1. 什么是Buffer

- 缓冲区Buffer是暂时存放输入输出数据的一段内存。
- JS语言没有二进制数据类型，而在处理TCP和文件流的时候，必须要处理二进制数据。
- NodeJS提供了一个Buffer对象来提供对二进制数据的操作
- 是一个表示固定内存分配的全局对象，也就是说要放到缓存区中的字节数需要提前确定
- Buffer好比由一个多位字节元素组成的数组，可以有效的在javascript中表示二进制数据

2. 什么是字节

- 字节(Byte)是计算机存储时的一种计量单位，一个字节等于8位二进制数
- 一个位就代表一个0或1，每8个位（bit）组成一个字节（Byte）
- 字节是通过网络传输信息的单位
- 一个字节最大值十进制数是255

```
var sum =0;
for(var i=0;i<8;i++){
```

```
    sum += Math.pow(2,i);
}
```

3. 定义buffer的三种方式

3.1 通过长度定义buffer

```
new Buffer(size);
```

3.2 通过数组定义buffer

```
new Buffer(array);
```

> 正常情况下为0-255之间;

3.3 字符串创建

```
new Buffer(str,[encoding]);
```

4.buffer常用方法

4.1 fill方法

手动初始化,擦干净桌子,将buffer内容清0

```
buffer.fill(0);
```

4.2 write方法

string, offset, length, encoding

```
buffer.write('珠',0,3,'utf8');
buffer.write('峰',3,3,'utf8'); //珠峰
```

4.3 toString方法

将buffer转换成字符串类型 start end 是截取的buffer的长度

```
buffer.toString('utf8',3,6)
```

4.4 slice方法

```
buffer.slice(0,4);
```

截取乱码问题

```
var StringDecoder  = require('string_decoder').StringDecoder;
var sd = new StringDecoder;
var buffer = new Buffer('珠峰');
console.log(sd.write(buffer.slice(0,4)));
console.log(sd.write(buffer.slice(4)));
```

4.5 copy方法

复制Buffer 把多个buffer拷贝到一个大buffer上

```
sourceBuffer.copy(targetBuffer,targetstart,sourcestart,sourceend);
```

4.6 concat方法

```
Buffer.concat([buf1,buf2],length);
```

> 实现concat方法

4.7 isBuffer

判断是否是buffer

```
Buffer.isBuffer
```

4.8 length

获取字节长度(显示是字符串所代表buffer的长度)

```
Buffer.byteLength("珠峰");
buffer.length;
```

5.进制转换

- 将任意进制字符串转换为十进制
  - parseInt("11", 2); // 3 2进制转10进制
  - parseInt("77", 8); // 63 8进制转10进制
  - parseInt("e7", 16); //175 16进制转10进制
- 将10进制转换为其它进制字符串
  - (3).toString(2)) // "11" 十进制转2进制
  - (17).toString(16) // "11" 十进制转16进制
  - (33).toString(32) // "11" 十提制转32进制

> base64的转换