Node.js第二章 - 常用系统API

本章主要内容

```
1. 了解掌握Buffer（缓冲区）的创建与使用方式
2. 了解掌握File System(文件系统)的使用方式以及常用方法
3. 了解掌握Node.js中负责事件监听与处理的Events模块
4. 熟悉、掌握Node.js模块的加载与使用方法
```

Buffer

> Buffer是Node.js中专门用于存放二进制数据的缓冲区

Buffer的创建

```
Buffer.alloc()
Buffer.from()

注：  支持以下编码格式
ASCII
UTF-8
UTF-16LE/UCS-2
Base64
Binary
Hex
```

Buffer的输出

```
// 直接在控制台输出
console.log(buf)

// 将Buffer对象转为字符串输出
console.log(buf.toString())
```

Buffer的其他方法

```
Buffer.concat([buf1,buf2...])
Buffer.compaare(buf1,buf2)
Buffer.isBuffer(buf)
Buffer.byteLength()

buf[index]
buf.compare()
buf.copy()
buf.fill()
buf.indexOf()
buf.lastIndexOf()
buf.slice()
```

> 注：多查手册 按套路出牌

FileSystem 文件系统

> 文件系统模块是操作文件的模块，可用于文件的读取、创建、删除等操作

fs模块的加载与基本使用

```
// 加载模块
var fs = require('fs');
// 调用方法
fs.mkdir('./memed',function(err){
    // err 接收错误信息
})
```

fs模块常用方法

```
// 目录相关
fs.mkdir()
fs.rmdir()
fs.readdir()

// 文件相关
fs.redFile()
fs.writeFile()
fs.appendFile()
fs.unlink()

// 文件字节操作
fs.open()        // 必须要选择打开方式
fs.read()
fs.write()
fs.close()

// 共有方法
fs.rename()
fs.access()

// 读取文件状态
fs.stat(file,function(err,stats){
    // stats 包含了文件相关信息

    // 判断文件类型
    stats.isFile();
    stats.isDirectory();
})
```

文件流式操作

可读流

```
var rs = fs.createReadStream();

    // 事件
        data        数据传输事件
        end         传输完毕事件
        error       监听错误事件
```

可写流

```
var ws = fs.creaateWriteStream();

    // 事件
        finish      数据传输完毕
        error       数据写入有误

    // 写入
    ws.write(data);

    // 标记写入完毕
    ws.end();
```

管道流

```
// 可读流向可写流写入
pipe()
```

压缩文件

```
// 借助文件系统、管道流与zlib模块完成文件压缩
var fs = require('fs');
var zlib = require('zlib');

// 创建可读流与可写流
var rs = fs.createReadStrem('./1.txt');
var ws = fs.createWriteStream('./1.txt.rar');

// 创建压缩方式
var gzip = zlib.creaateGzip();

// 压缩走起
rs.pipe(gzip).pipe(ws);
```

Events 事件模块

> Node.js的API大多数是采用异步事件驱动的架构，事件环机制，基于Events事件模块

基本使用

```
var events = require('events');

// 实例化创建监听器
var em = new events.EventEmitter();

// 设置监听事件（同一事件我们可以添加多个处理事件）
em.on('papapa',function(){});
em.on('biaobai',function(){});

// 触发事件
em.emit('papapa');

once() 创建一次性监听器
addListener() 添加监听器
removeListener() 移除监听器(指定监听器与事件处理函数)
```