fs

Node.js内置的fs模块就是文件系统模块，负责读写文件。

和所有其它JavaScript模块不同的是，fs模块同时提供了异步和同步的方法。

异步读文件

```
'use strcit';
var fs = require('fs');
// 异步读取文件
fs.readFile('../index.js', 'utf-8' , function (err,data) {
    if(err) {
        console.log(err);
    }else {
        console.log(data);
    }
})
```

当正常读取时，err参数为null，data参数为读取到的String。当读取发生错误时，err参数代表一个错误对象，data为undefined。如果不加utf-8，则返回Buffer对象（二进制数据）

```
// String -> Buffer
var buf = Buffer.from(text, 'utf-8');
console.log(buf);
```

同步读文件

```
try {
    var data = fs.readFileSync('sample.txt', 'utf-8');
    console.log(data);
} catch (err) {
    // 出错了
}
```

写文件

```
var data = 'hello node';
fs.writeFile('fs.txt', data, function(err) {
    if(err) {
        console.log(err)
    }else {
        console.log('ok')
    }
```

```
})
```

注意：覆盖原文件内容

和readFile()类似，writeFile()也有一个同步方法，叫writeFileSync()：

stat

获取文件信息

```
fs.stat('fs.txt', function(err, stat) {
    if(err) {
        console.log(err);
    }else {
        // 是否是文件
        console.log('isFile:' + stat.isFile());
        // 是否是目录
        console.log('isDirectory:' + stat.isDirectory());
        if(stat.isFile) {
            // 文件大小
            console.log('size:' + stat.size);
            // 文件创建时间
            console.log('birth time:' + stat.birthtime);
            // 修改时间
            console.log('mtime:' + stat.mtime);
        }
    }
})
```

stat()也有一个对应的同步函数statSync()，请试着改写上述异步代码为同步代码。

由于Node环境执行的JavaScript代码是服务器端代码，所以，**绝大部分需要在服务器运行期反复执行业务逻辑的代码，必须使用异步代码**，否则，同步代码在执行时期，服务器将停止响应，因为JavaScript只有一个执行线程。

服务器**启动时如果需要读取配置文件，或者结束时需要写入到状态文件时，可以使用同步代码**，因为这些代码只在启动和结束时执行一次，不影响服务器正常运行时的异步执行。