# Apostila Node.js: Módulos fs e path

***Explicação detalhada, exemplos práticos e exercícios***

Gerado automaticamente por ChatGPT – 12/06/2025

# Parte I – Módulo fs (File System)

O módulo fs oferece uma API para interagir com o sistema de arquivos: ler, criar, escrever, copiar, mover e deletar arquivos e diretórios. Ele possui tanto versões assíncronas (com callback ou Promises) quanto síncronas.

**fs.readFile / fs.readFileSync** – Lê o conteúdo de um arquivo.

**fs.writeFile / fs.writeFileSync** – Escreve (sobrescreve) dados em um arquivo.

**fs.appendFile / fs.appendFileSync** – Acrescenta dados ao final de um arquivo.

**fs.readdir / fs.readdirSync** – Lista arquivos de um diretório.

**fs.stat / fs.statSync** – Retorna metadados de arquivos.

**fs.mkdir / fs.mkdirSync** – Cria diretórios.

**fs.copyFile / fs.copyFileSync** – Copia arquivos.

**fs.rm / fs.rmSync** – Remove arquivos ou diretórios.

### *Exemplo prático: Ler e escrever arquivos*

```
const fs = require('fs');
fs.readFile('entrada.txt', 'utf8', (err, data) => {
  if (err) return console.error(err);
  fs.writeFile('saida.txt', data.toUpperCase(), err => {
    if (err) throw err;
    console.log('Arquivo salvo!');
  });
});
```

### *Exemplo prático: Streams para arquivos grandes*

```
const fs = require('fs');
fs.createReadStream('grande.iso')
  .pipe(fs.createWriteStream('copia.iso'))
  .on('finish', () => console.log('Cópia concluída!'));
```

## Exercícios – Módulo fs

■ 1. Crie um script que verifique se o arquivo 'dados.txt' existe; se não existir, crie-o com o conteúdo 'Hello!'.

■ 2. Escreva um programa que leia 'dados.txt' e imprima seu conteúdo no console.

■ 3. Acrescente a linha 'Nova linha' ao final de 'dados.txt'.

■■ 4. Liste todos os arquivos do diretório atual exibindo tamanho em bytes.

■■ 5. Copie todos os arquivos '.txt' do diretório atual para uma pasta 'backup'.

■■ 6. Usando Promises, leia três arquivos em paralelo e junte seus conteúdos em 'todos.txt'.

■■ 7. Utilize streams para comprimir 'grande.log' em 'grande.log.gz' (dica: zlib).

■■ 8. Crie um script que leia JSON de 'config.json', modifique uma propriedade e salve novamente.

■■■ 9. Implemente uma função async que apague todos os arquivos com mais de 30 dias em 'logs/'.

■■■ 10. Crie um utilitário CLI que conte quantas linhas, palavras e caracteres existem em todos os '.md' de um diretório (estilo wc).

# Respostas – Módulo fs

### *1.*

```
const fs = require('fs');
if (!fs.existsSync('dados.txt')) fs.writeFileSync('dados.txt', 'Hello!');
```

### *2.*

```
const data = require('fs').readFileSync('dados.txt', 'utf8');
console.log(data);
```

### *3.*

```
require('fs').appendFileSync('dados.txt', '\nNova linha');
```

### *4.*

```
const fs = require('fs');
fs.readdirSync('.').forEach(f => console.log(f, fs.statSync(f).size));
```

### *5.*

```
const fs = require('fs');
fs.mkdirSync('backup', { recursive: true });
fs.readdirSync('.').filter(f => f.endsWith('.txt'))
  .forEach(f => fs.copyFileSync(f, `backup/${f}`));
```

### *6.*

```
const { readFile, writeFile } = require('fs').promises;
Promise.all(['a.txt','b.txt','c.txt'].map(f=>readFile(f,'utf8')))
  .then(parts => writeFile('todos.txt', parts.join('\n')));
```

### *7.*

```
const fs = require('fs');
const zlib = require('zlib');
fs.createReadStream('grande.log')
  .pipe(zlib.createGzip())
  .pipe(fs.createWriteStream('grande.log.gz'));
```

### *8.*

```
const fs = require('fs');
const cfg = JSON.parse(fs.readFileSync('config.json'));
cfg.ativo = !cfg.ativo;
fs.writeFileSync('config.json', JSON.stringify(cfg,null,2));
```

### *9.*

```
const fs = require('fs').promises;
const path = require('path');
async function limparLogs() {
  const limite = Date.now() - 30*24*60*60*1000;
  for (const f of await fs.readdir('logs')) {
    const p = path.join('logs', f);
    const s = await fs.stat(p);
    if (s.mtimeMs < limite) await fs.rm(p);
  }
}
```

```
limparLogs();
```

### *10.*

```
// Resposta extensa; ver gist.
```

## Parte II – Módulo path

O módulo path fornece utilidades para manipular caminhos de forma independente de SO.

**path.join()** – Une segmentos normalizando separadores.

**path.resolve()** – Gera caminho absoluto.

**path.basename()** – Obtém o nome do arquivo.

**path.extname()** – Retorna a extensão.

**path.dirname()** – Diretório pai.

**path.parse()** – Decompõe caminho em objeto.

**path.format()** – Monta string a partir de objeto.

**path.relative()** – Caminho relativo entre dois absolutos.

### *Exemplo prático: Construir caminho portável*

```
const path = require('path');
const full = path.join(__dirname, 'public', 'img', 'logo.png');
console.log(full);
```

## Exercícios – Módulo path

■ 1. Obtenha a extensão do arquivo 'foto.jpeg'.

■ 2. Retorne apenas o nome do arquivo (sem extensão) de '/usr/local/bin/script.sh'.

■ 3. Construa o caminho para 'logs/app.log' baseado no diretório atual.

■■ 4. Calcule o caminho relativo entre '/var/www' e '/var/www/assets/img/logo.png'.

■■ 5. Crie uma função que substitua a extensão de um arquivo por '.bak'.

■■ 6. Use path.normalize para limpar o caminho 'folder///sub//..//file.txt'.

■■ 7. Verifique se dois caminhos se referem ao mesmo arquivo.

■■ 8. Gere o caminho absoluto do diretório pai do arquivo atual.

■■■ 9. Implemente um utilitário CLI que converta caminhos Windows em POSIX e vice■versa.

■■■ 10. Analise todos os imports de um arquivo JS e converta para caminhos absolutos.

# Respostas – Módulo path

### *1.*

```
const path = require('path');
console.log(path.extname('foto.jpeg'));
```

### *2.*

```
const { name } = require('path').parse('/usr/local/bin/script.sh');
console.log(name);
```

### *3.*

```
const path = require('path');
console.log(path.join(process.cwd(), 'logs', 'app.log'));
```

### *4.*

```
const path = require('path');
console.log(path.relative('/var/www', '/var/www/assets/img/logo.png'));
```

### *5.*

```
const path = require('path');
function toBak(file) {
  const { dir, name } = path.parse(file);
  return path.join(dir, name + '.bak');
}
```

### *6.*

```
const path = require('path');
console.log(path.normalize('folder///sub//..//file.txt'));
```

### *7.*

```
const path = require('path');
function same(a,b){return path.resolve(a)===path.resolve(b);}
```

### *8.*

```
const path = require('path');
console.log(path.dirname(__dirname));
```

### *9.*

```
// Use path.win32 e path.posix junto com regex CLI.
```

### *10.*

```
// Envolve regex + path.resolve para cada import.
```