

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

Buffer (cont)

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

Buffer (cont)

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

Buffer 迭代

`for..of`
`buf.values()`
`buf.keys()`
`buf.entries()`

File

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

buffer 模块 API

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



By **Ran Qing (Ran Qing)**
cheatography.com/ran-qing/

Not published yet.
 Last updated 6th December, 2023.
 Page 3 of 3.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>