目前没有整理的

（1）文档为英文：

Cluster (集群)

Crypto (加密)

DNS (域名服务器)

Domain (域)

Process (进程)

punycode

TLS (安全传输层)

V8

VM (虚拟机)

ZLIB (压缩)

1. 暂时无用

C/C++ 插件

Debugger (调试器)

1 Assert (断言)

断好失 等深严两否 扔不 如

assert(value[, message])

assert.ok(value[, message])

assert.fail(actual, expected, message, operator)

assert.equal(actual, expected[, message]) assert.notEqual(actual, expected[, message])

assert.deepEqual(actual, expected[, message]) assert.notDeepEqual(actual, expected[, message])

assert.strictEqual(actual, expected[, message]) assert.notStrictEqual(actual, expected[, message])

assert.deepStrictEqual(actual, expected[, message]) assert.notDeepStrictEqual(actual, expected[, message])

assert.throws(block[, error][, message])

assert.doesNotThrow(block[, error][, message])

assert.ifError(value)

2 Buffer

池从4分不慢 是缓编 字比接

长项 换三

读写有无大小 正 16 32

读写有无 8

读写大小 浮达

转串j前后 比复入等填包键段值写

类

Buffer.poolSize

Buffer.from(array)

Buffer.from(arrayBuffer[, byteOffset[, length]])

Buffer.from(buffer)

Buffer.from(string[, encoding])

Buffer.alloc(size[, fill[, encoding]])

Buffer.allocUnsafe(size)

Buffer.allocUnsafeSlow(size)

Buffer.isBuffer(obj)

Buffer.isEncoding(encoding)

Buffer.byteLength(string[, encoding])

Buffer.compare(buf1, buf2)

Buffer.concat(list[, totalLength])

实例

buf.length

buf[index]

buf.swap16()

buf.swap32()

buf.swap64()

buf.readIntBE(offset, byteLength[, noAssert])

buf.readInt16BE(offset[, noAssert])

buf.readInt32BE(offset[, noAssert])

buf.readIntLE(offset, byteLength[, noAssert])

buf.readInt16LE(offset[, noAssert])

buf.readInt32LE(offset[, noAssert])

buf.readUInt16BE(offset[, noAssert])

buf.readUInt32BE(offset[, noAssert])

buf.readUIntBE(offset, byteLength[, noAssert])

buf.readUInt16LE(offset[, noAssert])

buf.readUInt32LE(offset[, noAssert])

buf.readUIntLE(offset, byteLength[, noAssert])

buf.writeIntBE(value, offset, byteLength[, noAssert])

buf.writeInt16BE(value, offset[, noAssert])

buf.writeInt32BE(value, offset[, noAssert])

buf.writeIntLE(value, offset, byteLength[, noAssert])

buf.writeInt16LE(value, offset[, noAssert])

buf.writeInt32LE(value, offset[, noAssert])

buf.writeUIntBE(value, offset, byteLength[, noAssert])

buf.writeUInt16BE(value, offset[, noAssert])

buf.writeUInt32BE(value, offset[, noAssert])

buf.writeUIntLE(value, offset, byteLength[, noAssert])

buf.writeUInt16LE(value, offset[, noAssert])

buf.writeUInt32LE(value, offset[, noAssert])

buf.readInt8(offset[, noAssert])

buf.readUInt8(offset[, noAssert])

buf.writeInt8(value, offset[, noAssert])

buf.writeUInt8(value, offset[, noAssert])

buf.readFloatBE(offset[, noAssert])

buf.readDoubleBE(offset[, noAssert])

buf.readFloatLE(offset[, noAssert])

buf.readDoubleLE(offset[, noAssert])

buf.writeFloatBE(value, offset[, noAssert])

buf.writeDoubleBE(value, offset[, noAssert])

buf.writeFloatLE(value, offset[, noAssert])

buf.writeDoubleLE(value, offset[, noAssert])

buf.toString([encoding[, start[, end]]])

buf.toJSON()

buf.indexOf(value[, byteOffset][, encoding])

buf.lastIndexOf(value[, byteOffset][, encoding])

buf.compare(target[, targetStart[, targetEnd[, sourceStart[,

sourceEnd]]]])

buf.copy(target[, targetStart[, sourceStart[, sourceEnd]]])

buf.entries()

buf.equals(otherBuffer)

buf.fill(value[, offset[, end]][, encoding])

buf.includes(value[, byteOffset][, encoding])

buf.keys()

buf.slice([start[, end]])

buf.values()

buf.write(string[, offset[, length]][, encoding])

3 Child Processes (子进程)

执文产异叉

消断关错退

标连管进出错

发断杀

child\_process.exec(command[, options][, callback])

child\_process.execFile(file[, args][, options][, callback])

child\_process.spawn(command[, args][, options])

child\_process.execFileSync(file[, args][, options])

child\_process.execSync(command[, options])

child\_process.spawnSync(command[, args][, options])

child\_process.fork(modulePath[, args][, options])

'message' 事件

'disconnect' 事件

'close' 事件

'error' 事件

'exit' 事件

child.pid

child.connected

child.stdio

child.stdin

child.stdout

child.stderr

child.send(message[, sendHandle[, options]][, callback])

child.disconnect()

child.kill([signal])

4 CLI (命令行选项)

版帮评印查互要 不赞警 追赞警同 扔零保轨专安能迫开vi

排路废替终额i

-v, --version

-h, --help

-e, --eval "script"

-p, --print "script"

-c, --check

-i, --interactive

-r, --require module

--no-deprecation

--no-warnings

--trace-deprecation

--trace-warnings

--trace-sync-io

--throw-deprecation

--zero-fill-buffers

--preserve-symlinks

--track-heap-objects

--prof-process

--tls-cipher-list=list

--enable-fips

--force-fips

--openssl-config=file

--v8-options

--icu-data-dir=file

环境变量

NODE\_DEBUG=module[,…]

NODE\_PATH=path[:…]

NODE\_DISABLE\_COLORS=1

NODE\_REPL\_HISTORY=file

NODE\_TTY\_UNSAFE\_ASYNC=1

NODE\_EXTRA\_CA\_CERTS=file

NODE\_ICU\_DATA=file

5 Console (控制台)

新时尾断目错信日追警

new Console(stdout[, stderr])

console.time(label)

console.timeEnd(label)

console.assert(value[, message][, ...args])

console.dir(obj[, options])

console.error([data][, ...args])

console.info([data][, ...args])

console.log([data][, ...args])

console.trace(message[, ...args])

console.warn([data][, ...args])

6 Error (错误)

错新捕堆 信堆

范引语类

Error 类

new Error(message)

Error.captureStackTrace(targetObject[, constructorOpt])

Error.stackTraceLimit

error.message

error.stack

RangeError 类

ReferenceError 类

SyntaxError 类

TypeError 类

7 Events (事件)

新移 默

设得 听数 加移所 预一 监一 触事

'newListener' 事件

'removeListener' 事件

EventEmitter.defaultMaxListeners

emitter.setMaxListeners(n)

emitter.getMaxListeners()

emitter.listeners(eventName)

emitter.listenerCount(eventName)

emitter.addListener(eventName, listener)

emitter.removeListener(eventName, listener)

emitter.removeAllListeners([eventName])

emitter.prependListener(eventName, listener)

emitter.prependOnceListener(eventName, listener)

emitter.on(eventName, listener)

emitter.once(eventName, listener)

emitter.emit(eventName[, ...args])

emitter.eventNames()

8 File System (文件系统)

观变错 关

状

读写 开关字路

异同读写 读文

异同空lf 改用访状

异同空f 缩戳

异同 通添开关存真重移写 f数同 连不符 建目临 读目连

常 创读写 看文不

fs.FSWatcher 类

'change' 事件

'error' 事件

watcher.close()

fs.Stats 类

Stat 时间值

fs.ReadStream 类

'open' 事件

'close' 事件

readStream.bytesRead

readStream.path

fs.WriteStream 类

'open' 事件

'close' 事件

writeStream.bytesWritten

writeStream.path

fs.read(fd, buffer, offset, length, position, callback)

fs.readFile(file[, options], callback)

fs.write(fd, buffer, offset, length[, position], callback)

fs.writeFile(file, data[, options], callback)

fs.readSync(fd, buffer, offset, length, position)

fs.readFileSync(file[, options])

fs.writeSync(fd, buffer, offset, length[, position])

fs.writeFileSync(file, data[, options])

fs.chown(path, uid, gid, callback)

fs.chmod(path, mode, callback)

fs.stat(path, callback)

fs.lchown(path, uid, gid, callback)

fs.lchmod(path, mode, callback)

fs.lstat(path, callback)

fs.fchown(fd, uid, gid, callback)

fs.fchmod(fd, mode, callback)

fs.fstat(fd, callback)

fs.chownSync(path, uid, gid)

fs.chmodSync(path, mode)

fs.statSync(path)

fs.lchownSync(path, uid, gid)

fs.lchmodSync(path, mode)

fs.lstatSync(path)

fs.fchownSync(fd, uid, gid)

fs.fchmodSync(fd, mode)

fs.fstatSync(fd)

fs.truncate(path, len, callback)

fs.utimes(path, atime, mtime, callback)

fs.ftruncate(fd, len, callback)

fs.futimes(fd, atime, mtime, callback)

fs.truncateSync(path, len)

fs.utimesSync(path, atime, mtime)

fs.ftruncateSync(fd, len)

fs.futimesSync(fd, atime, mtime)

fs.access(path[, mode], callback)

fs.appendFile(file, data[, options], callback)

fs.open(path, flags[, mode], callback)

fs.close(fd, callback)

fs.exists(path, callback)

fs.realpath(path[, options], callback)

fs.rename(oldPath, newPath, callback)

fs.rmdir(path, callback)

fs.write(fd, data[, position[, encoding]], callback)

fs.fdatasync(fd, callback)

fs.fsync(fd, callback)

fs.link(existingPath, newPath, callback)

fs.unlink(path, callback)

fs.symlink(target, path[, type], callback)

fs.mkdir(path[, mode], callback)

fs.mkdtemp(prefix[, options], callback)

fs.readdir(path[, options], callback)

fs.readlink(path[, options], callback)

fs.accessSync(path[, mode])

fs.appendFileSync(file, data[, options])

fs.openSync(path, flags[, mode])

fs.closeSync(fd)

fs.existsSync(path)

fs.realpathSync(path[, options])

fs.renameSync(oldPath, newPath)

fs.rmdirSync(path)

fs.writeSync(fd, data[, position[, encoding]])

fs.fdatasyncSync(fd)

fs.fsyncSync(fd)

fs.linkSync(existingPath, newPath)

fs.unlinkSync(path)

fs.symlinkSync(target, path[, type])

fs.mkdirSync(path[, mode])

fs.mkdtempSync(prefix[, options])

fs.readdirSync(path[, options])

fs.readlinkSync(path[, options])

fs.constants

fs.createReadStream(path[, options])

fs.createWriteStream(path[, options])

fs.watch(filename[, options][, listener])

fs.watchFile(filename[, options], listener)

fs.unwatchFile(filename[, listener])

9 Global (全局变量)

目文 设清立间尽 缓控全模进出 要藏解

\_\_dirname

\_\_filename

setImmediate(callback[, ...args])

setInterval(callback, delay[, ...args])

setTimeout(callback, delay[, ...args])

clearImmediate(immediateObject)

clearInterval(intervalObject)

clearTimeout(timeoutObject)

Buffer 类

console

global

module

process

exports

require()

require.cache

require.resolve()

10 HTTP

代新 套大免请 创毁得

客 连升 完终 继回套 完终尾刷写 设不套尽

服 连名升 检继期 客关请 大听时 听3尽关

响 关完 完头发 状码信 写继头 得设移头 添尾尽

进 终关 版方套地头追 生头追 状码信 毁尽

方状全 请得创

http.Agent 类

new Agent([options])

agent.sockets

agent.maxSockets

agent.freeSockets

agent.maxFreeSockets

agent.requests

agent.createConnection(options[, callback])

agent.destroy()

agent.getName(options)

http.ClientRequest 类

'connect' 事件

'upgrade' 事件

'aborted' 事件

'abort' 事件

'continue' 事件

'response' 事件

'socket' 事件

request.aborted

request.abort()

request.end([data][, encoding][, callback])

request.flushHeaders()

request.write(chunk[, encoding][, callback])

request.setNoDelay([noDelay])

request.setSocketKeepAlive([enable][, initialDelay])

request.setTimeout(timeout[, callback])

http.Server 类

'connect' 事件

'connection' 事件

'upgrade' 事件

'checkContinue' 事件

'checkExpectation' 事件

'clientError' 事件

'close' 事件

'request' 事件

server.maxHeadersCount

server.listening

server.timeout

server.listen(handle[, callback])

server.listen(path[, callback])

server.listen([port][, hostname][, backlog][, callback])

server.setTimeout(msecs, callback)

server.close([callback])

http.ServerResponse 类

'close' 事件

'finish' 事件

response.finished

response.headersSent

response.sendDate

response.statusCode

response.statusMessage

response.write(chunk[, encoding][, callback])

response.writeContinue()

response.writeHead(statusCode[, statusMessage][, headers])

response.getHeader(name)

response.setHeader(name, value)

response.removeHeader(name)

response.addTrailers(headers)

response.end([data][, encoding][, callback])

response.setTimeout(msecs, callback)

http.IncomingMessage 类

'aborted' 事件

'close' 事件

message.httpVersion

message.method

message.socket

message.url

message.headers

message.trailers

message.rawHeaders

message.rawTrailers

message.statusCode

message.statusMessage

message.destroy([error])

message.setTimeout(msecs, callback)

http.METHODS

http.STATUS\_CODES

http.globalAgent

http.request(options[, callback])

http.get(options[, callback])

http.createServer([requestListener])