Assertion testing

断言模式

严格断言模式	require('node:assert/strict')
传统断言模式	require('node:assert')

断言错误

类	new assert.AssertionError(options)
错误类型	name
错误信息	message
真实值	actual
期望值	expect
ERR_ASSERTION	code
断言操作	operator
是否自动生成信息	generatedMessage

断言操作

真值	ok
相等	equal(actual, expected[, message])
严格相等	strictEqual(actual, expected[, message])
深层相等	deepEqual(actual, expected[, message])
深层严格相等	deepStrictEqual(actual, expected[, message])
不相等	notEqual(actual, expected[, message])
不严格相等	notStrictEqual(actual, expected[, message])
不深层相等	notDeepEqual(actual, expected[, message])
不深层严格相等	notDeepStrictEqual(actual, expected[, message])
正则匹配	match(string, regexp[, message])
正则不匹配	doesNotMatch(string, regexp[, message])
异步函数拒绝	rejects(asyncFn[, error][, message])
异步函数非拒绝	doesNotReject(asyncFn[, error][, message])
函数抛出错误	throws(fn[, error][, message])
函数不抛出错误	doesNotThrow(fn[, error][, message])

断言操作 (cont)

期望undefined和null	ifError(value)
抛出错误	fail([message])

Buffer

编码格式

character	utf-8
	utf16le
	latin1
binary-to-text	base64
	base64url
	hex
legacy character	ascii
	binary
	ucs2

Blob

类	new buffer.Blob([sources[, options]])
返回 Promise 包含数据的 ArrayBuffer	blob.arrayBuffer()
数据字节大小	blob.size
截取	blob.slice([start[, end[, type]]])
返回 ReadableStream	blob.stream()
解码成字符串 utf-8格式	blob.text()
数据类型	blob.type

Buffer

类	Buffer
静态方法	
创建对象	Buffer.alloc(size[, fill[, encoding]])
创建对象-不初始化	Buffer.allocUnsafe(size)
创建对象-不初始化-慢速路径	Buffer.allocUnsafeSlow(size)
返回字符串的字节长度	Buffer.byteLength(string[, encoding])
比较字节长度	Buffer.compare(buf1, buf2)
连接字节数组	Buffer.concat(list[, totalLength])

Buffer (cont)	
复制字节数组	Buffer.copyBytesFrom(view[, offset[, length]])
创建对象-纯数值数组	Buffer.from(array)
创建对象-共享 arrayBuffer 内存	Buffer.from(arrayBuffer[, byteOffset[, length]])
创建对象-复制 buffer	Buffer.from(buffer)
创建对象-某些特定对象如 Array	Buffer.from(object[, offsetOrEncoding[, length]])
创建对象-字符串	Buffer.from(string[, encoding])
判断是否为 Buffer 对象	Buffer.isBuffer(obj)
判断 Buffer 是否支持该编码	Buffer.isEncoding(encoding)
Node.js 内部 Buffer 对象池的大小	Buffer.poolSize
实例对象	
指定索引处的字节值	buf.buf[index]
与 Buffer 共享的 ArrayBuffer 对象	buf.buffer
Buffer 对象的字节长度	buf.length
Buffer 在 ArrayBuffer 中的偏移量	buf.byteOffset
与目标 Buffer 或 ArrayBuffer比较	buf.compare(target[, targetStart[, targetEnd[, sourceStart[, sourceEnd]]]])
复制到目标 Buffer	buf.copy(target[, targetStart[, sourceStart[, sourceEnd]]])
填充	buf.fill(value[, offset[, end]][, encoding])
是否包含指定的值	buf.includes(value[, byteOffset][, encoding])
指定值在 Buffer 中第一次出现的索引	buf.indexOf(value[, byteOffset][, encoding])
截取数组	buf.subarray([start[, end]])
每16字节 字节翻转顺序	buf.swap16()
每32字节 字节翻转顺序	buf.swap32()
每64字节 字节翻转顺序	buf.swap64()

Buffer (cont)	
读取数据（BE大端 LE小端）	
对象变 JSON 字符串	buf.toJSON()
解码成字符串	buf.toString([encoding[, start[, end]]])
8 位有符号整数	buf.readInt8([offset])
16 位有符号整数	buf.readInt16BE([offset])
16 位有符号整数	buf.readInt16LE([offset])
32 位有符号整数	buf.readInt32BE([offset])
32 位有符号整数	buf.readInt32LE([offset])
64 位有符号整数	buf.readBigInt64BE([offset])
64 位有符号整数	buf.readBigInt64LE([offset])
xx 位的有符号整数	buf.readIntBE(offset, byteLength)
xx 位的有符号整数	buf.readIntLE(offset, byteLength)
64 位的无符号整数	buf.readBigUInt64BE([offset])
64 位的无符号整数	buf.readBigUInt64LE([offset])
8 位无符号整数	buf.readUInt8([offset])
16 位无符号整数	buf.readUInt16BE([offset])
16 位无符号整数	buf.readUInt16LE([offset])
32 位无符号整数	buf.readUInt32BE([offset])
32 位无符号整数	buf.readUInt32LE([offset])
xx 位无符号整数	buf.readUIntBE(offset, byteLength)
xx 位无符号整数	buf.readUIntLE(offset, byteLength)
32 位的单精度浮点数	buf.readFloatBE([offset])
32 位的单精度浮点数	buf.readFloatLE([offset])
64 位的双精度浮点数	buf.readDoubleBE([offset])
64 位的双精度浮点数	buf.readDoubleLE([offset])
写入数据（BE大端 LE小端）	
字符串	buf.write(string[, offset[, length]][, encoding])
8 位有符号整数	buf.writeInt8(value[, offset])
16 位有符号整数	buf.writeInt16BE(value[, offset])
16 位有符号整数	buf.writeInt16LE(value[, offset])

Buffer (cont)	
32 位有符号整数	buf.writeInt32BE(value[, offset])
32 位有符号整数	buf.writeInt32LE(value[, offset])
64 位有符号整数	buf.writeBigInt64BE(value[, offset])
64 位有符号整数	buf.writeBigInt64LE(value[, offset])
xx 位有符号整数	buf.writeIntBE(value, offset, byteLength)
xx 位有符号整数	buf.writeIntLE(value, offset, byteLength)
8 位无符号整数	buf.writeUInt8(value[, offset])
16 位无符号整数	buf.writeUInt16BE(value[, offset])
16 位无符号整数	buf.writeUInt16LE(value[, offset])
32 位无符号整数	buf.writeUInt32BE(value[, offset])
32 位无符号整数	buf.writeUInt32LE(value[, offset])
64 位无符号整数	buf.writeBigUInt64BE(value[, offset])
64 位无符号整数	buf.writeBigUInt64LE(value[, offset])
xx 位无符号整数	buf.writeUIntBE(value, offset, byteLength)
xx 位无符号整数	buf.writeUIntLE(value, offset, byteLength)
32 位的单精度浮点数	buf.writeFloatBE(value[, offset])
32 位的单精度浮点数	buf.writeFloatLE(value[, offset])
64 位的双精度浮点数	buf.writeDoubleBE(value[, offset])
64 位的双精度浮点数	buf.writeDoubleLE(value[, offset])

Buffer 迭代
for..of
buf.values()
buf.keys()
buf.entries()

File	
类	new buffer.File(sources, fileName[, options])
文件名	file.name
文件上次修改时间	file.lastModified

buffer 模块 API	
判断输入的数据是否为 ASCII 编码	buffer.isAscii(input)
判断输入的数据是否为 UTF-8 编码	buffer.isUtf8(input)
限制 Buffer 对象在 调试输出等 时显示的最大字节数	buffer.INSPECT_MAX-_BYTES
Buffer 对象的最大长度	buffer.kMaxLength
Buffer 对象的最大长度	buffer.constants.MAX_-LENGTH
字符串的最大长度	buffer.kStringMaxLength
字符串的最大长度	buffer.constants.MAX_STR-ING_LENGTH
将二进制数据从一个字符编码转换为另一个字符编码	buffer.transcode(source, fromEnc, toEnc)