Programação Web Back-end

Assunto: Manipulação de arquivos e diretórios

Adriano Rivolli

rivolli@utfpr.edu.br

Universidade Tecnológica Federal do Paraná (UTFPR)
Câmpus Cornélio Procópio
Departamento de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Conteúdo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Módulo path

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Visão geral

- Módulo que proporciona funcionalidades para manipulação de **paths**
- Faz a abstração do sistema operacional em relação aos padrões de nomes
- Para incluir o módulo basta utilizar:

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

Principais métodos

- `path.basename(caminho)`

  Obtém o nome do arquivo a partir de um caminho completo

- `path.dirname(caminho)`

  Obtém o diretório pai de um arquivo

- `path.extname(caminho)`

  Obtém a extensão do arquivo

- `path.isAbsolute(caminho)`

  Verifica se o path representa um caminho absoluto

- `path.join(parte1, parte2)`

  Concatena caminhos de forma inteligente

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Exemplos

```
const filePath = '/user/local/file.txt';
const fileName = path.basename(filePath);
console.log(fileName); // Saída: 'file.txt'
const dirName = path.dirname(filePath);
console.log(dirName); // Saída: '/user/local'
const ext = path.extname(filePath);
console.log(ext); // Saída: '.txt'
const completePath = path.join('/user', 'local', 'folder', 'file.txt');
console.log(completePath); // Saída: '/user/local/folder/file.txt'
```

Diretório atual

- `__dirname` Representa o diretório atual do arquivo em execução
- Exemplo:

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

// Gera um caminho para o arquivo 'config.json' no mesmo diretório do script atual
const filePath = path.join(__dirname, 'config.json');

console.log(filePath);
// Saída: '/caminho/completo/do/diretorio/atual/config.json'
```

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Módulo fs

Visão geral

- **fs** = ***f****ile* ***s****ystem*
- Módulo que permite trabalhar com o sistema de arquivos
- Funções principais
  - Ler arquivos
  - Criar arquivos
  - Alterar arquivos
  - Renomear arquivos
  - Manipular diretórios

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Importação

- `const fs = require("fs");`
  Importa o módulo fs
- Os métodos disponíveis por padrão são assíncronos
- Cada método possui uma versão síncrona (acresce o sufixo `Sync`)
- Cada método possui uma versão com **Promisses** (permite o uso de `async`/`await`)

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Arquivos

Ler o conteúdo de um arquivo

- `fs.readFile()`
  Lê o conteúdo do arquivo
- Argumentos:
  - nome do arquivo
  - Codificação do arquivo (default = 'utf8')
  - Função de *callback* que recebe o **erro** e o **conteúdo**
- Exemplo:

```
// Use fs.readFile() method to read the file
fs.readFile('Demo.txt', 'utf8', function(err, data){
    // Display the file content
    console.log(data);
});
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Criando arquivos

- `fs.appendFile()`
  Adiciona conteúdo a um arquivo. Se o arquivo não existir, um novo é criado
- `fs.writeFile()`
  Sobrescreve um arquivo com o novo conteúdo. Se o arquivo não existir, um novo é criado
- Argumentos:
  - nome do arquivo
  - Conteúdo do arquivo
  - Callback de erro

Criando um arquivo - Exemplos

```
fs.appendFile('mynewfile1.txt', 'Hello content!', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

Excluindo arquivos

- `fs.unlink()`
  Apaga um arquivo
- Argumentos:
  - Nome do arquivo
  - Callback de erro
- Exemplo:

```
fs.unlink('mynewfile2.txt', function (err) {
  if (err) throw err;
  console.log('File deleted!');
});
```

Renomeando arquivos

- `fs.rename()`
  Renomeia um arquivo
- Argumentos:
  - Nome do arquivo
  - Novo nome do arquivo
  - Callback de erro
- Exemplo:

```
fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) {
  if (err) throw err;
  console.log('File Renamed!');
});
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Verificar se um arquivo existe

- `fs.existsSync()`
  Verifica se um arquivo existe
- Argumentos:
  - Nome do arquivo
- Exemplo:

```
if (fs.existsSync(fname)) { ... } else { ... }
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Copiar um arquivo

- `fs.copyFile()`
  Faz a cópia de um arquivo
- Argumentos:
  - Nome do arquivo de origem
  - Nome do arquivo de destino
  - Callback de erro

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Diretórios

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Lê o conteúdo de um diretório

- `fs.readdir()`
  Lê o conteúdo de um diretório
- Argumentos:
  - Nome do diretório
  - Encoding (default: 'utf8')
  - Função de *callback*, recebe o **erro** e a lista de **arquivos**

Exemplo de leitura

```
// Function to get current filenames
// in directory
fs.readdir(__dirname, (err, files) => {
  if (err)
    console.log(err);
  else {
    console.log("\nCurrent directory filenames:");
    files.forEach(file => {
      console.log(file);
    })
  }
})
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Criar um diretório

- `fs.mkdir()`
  Cria um diretório
- Argumentos:
  - Nome do diretório
  - Opções: `{recursive: true}`
  - Callback de erro

Exemplo de criação

```
// Include fs and path module
const fs = require('fs');
const path = require('path');

fs.mkdir(path.join(__dirname, 'test'),
    (err) => {
        if (err) {
            return console.error(err);
        }
        console.log('Directory created successfully!');
    });
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Excluir um diretório

- `fs.rmdir()`
  Exclui um diretório vazio
- Argumentos:
  - Nome do diretório
  - Opções: `{recursive: true}`
  - Callback de erro

Exemplo de exclusão

```
// Trying to delete nested directories
// without the recursive parameter
fs.rmdir("directory_one", {
  recursive: false,
}, (error) => {
  if (error) {
    console.log(error);
  }
  else {
    console.log("Non Recursive: Directories Deleted!");
  }
});
```

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Meta-dados

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Statisticas

- `fs.stat()`
  Retorna informações sobre um arquivo ou diretório
- Argumentos:
  - Nome do arquivo/diretório
  - Callback de retorno recebe o **erro** e um objeto **stats**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Objeto Stats

- `stats.isDirectory()`
  Retorna **true** se o objeto é um diretório
- `stats.isFile()`
  Retorna **true** se o objeto é um arquivo
- `stats.isSymbolicLink()`
  Retorna **true** se o objeto é um atalho/link

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Exemplo de meta-dados

```
// Usa fs.stat para obter informações sobre o arquivo
fs.stat(filePath, (err, stats) => {
  if (err) {
    return console.error(`Erro ao acessar o arquivo: ${err.message}`);
  }

  // Exibe algumas informações do arquivo
  console.log(`É um diretório? ${stats.isDirectory()}`);
  console.log(`É um arquivo? ${stats.isFile()}`);
  console.log(`Tamanho do arquivo: ${stats.size} bytes`);
  console.log(`Data de criação: ${stats.birthtime}`);
  console.log(`Última modificação: ${stats.mtime}`);
});
```

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Usando fs com Promises

Promises

- Um mecanismo para lidar com operações assíncronas
- Representam um valor que pode estar disponível agora, no futuro, ou nunca
- Uma **Promise** está em um destes estados:
  - **Pending (Pendente):** Estado inicial, nem cumprida nem rejeitada
  - **Fulfilled (Cumprida):** Significa que a operação foi concluída com sucesso
  - **Rejected (Rejeitada):** Significa que a operação falhou

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Async/Await

- São extensões da sintaxe de JavaScript que simplificam o trabalho com *Promises*
- Torna o código assíncrono mais fácil de escrever e de entender
- A palavra **async** é utilizada na definição da função assíncrona

  Isso indica que a função retornará uma Promisse
- A palavra **await** é utilizada na chamada de uma função assíncrona

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Funções Async

- Para declarar uma função assíncrona, basta incluir a palavra-chave **async** antes da declaração da função `async function minhaFuncaoAsync(){ ... }`
- Isso faz com que a função retorne uma promessa automaticamente
  - Se a função retorna um valor, a *Promise* será resolvida com esse valor
  - Se a função gera um erro, a *Promise* será rejeitada com esse erro

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Await

- A palavra-chave **await** é utilizada antes de chamar uma função marcada com **async**

```
let x = await minhaFuncaoAsync()
```

- Isso faz com que a resolução da **Promise** seja aguardada
- Ao usar **await** dentro de uma função, esta função se torna assíncrona e deve ser marcada com a palavra-chave **async**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Usando fs com Promises

- Para utilizar as funções de **fs** baseadas em *Promises*:
  `const fs = require("fs").promises;`
- Não é necessário passar uma função de *callback* para as chamadas
- É necessário utilizar a palavra **await** nas chamadas das funções
- Também é necessário incluir os códigos em funções marcados com **async**

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Exemplos

```
1   const fs = require('fs').promises;
2
3   async function lerArquivo(caminho) {
4     return await fs.readFile(caminho, 'utf8');
5   }
6
7   async function escreverArquivo(caminho, conteudo) {
8       await fs.writeFile(caminho, conteudo);
9   }
10
11  async function adicionarAoArquivo(caminho, conteudo) {
12      await fs.appendFile(caminho, conteudo);
13  }
14
15  async function lerDiretorio(caminho) {
16      const arquivos = await fs.readdir(caminho);
17      return arquivos
18  }
```

Tipos de arquivos

Texto

- Contêm apenas texto sem uma estrutura definida
- Usam codificação de caracteres, como UTF-8, ASCII, etc.
- Utilizados para anotações, logs, páginas HTML, etc.
- Cada linha é separada por quebras de linha (\n ou \r\n)
- Vantagens e desvantagens:
  - Fácil de manipular e compatível com qualquer editor
  - Não é adequado para representar dados estruturados

CSV

- **Comma-Separated Values**
- Arquivos de texto que armazenam dados tabulares (linhas e colunas) separados por vírgulas
  - A linha do arquivo representa um registro
  - Cada campo é separado por uma vírgula (ou outro delimitador)
- Usado para troca de dados entre aplicações
- Dificulta a leitura se possui conteúdo de texto com vírgulas e/ou espaços

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JSON

- **JavaScript Object Notation**
- Formato de texto leve para troca de dados estruturados
- Representa dados como objetos, listas e estruturas aninhadas
- Requer o uso das funções: `JSON.stringify()` e `JSON.parse()`