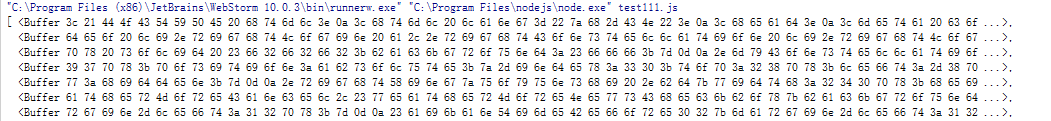
## buffer拼接

### 1：buffer数组与buffer对象

buffer对象



buffer数组



**buffer数组—》buffer对象**

**var** buf=Buffer.concat(***result***);

* result是一个buffer数组
* buf是一个buffer对象
* buffer数组转buffer对象通过Buffer.concat(buffer数组)实现

## 2：Buffer转字符串

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

**问题：---buf.toString()支持的字符编码有限**

encoding并不支持GBK GB2312 BIG-5这些中文常见的格式

解决方法--（buffer拼接+第三方的模块iconv-lite）

**var *http***=*require*(**'http'**);  
**const *BufferList*** = *require*(**'bl'**);  
  
  
**var *iconv***=*require*(**'iconv-lite'**);  
**var *result***=[];  
*//process.argv[2]****http***.get(**'http://www.qq.com'**,**function**(res){  
 **var** converterStream = ***iconv***.decodeStream(**'win1251'**);  
  
 res.on(**'data'**,**function**(chunk){  
 ***result***.push(chunk);  
 });  
  
 res.on(**'end'**,**function**(){  
  
 *// console.log(iconv.decode(bl,'gb2312'));  
 // var buf=Buffer.concat(res,size);* **var** buf=Buffer.concat(***result***);  
 ***console***.log(***iconv***.decode(buf,**'gb2312'**));  
 });  
  
});

## 3:字符乱码问题

在http模块中获取数据的时候，可能会产生字符乱码的问题

**解决方式有两种**

* 设置可读流的setEncoding()
* buf拼接+第三方模块iconv-lite

## readable.setEncoding(encoding)

**作用：**

让data事件中传递的不再是一个Buffer对象，而是一个编码之后的字符串。

**乱码问题得以解决的关键**

* 在调用setEncoding的时候，可读流对象在内部设置了一个decoder对象，每次data事件都通过该decoder对象进行Buffer到字符串的解码，然后传递给调用者。

**局限性**

* 支持的编码过少。它目前只能处理UTF-8 Base-64 UCS-2/UTF-16这三种编码。

（不支持gk2312）

**针对局限性的解决方法**

* 可以选着buf拼接和第三方模块实现（iconv-lite）