#node.js-Buffe

创建Buffer

下列API来创建Buffer类：

序号	方法
Buffer.alloc(size,[,fill[,encoding]])	返回一个指定大小的Buffer实例，如果没有设置fill，则默认为0
Buffer.allocUnsafe(size)	返回一个指定大小的Buffer实例，但是他不会被初始化，所以他可能包含敏感的数据
Buffer.allocUnsafeSlow(size)	
Buffer.from(array)	返回一个被array的值初始化的新的Buffer实例（传入的array的元素之恶能是数字，不然就会被自动0覆盖）
Buffer.from(arrayBuffer[,byteOffset[,length]])	返回一个新建的与给定的 ArrayBuffer共享同一内存的Buffer。
Buffer.from(buffer)	复制传入的Buffer实例的数据，并返回一个新Buffer实例
Buffer，from(string[,encoding])	返回一个被Striung的值初始化的新的Buffer实例

实例

```
// 创建一个长度为 10、且用 0 填充的 Buffer。
const buf1 = Buffer.alloc(10);

// 创建一个长度为 10、且用 0x1 填充的 Buffer。
const buf2 = Buffer.alloc(10, 1);

// 创建一个长度为 10、且未初始化的 Buffer。
// 这个方法比调用 Buffer.alloc() 更快，
// 但返回的 Buffer 实例可能包含旧数据，
// 因此需要使用 fill() 或 write() 重写。
const buf3 = Buffer.allocUnsafe(10);

// 创建一个包含 [0x1, 0x2, 0x3] 的 Buffer。
const buf4 = Buffer.from([1, 2, 3]);

// 创建一个包含 UTF-8 字节 [0x74, 0xc3, 0xa9, 0x73, 0x74] 的 Buffer。
const buf5 = Buffer.from('tést');

// 创建一个包含 Latin-1 字节 [0x74, 0xe9, 0x73, 0x74] 的 Buffer。
const buf6 = Buffer.from('tést', 'latin1');
```

写入缓冲区

语法

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

参数

序号	参数	描述
1	string	写入缓冲区的字符串
2	offset	缓冲区开始写入的索引值，默认为
3	length	写入的字节数，默认为buffer length
4	encoding	使用的编码，默认为'utf8'

如果buf没有足够的空间保存整个字符串，则只会写入string的一部分。

返回值

返回实际写入的大小。如果buffer空间不足，则只会写入部分字符串

实例

```
buf = Buffer.alloc(256);
len = buf.write('www.runoob.com');

console.log("写入字节数" + len）;
```

从缓冲区读取数据

语法

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

参数

序号	参数	描述
1	encoding	使用的编码。默认为'utf8'
2	start	指定开始读取的索引位置.默认为0
3	end	结束位置。默认为缓冲区的末尾

返回值

解码缓冲区数据并使用指定的编码返回字符串

实例

```
buf = Buffer.alloc(26);
for(var i = 0 ; i < 26 ; i++) {
  buf[i] = i + 97;
}

console.log( buf.toString('ascii'));       // 输出: abcdefghijklmnopqrstuvwxyz
```

```
console.log( buf.toString('ascii',0,5));   // 输出: abcde
console.log( buf.toString('utf8',0,5));    // 输出: abcde
console.log( buf.toString(undefined,0,5)); // 使用 'utf8' 编码, 并输出: abcde
```

将Buffer转换为JSON对象

语法

buf.toJOSN()
当字符串化为一个Buffer实例时，JSON.stringify()会隐式调用该toJSON()。

返回值

返回JSON对象

实例

```
const buf = Buffer.from([0x1, 0x2, 0x3, 0x4, 0x5]);
const json = JSON.stringify(buf);

// 输出: {"type":"Buffer","data":[1,2,3,4,5]}
console.log(json);

const copy = JSON.parse(json,(key,value) = > {
  return value && value.type === 'Buffer' ? Buffer.from(value.data) : value;
});

// 输出: <Buffer 01 02 03 04 05>
console.log(copy);
```

缓冲区合并

语法

Buffer.concat(list[,totalLength])

参数

序号	参数	描述
1	list	用于合并Buffer对象数组列表
2	totalLength	指定合并后Buffer对象的长度

返回值

返回一个多个成员合并的新Buffer对象

实例

```
var buffer1 = Buffer.from('阮涵州');
var buffer2 = Buffer.from('rhz');
var buffer3 = Buffer.concat([Buffer1,Buffer2]);

```

```
5
    console.log("buffer3 内容:" + budder3.toString());
```

缓冲区比较

语法

buf.compare(otherBuffer);

参数

序号	参数	描述
1	otherBuffer	比较的另一个对象

返回值

返回一个数字，表示buf与otherBuffer之前，之后或相同

实例

```
1  var buf1 = Buffer.from('ABC');
2  var buf2 = Buffer.from('ABCD');
3
4  var result = buf1.compare(buf2);
5
6  if(result < 0) {
7     console.log(buffer1 + " 在 " + buffer2 + "之前");
8  }else if(result == 0){
9     console.log(buffer1 + " 与 " + buffer2 + "相同");
10 }else {
11    console.log(buffer1 + " 在 " + buffer2 + "之后");
12 }
```

拷贝缓冲区

语法

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

参数

序号	参数	描述
1	targetBuffer	目标Buffer对象
2	targetStart	目标Buffer的覆盖起始位置，默认0
3	sourceStart	源Buffer需要拷贝的起始位置，默认0
4	sourceEnd	源Buffer拷贝的结束位置，默认buf.length

返回值

没有返回值

实例

```
var buf1 = Buffer.from('abcdefghijkl');
var buf2 = Buffer.from('RUNOOB');

//将 buf2 插入到 buf1 指定位置上
buf2.copy(buf1, 2);

console.log(buf1.toString());
```

缓冲区裁剪

语法

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

参数

序号	参数	描述
1	start	裁剪起始位置,默认0
2	end	裁剪结束位置，默认buf.length

返回值

返回一个新的缓冲区，他和旧缓冲区指向同一块内存，但是从索引start到end的位置剪切

实例

```
var buffer1 = Buffer.from('runoob');
// 剪切缓冲区
var buffer2 = buffer1.slice(0,2);
//输出结果"buffer2 content: ru"
console.log("buffer2 content: " + buffer2.toString());
```

缓冲区长度

语法

buf.length;

返回值

返回Buffer对象所占据的内存长度。

实例

```
var buffer = Buffer.from('www.runoob.com');
//  缓冲区长度
console.log("buffer length: " + buffer.length);
```