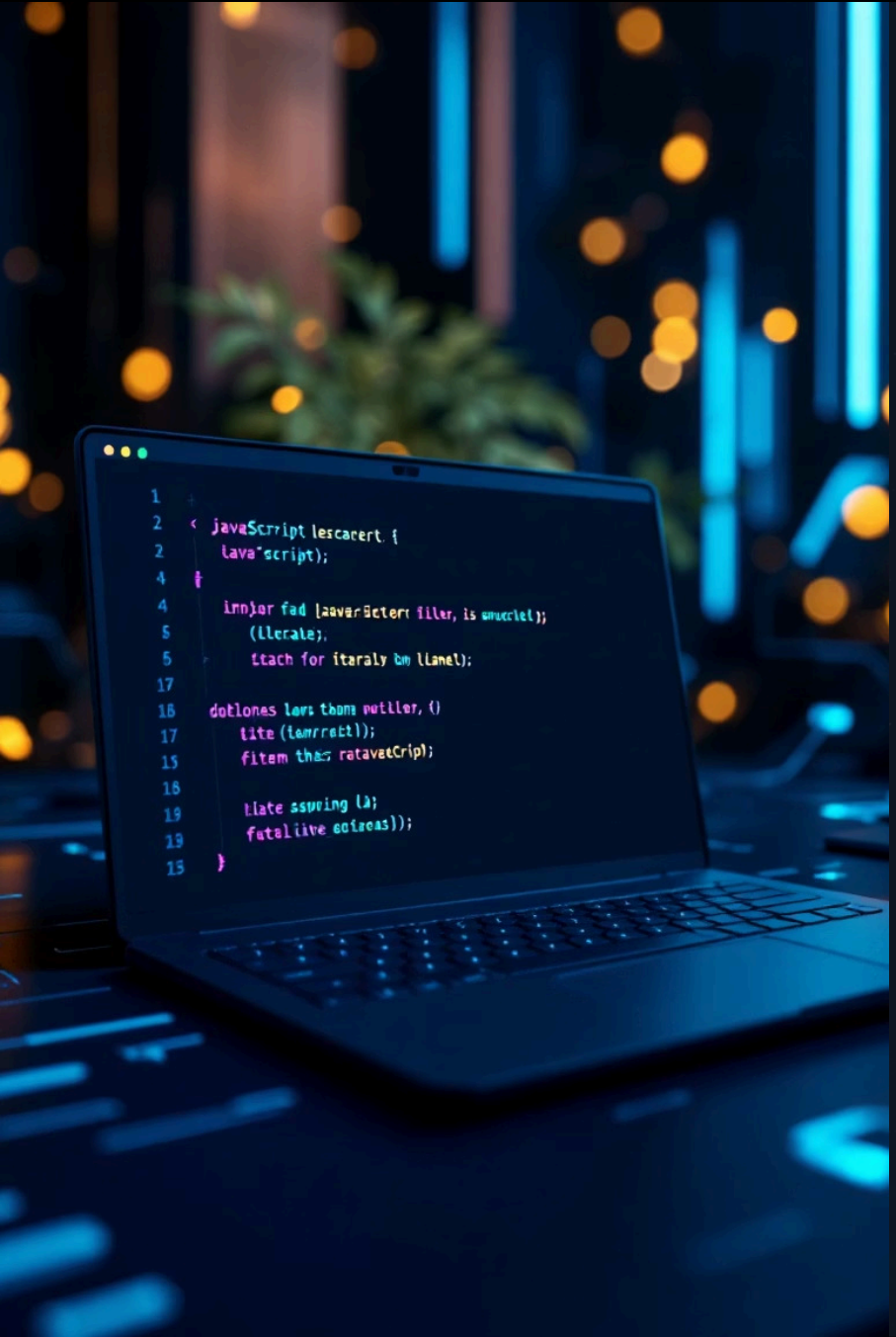
Passos para Verificação

1. Abra o terminal do sistema (Prompt de Comando no Windows, Terminal no Linux/macOS)
2. Feche e reabra editores de código como VS Code para carregar variáveis de ambiente
3. Digite `node -v` e pressione Enter
4. Observe o número da versão exibido

```
$ node -v
v20.10.0

$ npm -v
10.2.3
```

Se os números de versão aparecerem, a instalação foi concluída corretamente e você está pronto para começar a programar.



Executando Seu Primeiro Código

Criar Arquivo JavaScript

Crie um novo arquivo com extensão .js, por exemplo: `meu_script.js`. Esta extensão indica que o arquivo contém código JavaScript.

Escrever o Código

Dentro do arquivo, escreva seu código. Exemplo simples: `console.log("Olá, Node.js!");`

Executar no Terminal

Navegue até a pasta do arquivo no terminal e execute: `node meu_script.js`. A mensagem será exibida no console.

Declaração de Variáveis em JavaScript

A escolha entre var, let e const afeta o escopo e a capacidade de reatribuição das variáveis em seu código.

var

- Escopo de função
- Pode ser redeclarada
- Pode ser reatribuída
- Uso tradicional, menos recomendado atualmente

let

- Escopo de bloco
- Não pode ser redeclarada
- Pode ser reatribuída
- Recomendado para valores que mudam

const

- Escopo de bloco
- Não pode ser redeclarada
- Não pode ser reatribuída
- Recomendado para valores constantes

Tipos Primitivos em JavaScript

String

Representa texto entre aspas simples, duplas ou crases. Exemplo: `'Olá'`, `"mundo"`

Number

Representa números inteiros e decimais. Exemplo: `10`, `3.14`

Boolean

Valor lógico verdadeiro ou falso. Exemplo: `true`, `false`

Null

Representa ausência intencional de valor. Exemplo: `null`

Undefined

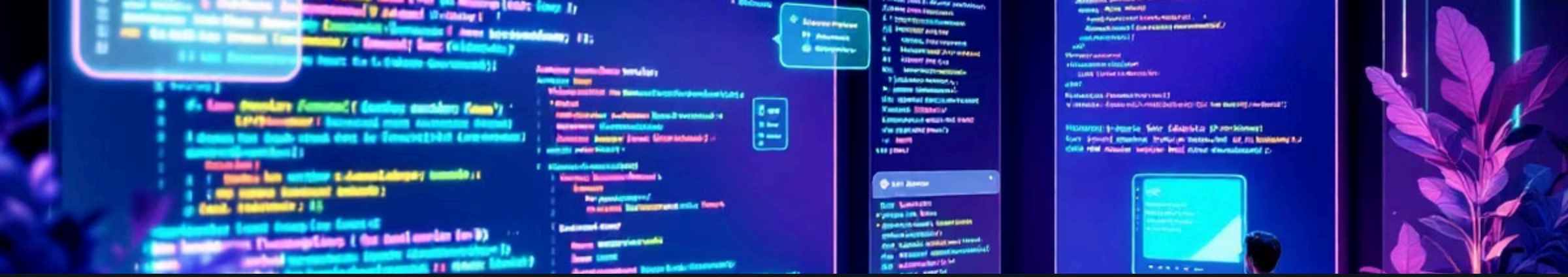
Variável declarada sem valor atribuído. Exemplo: `let x;`

BigInt

Inteiros com precisão arbitrária para valores muito grandes

Symbol

Identificadores únicos para propriedades de objetos



Tipos de Objetos em JavaScript

Object

Coleção de propriedades com chave e valor, estruturando dados relacionados.

```
{
  nome: 'Maria',
  idade: 25
}
```

Array

Estrutura para armazenar múltiplos valores acessados por índices numéricos começando em 0.

```
[1, 'texto', true]
```

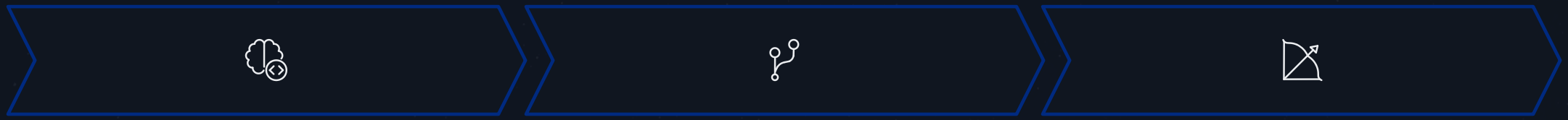
Function

Funções são tratadas como valores, podendo ser atribuídas a variáveis para execução posterior.

```
const somar =
 function(a, b) {
 return a + b;
 }
```

Estruturas Condicionais: if/else

As estruturas condicionais permitem que o programa tome decisões baseadas em condições específicas.



if

Testa uma condição inicial. Se verdadeira, executa o bloco de código correspondente.

else if

Avalia condições adicionais quando a anterior é falsa, permitindo múltiplas verificações encadeadas.

else

Executado quando todas as condições anteriores são falsas, funcionando como opção padrão.

```
let hora = 17;

if (hora >= 6 && hora < 12) {
 console.log("Bom dia!");
} else if (hora >= 12 && hora < 18) {
 console.log("Boa tarde!");
} else {
 console.log("Boa noite!");
}
```

Próximos Passos no Desenvolvimento Backend

Aprofundar Conhecimentos

Explore módulos nativos do Node.js como fs (sistema de arquivos), http (servidores web) e path (manipulação de caminhos).

Frameworks Populares

Estude frameworks como Express.js para criar APIs REST de forma eficiente e estruturada.

Gerenciamento de Pacotes

Domine o npm (Node Package Manager) para instalar, gerenciar e compartilhar bibliotecas JavaScript.

Prática Contínua

Desenvolva projetos pessoais, contribua em projetos de código aberto e mantenha-se atualizado com as melhores práticas da comunidade.