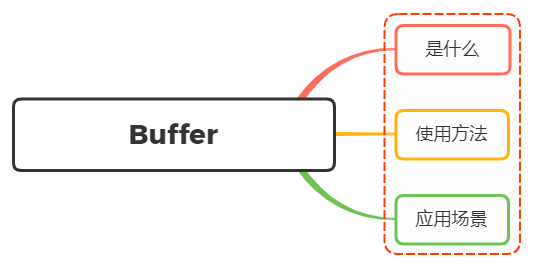
# 面试官：说说对 Node 中的 Buffer 的理解？应用场景？



## 一、是什么

在Node应用中，需要处理网络协议、操作数据库、处理图片、接收上传文件等，在网络流和文件的操作中，要处理大量二进制数据，而Buffer就是在内存中开辟一片区域（初次初始化为8KB），用来存放二进制数据

在上述操作中都会存在数据流动，每个数据流动的过程中，都会有一个最小或最大数据量

如果数据到达的速度比进程消耗的速度快，那么少数早到达的数据会处于等待区等候被处理。反之，如果数据到达的速度比进程消耗的数据慢，那么早先到达的数据需要等待一定量的数据到达之后才能被处理

这里的等待区就指的缓冲区（Buffer），它是计算机中的一个小物理单位，通常位于计算机的 RAM 中

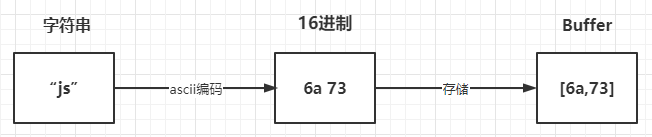
简单来讲，Nodejs不能控制数据传输的速度和到达时间，只能决定何时发送数据，如果还没到发送时间，则将数据放在Buffer中，即在RAM中，直至将它们发送完毕

上面讲到了Buffer是用来存储二进制数据，其的形式可以理解成一个数组，数组中的每一项，都可以保存8位二进制：00000000，也就是一个字节

例如：

const buffer = Buffer.from("why")

其存储过程如下图所示：



## 二、使用方法

Buffer 类在全局作用域中，无须require导入

创建Buffer的方法有很多种，我们讲讲下面的两种常见的形式：

* Buffer.from()
* Buffer.alloc()

### Buffer.from()

const b1 = Buffer.from('10');  
const b2 = Buffer.from('10', 'utf8');  
const b3 = Buffer.from([10]);  
const b4 = Buffer.from(b3);  
  
console.log(b1, b2, b3, b4); // <Buffer 31 30> <Buffer 31 30> <Buffer 0a> <Buffer 0a>

### Buffer.alloc()

const bAlloc1 = Buffer.alloc(10); // 创建一个大小为 10 个字节的缓冲区  
const bAlloc2 = Buffer.alloc(10, 1); // 建一个长度为 10 的 Buffer,其中全部填充了值为 `1` 的字节  
console.log(bAlloc1); // <Buffer 00 00 00 00 00 00 00 00 00 00>  
console.log(bAlloc2); // <Buffer 01 01 01 01 01 01 01 01 01 01>

在上面创建buffer后，则能够toString的形式进行交互，默认情况下采取utf8字符编码形式，如下

const buffer = Buffer.from("你好");  
console.log(buffer);  
// <Buffer e4 bd a0 e5 a5 bd>  
const str = buffer.toString();  
console.log(str);  
// 你好

如果编码与解码不是相同的格式则会出现乱码的情况，如下：

const buffer = Buffer.from("你好","utf-8 ");  
console.log(buffer);  
// <Buffer e4 bd a0 e5 a5 bd>  
const str = buffer.toString("ascii");  
console.log(str);   
// d= e%=

当设定的范围导致字符串被截断的时候，也会存在乱码情况，如下：

const buf = Buffer.from('Node.js 技术栈', 'UTF-8');  
  
console.log(buf) // <Buffer 4e 6f 64 65 2e 6a 73 20 e6 8a 80 e6 9c af e6 a0 88>  
console.log(buf.length) // 17  
  
console.log(buf.toString('UTF-8', 0, 9)) // Node.js �  
console.log(buf.toString('UTF-8', 0, 11)) // Node.js 技

所支持的字符集有如下：

* ascii：仅支持 7 位 ASCII 数据，如果设置去掉高位的话，这种编码是非常快的
* utf8：多字节编码的 Unicode 字符，许多网页和其他文档格式都使用 UTF-8
* utf16le：2 或 4 个字节，小字节序编码的 Unicode 字符，支持代理对（U+10000至 U+10FFFF）
* ucs2，utf16le 的别名
* base64：Base64 编码
* latin：一种把 Buffer 编码成一字节编码的字符串的方式
* binary：latin1 的别名，
* hex：将每个字节编码为两个十六进制字符

## 三、应用场景

Buffer的应用场景常常与流的概念联系在一起，例如有如下：

* I/O操作
* 加密解密
* zlib.js

### I/O操作

通过流的形式，将一个文件的内容读取到另外一个文件

const fs = require('fs');  
  
const inputStream = fs.createReadStream('input.txt'); // 创建可读流  
const outputStream = fs.createWriteStream('output.txt'); // 创建可写流  
  
inputStream.pipe(outputStream); // 管道读写

### 加解密

在一些加解密算法中会遇到使用 Buffer，例如 crypto.createCipheriv 的第二个参数 key 为 string 或 Buffer 类型

### zlib.js

zlib.js 为 Node.js 的核心库之一，其利用了缓冲区（Buffer）的功能来操作二进制数据流，提供了压缩或解压功能

## 参考文献

* http://nodejs.cn/api/buffer.html
* https://segmentfault.com/a/1190000019894714