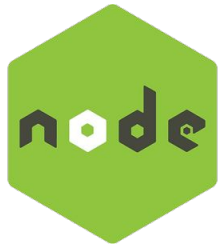
# NodeJS




Prof. Celso Henrique Masotti

# Você verá nesta aula:

**Objetivo**: operações de arquivo tipo texto.

FS
FS - Ações Desejadas
FS - Arquivos HTML

# FS

O “fs” ou “File System” é um módulo do nodeJS para manipulação de arquivos. Com este recurso você pode criar, renomear, escrever e deletar arquivos.

Precisamos requerer o módulo em nosso algoritmo.

var fs = require(“fs”);

Veja:

```
1   var http = require("http");
2   var fs = require("fs");
```

# FS

Vamos construir a função que trabalha com o servidor no interior da função com as propriedades de “fs”, bem como a gestão de erros, que porventura possam ocorrer, algo como “fs.[ação desejada](*param1*,*param2*)”.

- “param1” é o endereçamento das pastas com o arquivo que será manipulado;
- “param2” é uma função que recebe erros do callback.

Um dos desafios desse modelo é como fazer a gestão de erros. E uma solução encontrada é o que é feito no Node.js: o primeiro parâmetro de qualquer callback é o objeto de erro e isso é um padrão em Node.js, é o jeito default de se trabalhar com a tecnologia, ficando “function(err,arquivo)”, para em seguida verificar se retornou erro e imprimi-lo no console. Ficando algo como:

“ fs.[ação desejada](“info.txt”, function (err,text){ }); ”

# FS - Ações Desejadas

**writeFile** - Escrevendo em arquivo. Caso o arquivo não exista ele é criado.

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

/* No exemplo abaixo, informamos o local que será criado o arquivo
toda a informação que esse arquivo conterá, e por ultimo temos nossa função callback */
fs.writeFile("./files/example.txt",'Um breve texto aqui!', function(err){
        //Caro ocorra algum erro
  if(err){
            return console.log('erro')
        }
  //Caso não tenha erro, retornaremos a mensagem de sucesso
        console.log('Arquivo Criado');
});
```

# FS - Ações Desejadas

**readFile** ou **readFileSync** - leitura de arquivo de forma assíncrona ou síncrona.

```
let files = [];//Vetor para armazenar todos os nomes dos arquivos lidos

//Lendo todos os arquivos existenstes na pasta files de forma síncrona
fs.readdirSync('./files').forEach(file => {
        //Efetuando a leitura do arquivo
        fs.readFile('./files/'+file,'utf8', function(err,data){
                //Enviando para o console o resultado da leitura
                console.log(data);
        });
});
```

# FS - Ações Desejadas

**rename** - renomeia um arquivo.

```
1  //Enviando o caminho do arquivo que queremos renomear e o caminho/nome para sua nova situação
2  fs.rename('./files/example.txt', './files/007.txt', function(err){
3      //Caso a execução encontre algum erro
4                  if(err){
5                          //A execução irá parar e mostrará o erro
6                          throw err;
7                  }else{
8        //Caso não tenha erro, apenas a mensagem será exibida no terminal
9        console.log('Arquivo renomeado');
10     }
11         });
```

# FS - Ações Desejadas

**unlink** ou **unlinkSync** - exclui um arquivo de forma assíncrona ou síncrona.

```
1   //Informando o endereço do arquivo para remoção do mesmo
2   fs.unlink("./files/example.txt");
```

# FS - Arquivos HTML

Para arquivos HTML, além da solicitação “readFile”, no interior da função que cria o servidor criamos cabeçalho para sua leitura mais eficaz vinda do callback.

response.writeHeader([*código do status http*],[*especificação do cabeçalho*]);

exemplo

response.writeHeader(*200*, {“*Content-Type*”:”*text/html*”});

.

# FS - Arquivos HTML

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes:

- Respostas de informação (100-199),
- Respostas de sucesso (200-299),
- Redirecionamentos (300-399)
- Erros do cliente (400-499)
- Erros do servidor (500-599).

# FS - Arquivos HTML

A estrutura funcional para a leitura de um arquivo HTML fica assim:

proj04 > JS app_http.js > ...

```
1   var http = require("http");
2   var fs = require("fs");
3
4   fs.readFile('./index.html', function (err, html) {
5       if (err) {
6           throw err;
7       }
8       http.createServer(function(request, response) {
9           response.writeHeader(200, {"Content-Type": "text/html"});
10          response.write(html);
11          response.end();
12      }).listen(8080)
13  });
14  console.log("Servidor ativo");
```

Obrigado

Prof. Celso Henrique Masotti