基本模块

因为Node.js是运行在服务区端的JavaScript环境，服务器程序和浏览器程序相比，**最大的特点是没有浏览器的安全限制了**，而且，**服务器程序必须能接收网络请求，读写文件，处理二进制内容**，所以，Node.js内置的常用模块就是为了实现基本的服务器功能。**这些模块在浏览器环境中是无法被执行的，因为它们的底层代码是用C/C++在Node.js运行环境中实现的。**

global

类似于浏览器中的window对象（全局对象）

process

是Node.js提供的一个对象，它代表当前Node.js进程。通过process对象可以拿到许多有用信息

```
> process === global.process;
true
> process.version;
'v5.2.0'
> process.platform;
'darwin'
> process.arch;
'x64'
> process.cwd(); //返回当前工作目录
'/Users/michael'
> process.chdir('/private/tmp'); // 切换当前工作目录
undefined
> process.cwd();
'/private/tmp'
```

JavaScript程序是由事件驱动执行的单线程模型，Node.js也不例外。Node.js不断执行响应事件的JavaScript函数，直到没有任何响应事件的函数可以执行时，Node.js就退出了。

如果我们想要在下一次事件响应中执行代码，可以调用process.nextTick()：

```
process.nextTick(function () {
    console.log('nextTick callback!');
});
console.log('nextTick was set!');
```

输出:

```
nextTick was set!
nextTick callback!
```

这说明传入process.nextTick()的函数不是立刻执行，而是要等到下一次事件循环。

```
// 程序即将退出时的回调函数:
process.on('exit', function (code) {
    console.log('about to exit with code: ' + code);
});
```

判断JavaScript执行环境

```
if(typeof(window) === 'undefined') {
    console.log("node.js")
}else {
    console.log("browser")
}
```