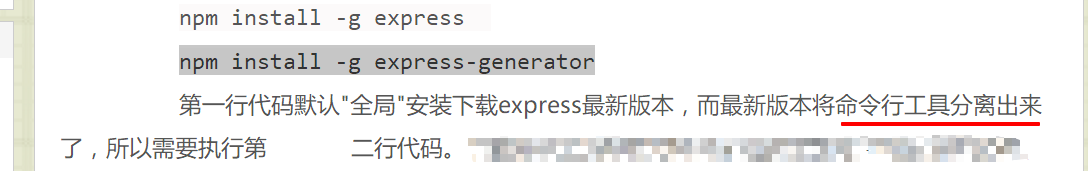
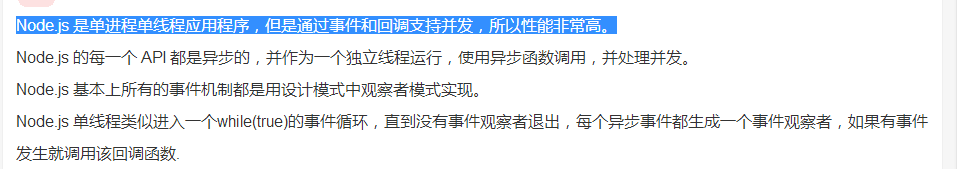
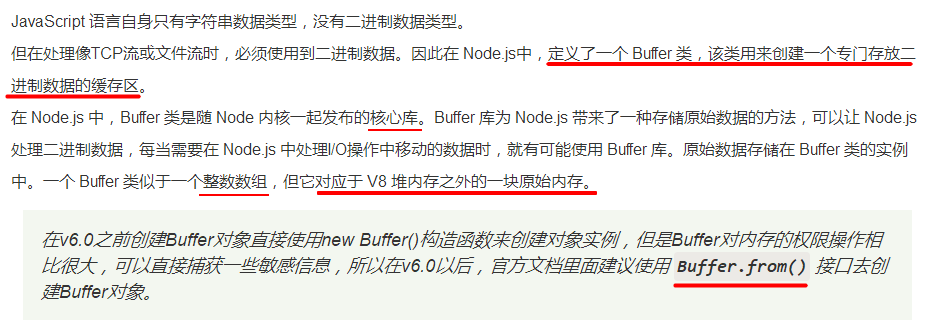
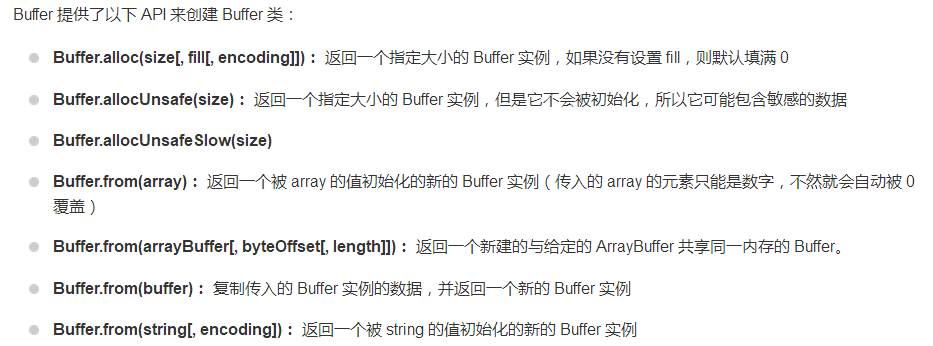
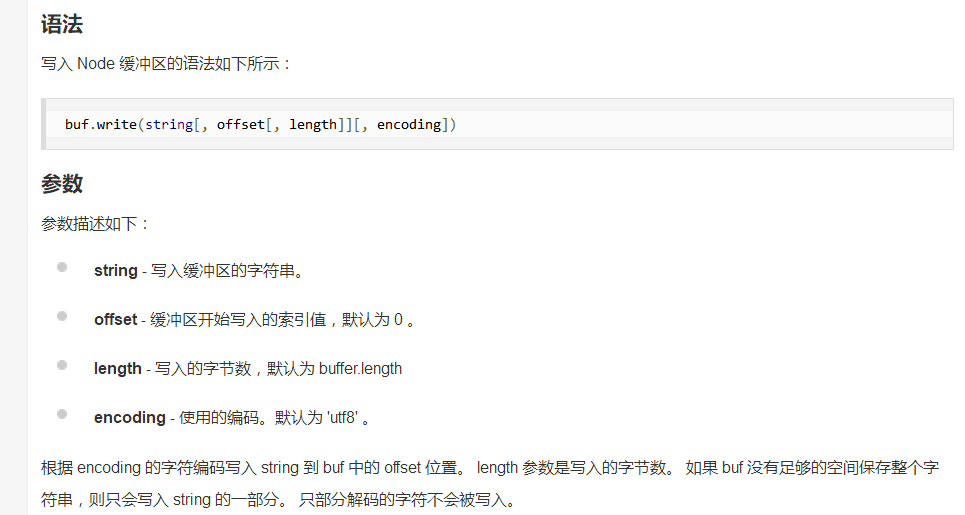
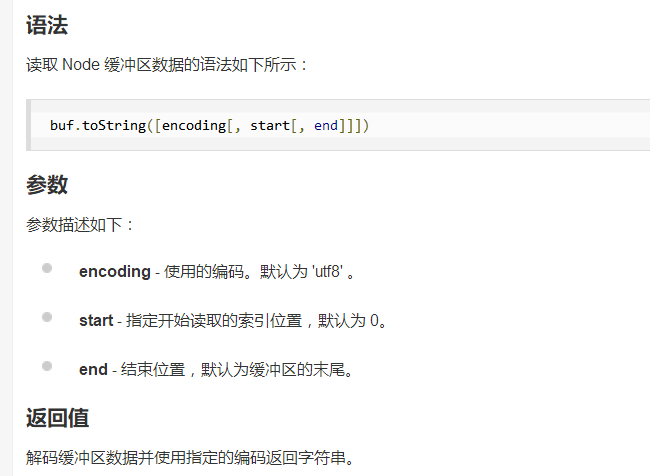
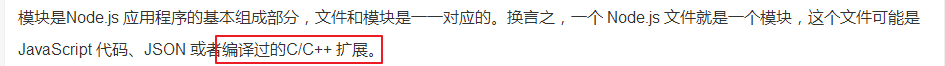
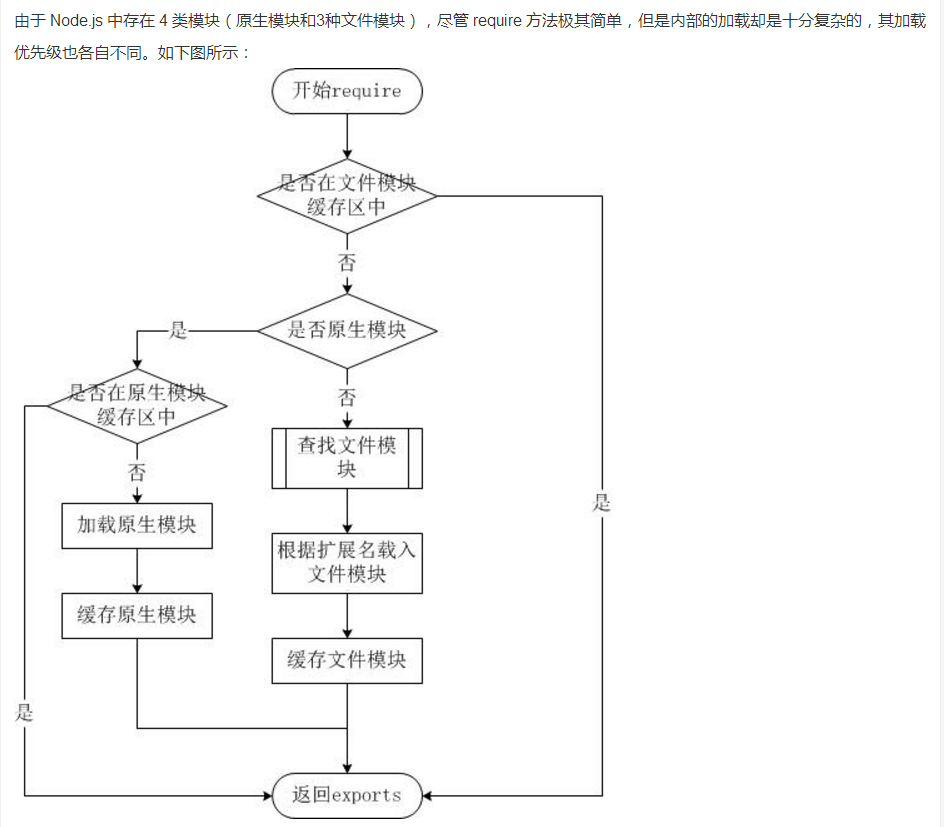
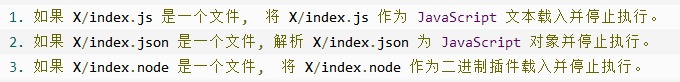
* Npm list -g
* Npm list <module-name>
* 
* Npm uninstall <module-name> 卸载模块
* 
* npm update -g 全局更新所有包
* npm update <name> -g 全局更新指定包
* npm update <name> --save 更新项目中生成环境的指定包
* npm update <name> --save-dev 更新项目中开发环境的指定包
* npm install -l 安装package.json中的module
* npm search <package name> 搜索包
* Npm install -g 默认是安装在nodejs的安装路径下的node\_module文件夹下
* 
* Nodejs异步读取文件：<http://www.runoob.com/nodejs/nodejs-callback.html>
* Node.js 是**单进程单线程**应用程序，但是通过**事件和回调**支持并发，所以性能非常高。

Node.js ***REPL***(交互式解释器)

* 两次按下 ctrl-c 按钮退出
* 使用下划线(\_)获取上一个表达式的运算结果
* ctrl + d - 退出 Node REPL.
* .help - 列出使用命令
* .break - 退出多行表达式
* JavaScript 语言自身只有字符串数据类型，没有二进制数据类型。但在处理像TCP流或文件流时，必须使用到二进制数据。因此在 Node.js中，定义了一个 Buffer 类，该类用来创建一个专门存放二进制数据的缓存区。

Node.js ***Buffer***(缓冲区)

* 为何提供buffer类：
* Buffer支持的编码格式：
* 创建buffer：
* 写入buffer：
* 读取buffer：
* Buffer合并：
* 更多node buffer的api：<http://www.runoob.com/nodejs/nodejs-buffer.html>
* 类似java的native：
* 关于exports，其实是一个对象，通过exports.XXX设置属性，提供给外部访问
* module.exports = {}，是将一个对象赋值给exports。（参考：<http://www.runoob.com/nodejs/nodejs-module-system.html）>
* Nodejs中模块加载策略（require策略）：
* Nodejs模块包括4类:原生，json，js，node；
* 所有的全局属性（除了global本身），都是global 对象的属性
* 隐式定义的变量（未定义直接赋值的变量）也是全局变量
* 全局变量：

1. \_\_filename
2. \_\_dirname
3. setTimeout(cb, ms) -- 指定的毫秒(ms)数后执行指定函数(cb)，执行一次
4. clearTimeout(t) -- t表示setTimeout创建的定时器
5. setInterval(cb, ms) -- 定时循环执行，使用clearInterval(t)清除
6. Console -- 

* Nodejs8+中实现sleep功能：<https://blog.csdn.net/zzwwjjdj1/article/details/78931272>
* util.inherits(sub, base) 这种方式sub只会继承base原型（prototype）中的属性和方法，base构造器中的属性和方法不会被继承
* ES6标准发布后，module成为标准，标准的使用是以export指令导出接口，以import引入模块，但是在我们一贯的node模块中，我们采用的是CommonJS规范，使用require引入模块，使用module.exports导出接口。
* 但是**import**则不同，它是**编译时**的（**require是运行时**的），它必须放在文件开头，而且使用格式也是确定的，不容置疑。它不会将整个模块运行后赋值给某个变量，而是只选择import的接口进行编译，这样在性能上比require好很多。