Módulos em Node.js

Existem vários módulos relevantes para o desenvolvimento de aplicações em Node.js, entre eles, **fs** (File System), **http**, **path**, **os** e **crypto**.

File System (fs)

O módulo **fs** permite interagir com o sistema de arquivos e diretórios.

- `readFile, writeFile, appendFile, unlink, readdir, mkdir, rm`

```
const fs = require('fs');

fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) {
    console.error('Erro ao ler o arquivo:', err);
    return;
  }
  console.log('Conteúdo do arquivo:', data);
});
```

Servidor Web (http)

O módulo **http** permite a criação de servidores Web e a manipulação de requisições e respostas HTTP.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello world, Node.js!\n');
});

server.listen(port, hostname, () => {
  console.log(`Servidor rodando em http://${hostname}:${port}/`);
});
```

Sistema Operacional (os)

O módulo **os** fornece funções utilitárias para obter informações sobre o sistema operacional, como a arquitetura, plataforma, memória disponível e informações da CPU.

```
const os = require('os');

// Informações sobre o sistema operacional
console.log('Arquitetura do Sistema:', os.arch());
console.log('Plataforma do Sistema:', os.platform());
console.log('Número de CPUs:', os.cpus().length);
console.log('Memória Total (bytes):', os.totalmem());
console.log('Memória Livre (bytes):', os.freemem());

// Informações sobre a rede
console.log('Interfaces de Rede:', os.networkInterfaces());
```

Manipulação de Caminhos (path)

O módulo path fornece utilitários para trabalhar com caminhos de arquivos e diretórios.

```
const path = require('path');

// Manipulação de caminhos
const dirName = path.dirname('/path/to/file.txt');
const baseName = path.basename('/path/to/file.txt');
const extName = path.extname('/path/to/file.txt');
const joinPath = path.join('/path', 'to', 'file.txt');
const resolvePath = path.resolve('file.txt');

console.log('Diretório:', dirName);
console.log('Nome do Arquivo:', baseName);
console.log('Extensão do Arquivo:', extName);
console.log('Caminho Combinado:', joinPath);
console.log('Caminho Absoluto:', resolvePath);
```

Criptografia (crypto)

O módulo crypto fornece funcionalidades de criptografia, como criação de hash para segurança de dados.

```
const crypto = require('crypto');

// Criação de um hash SHA-256 e SHA-512
const sha256Hash = crypto.createHash('sha256').update('senha123').digest('hex');
console.log('Hash SHA-256:', sha256Hash);

// Criação de um HMAC (Hash-based Message Authentication Code)
const secret = 'mySecretKey';
const hmac = crypto.createHmac('sha256', secret).update('mensagem').digest('hex');
console.log('HMAC SHA-256:', hmac);
```

Além destas criptografias, há outras como **MD5**, mais antiga e com cada vez menos uso, e **Bcrypt**, que é uma das melhores escolhas para armazenar senhas devido ao seu design que inclui um **sal** e permite ajustar a complexidade computacional (fator de custo).

É necessário instalar o pacote bcrypt: **`npm install bcrypt`**

```
const bcrypt = require('bcrypt');
const saltRounds = 10;
const senha = 'senha123';

bcrypt.hash(senha, saltRounds, (erro, hash) => {
  if (erro) throw erro;
  console.log('Hash Bcrypt:', hash);

  // Verificando a senha
  bcrypt.compare(senha, hash, (erro, resultado) => {
    if (erro) throw erro;
    console.log('Senha correta:', resultado);
  });
});
```