NodejsIO框架

fs (文件系统)
WHATWG URL object support
Buffer API
fs.FSWatcher 类
'change' 事件
'error' 事件
watcher.close()
fs.ReadStream 类
'close' 事件
'open' 事件
readStream.bytesRead
readStream.path
fs.Stats 类
Stat 时间值
fs.WriteStream 类
'close' 事件
'open' 事件
writeStream.bytesWritten
writeStream.path
fs.access(path[, mode], callback)
fs.accessSync(path[, mode])
fs.appendFile(file, data[, options], callback)
fs.appendFileSync(file, data[, options])
fs.chmod(path, mode, callback)
fs.chmodSync(path, mode)
fs.chown(path, uid, gid, callback)
fs.chownSync(path, uid, gid)
fs.close(fd, callback)
fs.closeSync(fd)
fs.constants
fs.createReadStream(path[, options])
fs.createWriteStream(path[, options])
fs.exists(path, callback)
fs.existsSync(path)
fs.fchmod(fd, mode, callback)
fs.fchmodSync(fd, mode)
fs.fchown(fd, uid, gid, callback)
fs.fchownSync(fd, uid, gid)
fs.fdatasync(fd, callback)
fs.fdatasyncSync(fd)
fs.fstat(fd, callback)
fs.fstatSync(fd)
fs.fsync(fd, callback)
fs.fsyncSync(fd)
fs.ftruncate(fd, len, callback)
fs.ftruncateSync(fd, len)
fs.futimes(fd, atime, mtime, callback)
fs.futimesSync(fd, atime, mtime)
fs.lchmod(path, mode, callback)
fs.lchmodSync(path, mode)
fs.lchown(path, uid, gid, callback)
fs.lchownSync(path, uid, gid)

fs.link(existingPath, newPath, callback)
fs.linkSync(existingPath, newPath)
fs.lstat(path, callback)
fs.lstatSync(path)
fs.mkdir(path[, mode], callback)
fs.mkdirSync(path[, mode])
fs.mkdtemp(prefix[, options], callback)
fs.mkdtempSync(prefix[, options])
fs.open(path, flags[, mode], callback)
fs.openSync(path, flags[, mode])
fs.read(fd, buffer, offset, length, position, callback)
fs.readdir(path[, options], callback)
fs.readdirSync(path[, options])
fs.readFile(path[, options], callback)
fs.readFileSync(path[, options])
fs.readlink(path[, options], callback)
fs.readlinkSync(path[, options])
fs.readSync(fd, buffer, offset, length, position)
fs.realpath(path[, options], callback)
fs.realpathSync(path[, options])
fs.rename(oldPath, newPath, callback)
fs.renameSync(oldPath, newPath)
fs.rmdir(path, callback)
fs.rmdirSync(path)
fs.stat(path, callback)
fs.statSync(path)
fs.symlink(target, path[, type], callback)
fs.symlinkSync(target, path[, type])
fs.truncate(path, len, callback)
fs.truncateSync(path, len)
fs.unlink(path, callback)
fs.unlinkSync(path)
fs.unwatchFile(filename[, listener])
fs.utimes(path, atime, mtime, callback)
fs.utimesSync(path, atime, mtime)
fs.watch(filename[, options][, listener])

说明

可用性

索引节点

文件名参数

fs.watchFile(filename[, options], listener)
fs.write(fd, buffer[, offset[, length[, position]]], callback)
fs.write(fd, string[, position[, encoding]], callback)
fs.writeFile(file, data[, options], callback)
fs.writeFileSync(file, data[, options])
fs.writeSync(fd, buffer[, offset[, length[, position]]])
fs.writeSync(fd, string[, position[, encoding]])

fs 常量

文件访问常量

文件打开常量

文件类型常量

文件模式常量

文件 I/O 是由简单封装的标准 POSIX 函数提供。 通过 require('fs') 使用该模块。 所有的方法都有异步和同步的形式。

异步方法的最后一个参数都是一个回调函数。 传给回调函数的参数取决于具体方法，但回调函数的第一个参数都会保留给异常。 如果操作成功完成，则第一个参数会是 null 或 undefined。

当使用同步方法时，任何异常都会被立即抛出。 可以使用 try/catch 来处理异常，或让异常向上冒泡。

异步方法的例子：

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

fs.unlink('/tmp/hello', (err) => {
  if (err) throw err;
  console.log('成功删除 /tmp/hello');
});
```

同步方法的例子：

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

fs.unlinkSync('/tmp/hello');
console.log('成功删除 /tmp/hello');
```

异步的方法不能保证执行顺序。 所以下面的例子可能会出错：

```
fs.rename('/tmp/hello', '/tmp/world', (err) => {
  if (err) throw err;
  console.log('重命名完成');
});
fs.stat('/tmp/world', (err, stats) => {
  if (err) throw err;
  console.log(`文件属性: ${JSON.stringify(stats)}`);
});
```

fs.stat 可能在 fs.rename 之前执行。 正确的方法是把回调链起来。

```
fs.rename('/tmp/hello', '/tmp/world', (err) => {
  if (err) throw err;
  fs.stat('/tmp/world', (err, stats) => {
    if (err) throw err;
    console.log(`文件属性: ${JSON.stringify(stats)}`);
  });
});
```

在繁忙的进程中，建议使用异步的方法。 同步的方法会阻塞整个进程，直到完成（停止所有连接）。

可以使用文件名的相对路径。 路径是相对 process.cwd() 的。

大多数 fs 函数可以省略回调函数，在这种情况下，会使用默认的回调函数。 若要追踪最初的调用点，可设置 NODE_DEBUG 环境变量：

注意：不建议省略异步方法的回调函数，未来的版本可能会导致抛出错误。

```
$ cat script.js
function bad() {
  require('fs').readFile('/');
}
bad();

$ env NODE_DEBUG=fs node script.js
fs.js:88
        throw backtrace;
        ^
Error: EISDIR: illegal operation on a directory, read
    <stack trace.>
```