Resource

Online
Official Website

Related
JavaScript

File

File System
fs.rename(oldPath, newPath, [callback])
fs.renameSync(oldPath, newPath)
fs.truncate(fd, len, [callback])
fs.truncateSync(fd, len)
fs.chown(path, uid, gid, [callback])
fs.chownSync(path, uid, gid)
fs.fchown(fd, uid, gid, [callback])
fs.fchownSync(fd, uid, gid)
fs.lchown(path, uid, gid, [callback])
fs.lchownSync(path, uid, gid)
fs.chmod(path, mode, [callback])
fs.chmodSync(path, mode)
fs.fchmod(fd, mode, [callback])
fs.fchmodSync(fd, mode)
fs.lchmod(path, mode, [callback])
fs.lchmodSync(path, mode)
fs.stat(path, [callback])
fs.lstat(path, [callback])
fs.fstat(fd, [callback])
fs.statSync(path)
fs.lstatSync(path)
fs.fstatSync(fd)
fs.link(srcpath, dstpath, [callback])
fs.linkSync(srcpath, dstpath)
fs.symlink(destination, path, [type], [callback])
fs.symlinkSync(destination, path, [type])
fs.readlink(path, [callback])
fs.readlinkSync(path)
fs.realpath(path, [cache], callback)
fs.realpathSync(path, [cache])
fs.unlink(path, [callback])
fs.unlinkSync(path)
fs.rmdir(path, [callback])
fs.rmdirSync(path)

Basic

Global Objects
global
process
console
Class: Buffer
require()
require.resolve()
require.cache
require.extensions
__filename
__dirname
module
exports
setTimeout(cb, ms)
clearTimeout(t)
setInterval(cb, ms)
clearInterval(t)

console
console.log([data], [...])
console.info([data], [...])
console.error([data], [...])
console.warn([data], [...])
console.dir(obj)
console.time(label)
console.timeEnd(label)
console.trace(label)
console.assert(expression, [message])

Timers
setTimeout(callback, delay, [arg], [...])
clearTimeout(timeoutId)
setInterval(callback, delay, [arg], [...])
clearInterval(intervalId)

Util
util.format(format, [...])
util.debug(string)
util.error([...])
util.puts([...])
util.print([...])
util.log(string)
util.inspect(object, [showHidden], [depth], [colors])
util.isArray(object)
util.isRegExp(object)
util.isDate(object)
util.isError(object)
util.pump(readableStream, writableStream, [callback])
util.inherits(constructor, superConstructor)

Module

Modules
Cycles
Core Modules
File Modules
Loading from `node_modules` Folders
Folders as Modules
Caching
Module Caching Caveats
The `module` Object
module.exports
module.require(id)
module.id
module.filename
module.loaded
module.parent
module.children
All Together...
Loading from the global folders
Accessing the main module
Addenda: Package Manager Tips

Addons
Addons
Hello world
Addon patterns
Function arguments
Callbacks
Object factory
Function factory
Wrapping C++ objects
Factory of wrapped objects
Passing wrapped objects around

Net

net
net.createServer([options], [connectionListener])
net.connect(options, [connectionListener])
net.createConnection(options, [connectionListener])
net.connect(port, [host], [connectListener])
net.createConnection(port, [host], [connectListener])
net.connect(path, [connectListener])

Buffer/Stream

Buffer
Buffer
Class: Buffer
new Buffer(size)
new Buffer(array)
new Buffer(str, [encoding])
buf.write(string, [offset], [length], [encoding])
buf.toString([encoding], [start], [end])
buf[index]
Class Method: Buffer.isBuffer(obj)
Class Method: Buffer.byteLength(string, [encoding])
Class Method: Buffer.concat(list, [totalLength])
buf.length
buf.copy(targetBuffer, [targetStart], [sourceStart], [sourceEnd])
buf.slice([start], [end])
buf.readUInt8(offset, [noAssert])
buf.readUInt16LE(offset, [noAssert])
buf.readUInt16BE(offset, [noAssert])
buf.readUInt32LE(offset, [noAssert])
buf.readUInt32BE(offset, [noAssert])
buf.readInt8(offset, [noAssert])
buf.readInt16LE(offset, [noAssert])
buf.readInt16BE(offset, [noAssert])
buf.readInt32LE(offset, [noAssert])
buf.readInt32BE(offset, [noAssert])
buf.readFloatLE(offset, [noAssert])
buf.readFloatBE(offset, [noAssert])
buf.readDoubleLE(offset, [noAssert])
buf.readDoubleBE(offset, [noAssert])
buf.writeUInt8(value, offset, [noAssert])
buf.writeUInt16LE(value, offset, [noAssert])
buf.writeUInt16BE(value, offset, [noAssert])
buf.writeUInt32LE(value, offset, [noAssert])

Process/Events

Process
Event: 'exit'
Event: 'uncaughtException'
Signal Events
process.stdout
process.stderr
process.stdin
process.argv
process.execPath
process.abort()
process.chdir(directory)
process.cwd()
process.env
process.exit([code])
process.getgid()
process.setgid(id)
process.getuid()
process.setuid(id)
process.version
process.versions
process.config
process.kill(pid, [signal])
process.pid
process.title
process.arch
process.platform
process.memoryUsage()
process.nextTick(callback)
process.umask([mask])
process.uptime()
process.hrtime()

Events
Class: events.EventEmitter
emitter.addListener(event, listener)
emitter.on(event, listener)
emitter.once(event, listener)
emitter.removeListener(event, listener)
emitter.removeAllListeners([event])
emitter.setMaxListeners(n)
emitter.listeners(event)
emitter.emit(event, [arg1], [arg2], [...])
Event: 'newListener'

Text

Path

Domain

Domain
Additions to Error objects
Implicit Binding
Explicit Binding
domain.create()
Class: Domain
domain.run(fn)
domain.members
domain.add(emitter)
domain.remove(emitter)
domain.bind(cb)
Example
domain.intercept(cb)
Example
domain.dispose()

Crypto
crypto.createCredentials(details)
crypto.createHash(algorithm)
Class: Hash
hash.update(data, [input_encoding])
hash.digest([encoding])
crypto.createHmac(algorithm, key)
Class: Hmac
hmac.update(data)
hmac.digest([encoding])
crypto.createCipher(algorithm, password)
crypto.createCipheriv(algorithm, key, iv)
Class: Cipher
cipher.update(data, [input_encoding], [output_encoding])
cipher.final([output_encoding])
cipher.setAutoPadding(auto_padding=true)
crypto.createDecipher(algorithm, password)
crypto.createDecipheriv(algorithm, key, iv)
Class: Decipher
decipher.update(data, [input_encoding], [output_encoding])
decipher.final([output_encoding])
decipher.setAutoPadding(auto_padding=true)
crypto.createSign(algorithm)
Class: Signer
signer.update(data)
signer.sign(private_key, [output_format])
crypto.createVerify(algorithm)

fs.mkdir(path, [mode], [callback])
fs.mkdirSync(path, [mode])
fs.readdir(path, [callback])
fs.readdirSync(path)
fs.close(fd, [callback])
fs.closeSync(fd)
fs.open(path, flags, [mode], [callback])
fs.openSync(path, flags, [mode])
fs.utimes(path, atime, mtime, [callback])
fs.utimesSync(path, atime, mtime)
fs.futimes(fd, atime, mtime, [callback])
fs.futimesSync(fd, atime, mtime)
fs.fsync(fd, [callback])
fs.fsyncSync(fd)
fs.write(fd, buffer, offset, length, position, [callback])
fs.writeSync(fd, buffer, offset, length, position)
fs.read(fd, buffer, offset, length, position, [callback])
fs.readSync(fd, buffer, offset, length, position)
fs.readFile(filename, [encoding], [callback])
fs.readFileSync(filename, [encoding])
fs.writeFile(filename, data, [encoding], [callback])
fs.writeFileSync(filename, data, [encoding])
fs.appendFile(filename, data, encoding='utf8', [callback])
fs.appendFileSync(filename, data, encoding='utf8')
fs.watchFile(filename, [options], listener)
fs.unwatchFile(filename)
fs.watch(filename, [options], [listener])
Caveats
Availability
Filename Argument
fs.exists(path, [callback])
fs.existsSync(path)
Class: fs.Stats
fs.createReadStream(path, [options])
Class: fs.ReadStream
Event: 'open'
fs.createWriteStream(path, [options])
fs.WriteStream
Event: 'open'
file.bytesWritten
Class: fs.FSWatcher
watcher.close()
Event: 'change'

HTTP

http

http.STATUS_CODES
http.createServer([requestListener])
http.createClient([port], [host])
http.request(options, callback)
http.get(options, callback)
http.globalAgent

Class: http.Server

Event: 'request'
Event: 'connection'
Event: 'close'
Event: 'checkContinue'
Event: 'connect'
Event: 'upgrade'
Event: 'clientError'
server.listen(port, [hostname], [backlog], [callback])
server.listen(path, [callback])
server.listen(handle, [listeningListener])
server.close([cb])
server.maxHeadersCount

Class: http.ServerRequest

Event: 'data'
Event: 'end'
Event: 'close'
request.method
request.url
request.headers
request.trailers
request.httpVersion
request.setEncoding([encoding])
request.pause()
request.resume()
request.connection

Class: http.ServerResponse

Event: 'close'
response.writeContinue()
response.writeHead(statusCode, [reasonPhrase], [headers])
response.statusCode
response.setHeader(name, value)
response.sendDate
response.getHeader(name)
response.removeHeader(name)
response.write(chunk, [encoding])

net.createConnection(path, [connectListener])
Class: net.Server
server.listen(port, [host], [backlog], [listeningListener])
server.listen(path, [listeningListener])
server.listen(handle, [listeningListener])
server.close([cb])
server.address()
server.maxConnections
server.connections
Event: 'listening'
Event: 'connection'
Event: 'close'
Event: 'error'
Class: net.Socket
new net.Socket([options])
socket.connect(port, [host], [connectListener])
socket.connect(path, [connectListener])
socket.bufferSize
socket.setEncoding([encoding])
socket.write(data, [encoding], [callback])
socket.end([data], [encoding])
socket.destroy()
socket.pause()
socket.resume()
socket.setTimeout(timeout, [callback])
socket.setNoDelay([noDelay])
socket.setKeepAlive([enable], [initialDelay])
socket.address()
socket.remoteAddress
socket.remotePort
socket.bytesRead
socket.bytesWritten
Event: 'connect'
Event: 'data'
Event: 'end'
Event: 'timeout'
Event: 'drain'
Event: 'error'
Event: 'close'
net.isIP(input)
net.isIPv4(input)
net.isIPv6(input)

UDP / Datagram Sockets

dgram.createSocket(type, [callback])
Class: Socket
Event: 'message'
Event: 'listening'
Event: 'close'
Event: 'error'
dgram.send(buf, offset, length, port, address, [callback])
dgram.bind(port, [address])

buf.writeUInt32BE(value, offset, [noAssert])
buf.writeInt8(value, offset, [noAssert])
buf.writeInt16LE(value, offset, [noAssert])
buf.writeInt16BE(value, offset, [noAssert])
buf.writeInt32LE(value, offset, [noAssert])
buf.writeInt32BE(value, offset, [noAssert])
buf.writeFloatLE(value, offset, [noAssert])
buf.writeFloatBE(value, offset, [noAssert])
buf.writeDoubleLE(value, offset, [noAssert])
buf.writeDoubleBE(value, offset, [noAssert])
buf.fill(value, [offset], [end])
buffer.INSPECT_MAX_BYTES
Class: SlowBuffer

Stream

Readable Stream
Event: 'data'
Event: 'end'
Event: 'error'
Event: 'close'
stream.readable
stream.setEncoding([encoding])
stream.pause()
stream.resume()
stream.destroy()
stream.pipe(destination, [options])
Writable Stream
Event: 'drain'
Event: 'error'
Event: 'close'
Event: 'pipe'
stream.writable
stream.write(string, [encoding], [fd])
stream.write(buffer)
stream.end()
stream.end(string, encoding)
stream.end(buffer)
stream.destroy()
stream.destroySoon()

TTY

tty.isatty(fd)
tty.setRawMode(mode)
Class: ReadStream
rs.isRaw
rs.setRawMode(mode)
Class WriteStream
ws.columns
ws.rows
Event: 'resize'

path.normalize(p)
path.join([path1], [path2], [...])
path.resolve([from ...], to)
path.relative(from, to)
path.dirname(p)
path.basename(p, [ext])
path.extname(p)
path.sep

Query String

querystring.stringify(obj, [sep], [eq])
querystring.parse(str, [sep], [eq], [options])
querystring.escape
querystring.unescape

punnycode

punycode.decode(string)
punycode.encode(string)
punycode.toUnicode(domain)
punycode.toASCII(domain)
punycode.ucs2
punycode.ucs2.decode(string)
punycode.ucs2.encode(codePoints)
punycode.version

Readline

readline.createInterface(options)
Class: Interface
rl.setPrompt(prompt, length)
rl.prompt([preserveCursor])
rl.question(query, callback)
rl.pause()
rl.resume()
rl.close()
rl.write(data, [key])
Events
Event: 'line'
Event: 'pause'
Event: 'resume'
Event: 'close'
Event: 'SIGINT'
Event: 'SIGTSTP'
Event: 'SIGCONT'
Example: Tiny CLI

REPL

repl.start(options)
Event: 'exit'
REPL Features

Code

Executing JS

Class: Verify
verifier.update(data)
verifier.verify(object, signature, [signature_format])
crypto.createDiffieHellman(prime_length)
crypto.createDiffieHellman(prime, [encoding])
Class: DiffieHellman
diffieHellman.generateKeys([encoding])
diffieHellman.computeSecret(other_public_key, [input_encoding], [output_encoding])
diffieHellman.getPrime([encoding])
diffieHellman.getGenerator([encoding])
diffieHellman.getPublicKey([encoding])
diffieHellman.getPrivateKey([encoding])
diffieHellman.setPublicKey(public_key, [encoding])
diffieHellman.setPrivateKey(public_key, [encoding])
crypto.getDiffieHellman(group_name)
crypto.pbkdf2(password, salt, iterations, keylen, callback)
crypto.randomBytes(size, [callback])

TSL(SSL)

Client-initiated renegotiation attack mitigation
NPN and SNI
tls.createServer(options, [secureConnectionListener])
tls.connect(options, [secureConnectListener])
tls.connect(port, [host], [options], [secureConnectListener])
tls.createSecurePair([credentials], [isServer], [requestCert], [rejectUnauthorized])
Class: SecurePair
Event: 'secure'
Class: tls.Server
Event: 'secureConnection'
Event: 'clientError'
server.listen(port, [host], [callback])
server.close()
server.address()
server.addContext(hostname, credentials)
server.maxConnections
server.connections
Class: tls.CleartextStream
Event: 'secureConnect'
cleartextStream.authorized

Event: 'error'

3rd Party

Third Party Modules

Module Installer:
npm
HTTP Middleware:
Connect
Web Framework:
Express
Web Sockets:
Socket.IO
HTML Parsing:
HTML5
mDNS/Zeroconf/Bonjour
/li>
RabbitMQ, AMQP
mysql
Serialization:
msgpack
Scraping:
Apricot
Debugger:
ndb
is a CLI debugger
inspector
is a web based tool.
pcap binding
ncurses
Testing/TDD/BDD:
vows
,
mocha
,
mjsunit.runner

response.addTrailers(headers)
response.end([data], [encoding])

Class: http.Agent

agent.maxSockets
agent.sockets
agent.requests

Class: http.ClientRequest

Event 'response'
Event: 'socket'
Event: 'connect'
Event: 'upgrade'
Event: 'continue'
request.write(chunk, [encoding])
request.end([data], [encoding])
request.abort()
request.setTimeout(timeout, [callback])
request.setNoDelay([noDelay])
request.setSocketKeepAlive([enable], [initialDelay])

http.ClientResponse

Event: 'data'
Event: 'end'
Event: 'close'
response.statusCode
response.httpVersion
response.headers
response.trailers
response.setEncoding([encoding])
response.pause()
response.resume()

HTTPS

Class: https.Server
https.createServer(options, [requestListener])
https.request(options, callback)
https.get(options, callback)
Class: https.Agent
https.globalAgent

URL

url.parse(urlStr, [parseQueryString], [slashesDenoteHost])
url.format(urlObj)
url.resolve(from, to)

dgram.close()
dgram.address()
dgram.setBroadcast(flag)
dgram.setTTL(ttl)
dgram.setMulticastTTL(ttl)
dgram.setMulticastLoopback(flag)
dgram.addMembership(multicastAddress, [multicastInterface])
dgram.dropMembership(multicastAddress, [multicastInterface])

DNS

dns.lookup(domain, [family], callback)
dns.resolve(domain, [rrtype], callback)
dns.resolve4(domain, callback)
dns.resolve6(domain, callback)
dns.resolveMx(domain, callback)
dns.resolveTxt(domain, callback)
dns.resolveSrv(domain, callback)
dns.resolveNs(domain, callback)
dns.resolveCname(domain, callback)
dns.reverse(ip, callback)
Error codes

System

Zlib

Examples
zlib.createGzip([options])
zlib.createGunzip([options])
zlib.createDeflate([options])
zlib.createInflate([options])
zlib.createDeflateRaw([options])
zlib.createInflateRaw([options])
zlib.createUnzip([options])
Class: zlib.Gzip
Class: zlib.Gunzip
Class: zlib.Deflate
Class: zlib.Inflate
Class: zlib.DeflateRaw
Class: zlib.InflateRaw
Class: zlib.Unzip
Convenience Methods
zlib.deflate(buf, callback)
zlib.deflateRaw(buf, callback)
zlib.gzip(buf, callback)
zlib.gunzip(buf, callback)
zlib.inflate(buf, callback)
zlib.inflateRaw(buf, callback)
zlib.unzip(buf, callback)
Options
Memory Usage Tuning
Constants

os

os.tmpDir()
os.hostname()
os.type()
os.platform()
os.arch()
os.release()
os.uptime()
os.loadavg()
os.totalmem()
os.freemem()
os.cpus()
os.networkInterfaces()
os.EOL

Debugger

Watchers
Commands reference
Stepping
Breakpoints
Info
Execution control
Various
Advanced Usage

Caveats
Sandboxes
Globals
vm.runInThisContext(code, [filename])
vm.runInNewContext(code, [sandbox], [filename])
vm.runInContext(code, context, [filename])
vm.createContext([initSandbox])
vm.createScript(code, [filename])
Class: Script
script.runInThisContext()
script.runInNewContext([sandbox])

Child Process

Class: ChildProcess
Event: 'exit'
Event: 'close'
Event: 'disconnect'
Event: 'message'
child.stdin
child.stdout
child.stderr
child.pid
child.kill([signal])
child.send(message, [sendHandle])
child.disconnect()
child_process.spawn(command, [args], [options])
child_process.exec(command, [options], callback)
child_process.execFile(file, args, options, callback)
child_process.fork(modulePath, [args], [options])

Assert

assert.fail(actual, expected, message, operator)
assert(value, message), assert.ok(value, [message])
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.throws(block, [error], [message])
assert.doesNotThrow(block, [error], [message])
assert.ifError(value)

cleartextStream.authorizationError
cleartextStream.getPeerCertificate()
cleartextStream.getCipher()
cleartextStream.address()
cleartextStream.remoteAddress
cleartextStream.remotePort

StringDecoder

Class: StringDecoder
StringDecoder.write(buffer)

Cluster

How It Works
cluster.settings
cluster.isMaster
cluster.isWorker
Event: 'fork'
Event: 'online'
Event: 'listening'
Event: 'disconnect'
Event: 'exit'
Event: 'setup'
cluster.setupMaster([settings])
cluster.fork([env])
cluster.settings
cluster.disconnect([callback])
cluster.workers
Class: Worker
worker.id
worker.process
worker.suicide
worker.send(message, [sendHandle])
worker.destroy()
worker.disconnect()
Event: 'message'
Event: 'online'
Event: 'listening'
Event: 'disconnect'
Event: 'exit'