Node.js API

File System(fs)模块 API	
方法	**描述**
fs.readFile(path,encoding,callback[err,data]);	Asynchronously reads the entire contents of a file.(异步读取一个文件的所有内容) 1. path:文件路径 2. encoding:字符集编码——utf8 3. callback:回调函数，此函数有两个参数 err，data——文件数据
fs.appendFile(path,data,encoding,callback[err]);	Asynchronously append data to a file, creating the file if it not yet exists. data can be a string or a buffer.(异步的追加数据到一个文件，如果这个文件不存在就创建。数据可以是一个字符串或者缓冲区) 1. path:文件路径 2. data:要写入文件的数据 3. encoding:字符集编码——utf8 4. callback:回调函数，此函数有一个参数 err
fs.writeFile(path,data,encoding,callback[err]);	Asynchronously writes data to a file, replacing the file if it already exists. data can be a string or a buffer.(异步写入数据到一个文件，如果文件存在就替换文件中的内容。数据可以是一个字符串或缓冲区) 1. path:文件路径 2. data:要写入文件的数据 3. encoding:字符集编码——utf8 4. callback:回调函数，此函数有一个参数 err
fs.exists(path, callback);	Test whether or not the given path exists by checking with the file system.Then call the callback argument with either true or false.(通过检测文件系统测试给定路径的文件是否存在，然后调用回调函数，无论返回的是 true 还是 false) 1.path:文件路径 2.callback:回调函数，此回调函数有一个参数，如果文件存在则返回 true，否则返回 false
fs.rename(oldPath, newPath, callback);	Asynchronous rename. No arguments other than a possible exception are given to the completion callback.(异步的对文件进行重命名。没有异常出现则回调函数不会传递任何参数) 1.oldPath:需要重命名文件的原路径 2.newPath:文件重命名以后的路径及文件名称 3.callback:回调函数，此函数接受一个参数 err，如果执行成功此参数的值为 null
fs.chown(path, uid, gid, callback);	Asynchronous chown. No arguments other than a possible exception are given to the completion callback.(异步修改文件所有者。没有出现异常，则回调函数不传递任何参数) 1.path:文件路径 2.uid:用户 id 3.gid:组 id 4.callback:回调函数，此函数接受一个参数，如果执行成功则此回调函数的参数为 null
fs.stat(path, callback);	Asynchronous stat. The callback gets two arguments (err, stats) where stats is a fs.Stats object.See the fs.Stats section below for more information.(异步获取文件状态信息。这个回调函数有两个参数 err，stats； stats 是一个 fs.stats 对象)
Class: fs.Stats	Objects returned from fs.stat(), fs.lstat() and fs.fstat() and their synchronous counterparts are of this type.(返回的对象来自于 fs.stat(); fs.last(); fs.fstat(); 和这些方法对应的同步方法，他们都属于这种类型) 1. stats.isFile() 是否是文件

	2. stats.isDirectory()是否是目录 3. stats.isBlockDevice()若是块设备则返回 true，在大多半 UNIX 体系中块设备凡是都在/dev 目录下 4. stats.isCharacterDevice()若是是字符设备返回 true 若是是字符设备返回 true 5. stats.isSymbolicLink() (only valid with fs.lstat()--->仅对 fs.lstat()有效)若是是文件链接返回 true 若是是文件链接返回 true 6. stats.isFIFO()若是个 FIFO（UNIX 定名管道的一个特别类型）返回 true 7. stats.isSocket()是否是 socket
I/O 设备大致分为两类：块设备和字符设备。 1. 块设备将信息存储在固定大小的块中，每个块都有自己的地址。数据块的大小通常在 512 字节到 32768 字节之间。块设备的基本特征是每个块都能独立于其它块而读写。磁盘是最常见的块设备。 2. 字符设备是指在 I/O 传输过程中以字符为单位进行传输的设备，例如键盘，打印机等。请注意，以字符为单位并不一定意味着是以字节为单位，因为有的编码规则规定，1 个字符占 16 比特，合 2 个字节。	
Class : fs.Stats property	{ dev: 2054, mode: 33188, nlink: 1, uid: 0, gid: 0, rdev: 0, blksize: 4096, ino: 776477, size: 32, blocks: 8, atime: Wed Sep 04 2013 11:11:08 GMT+0800 (CST), mtime: Wed Sep 04 2013 11:11:08 GMT+0800 (CST), ctime: Wed Sep 04 2013 11:11:08 GMT+0800 (CST) }
fs.utimes(path, atime, mtime, callback);	Change file timestamps of the file referenced by the supplied path. 改变由路径提供的参考文件的时间戳(此方法改变的是 atime——访问时间和 mtime——修改时间) 1. path:路径 2. atime:访问时间 3. mtime:修改时间 4. callback:回调函数
fs.mkdir(path, [mode], callback);	Asynchronous mkdir(2). No arguments other than a possible exception are given to the completion callback. mode defaults to 0777.(异步的创建目录，没有异常那么回调函数不传递任何参数)
fs.readdir(path, callback);	Asynchronous readdir(3). Reads the contents of a directory. The callback gets two arguments (err, files) where files is an array of the names of the files in the directory excluding '.' and '..'.(异步调用 readdir，读取目录中的内容。回调函数接受两个参数(err, files)，其中 files 参数是保存了目录中所有文件名的数组（'.'和'..'除外）)
fs.rmdir(path, callback);	Asynchronous rmdir. No arguments other than a possible exception are given to the completion callback.(异步的移除 path 所对应的文件或目录，没有异常那么回调函数不传递任何参数)
fs.link(srcpath, dstpath, [callback]);	Asynchronous link. No arguments other than a possible exception are given to the completion callback.(异步调用 link，创建符号连接，除非回调函数执行过程出现了异常，否则不会传递任何参数。)
fs.symlink(srcpath, dstpath, [type], callback);	Asynchronous symlink. No arguments other than a possible exception are given to the completion callback. type argument can be either 'dir', 'file', or 'junction' (default is 'file'). It is only used on Windows (ignored on other platforms). Note that Windows junction points require the destination path to be absolute.

	When using 'junction', the destination argument will automatically be normalized to absolute path. 异步调用 symlink——符号连接——软连接，除非回调函数执行过程出现了异常，否则不会传递任何参数。type 参数可能是目录，文件，节点，默认是文件。它仅用于 Windows（在其他平台上忽略）。注意 Windows 结点需要目的地的路径是绝对的。使用“连接”时，目标参数将自动被归一化到的绝对路径。