JSON.parse()与JSON.stringify()的区别

JSON.parse()【从一个字符串中解析出json对象】

例子：

//定义一个字符串

var data='{"name":"goatling"}'

//解析对象

JSON.parse(data)

结果是：

name:"goatling"

JSON.stringify()【从一个对象中解析出字符串】

var data={name:'goatling'}

JSON.stringify(data)

结果是：

'{"name":"goatling"}'

fs.writeFile

var fs = require('fs'); // 引入fs模块

// 写入文件内容（如果文件不存在会创建一个文件）
// 传递了追加参数 { 'flag': 'a' }
fs.writeFile('./try4.txt', 'HelloWorld', { 'flag': 'a' }, function(err) {
if (err) {
throw err;
}

fs.readFileSync()

语法：
fs.readFileSync(filename, [encoding])
接收参数：

filename 文件路径
options option对象，包含 encoding，编码格式，该项是可选的。
例子：
var fs = require(‘fs’);
var contentText = fs.readFileSync(‘data.json’,’utf-8’);

fs.stat() 获取文件信息

语法：
fs.stat(path, [callback(err, stats)])
接收参数：
path 文件路径
callback 回调，传递两个参数，异常参数err, 文件信息数组 stats

stats类中的方法有：

方法	描述
stats.isFile()	如果是文件返回 true，否则返回 false。
stats.isDirectory()	如果是目录返回 true，否则返回 false。
stats.isBlockDevice()	如果是块设备返回 true，否则返回 false。
stats.isCharacterDevice()	如果是字符设备返回 true，否则返回 false。
stats.isSymbolicLink()	如果是软链接返回 true，否则返回 false。
stats.isFIFO()	如果是FIFO，返回true，否则返回 false。FIFO是UNIX中的一种特殊类型的命令管道。
stats.isSocket()	如果是 Socket 返回 true，否则返回 false。

path.join 与 path.resolve 的区别

1. 对于以/开始的路径片段，path.join只是简单的将该路径片段进行拼接，而path.resolve将以/开始的路径片段作为根目录，在此之前的路径将会被丢弃，就像是在terminal中使用cd命令一样。

path.join(‘/a’, ‘/b’) // ‘a/b’
path.resolve(‘/a’, ‘/b’) // ‘/b’

2. path.resolve总是返回一个以相对于当前的工作目录（working directory）的绝对路径。

path.join(‘./a’, ‘./b’) // ‘a/b’
path.resolve(‘./a’, ‘./b’) // ‘/Users/username/Projects/webpack-demo/a/b’

get、post、put、delete、head请求方式

对资源的增，删，改，查操作，其实都可以通过GET/POST完成，不一定要用PUT和

DELETE。

一：Jersey框架，实现了restful风格，常用的注解@GET、@POST、@PUT、@DELETE如下：

GET：

对应get请求
作用：标识该操作是用于获取服务端的资源，可以理解为select操作

特点：GET方式提交的数据最多只能是2KB字节；数据通过browser地址栏进行传递，用户信息会暴露在browser地址了，不安全

POST：

对应post请求
作用：用于向服务端新增数据，常用于提交表单。可以理解为insert操作

特点：理论上POST方式，对提交的数据大小没有限制；数据通过http协议的body体中进行传递，不会暴露用户信息，相对安全

PUT：

对应put请求

作用：用于向服务端更新数据，与post的使用很相似。可以理解为update操作

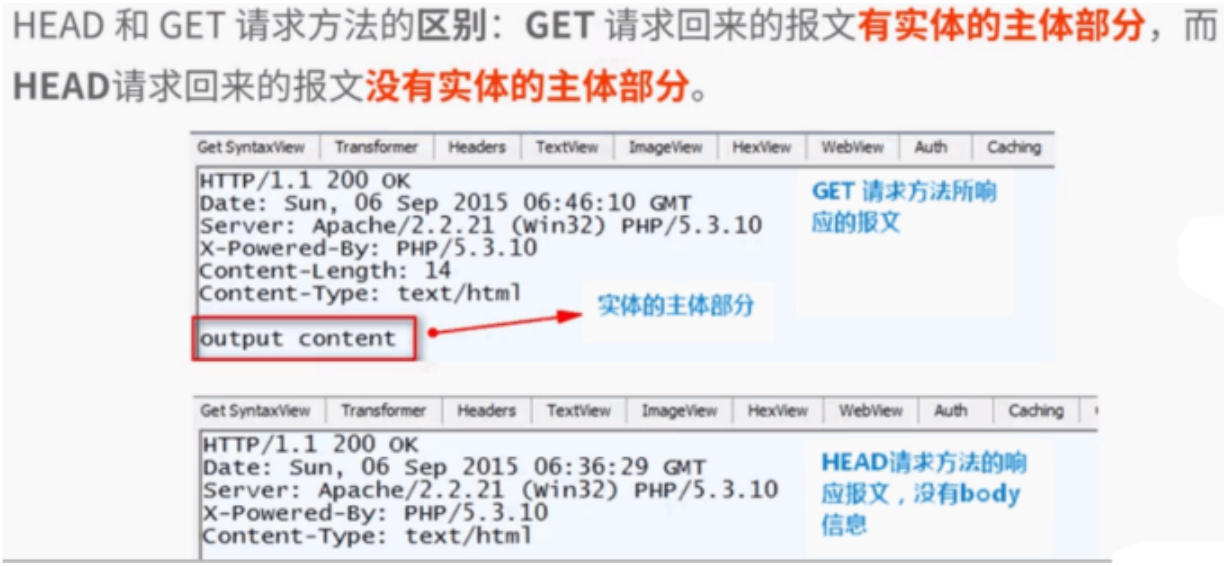
DELETE：

对应delete请求

作用：标识该操作是：用于删除服务端的资源，可以理解为delete操作

二：@HEAD：
作用：只请求页面首部，响应报文中没有实体的主体部分(没有body体)

与@GET的区别如图：



三：注：可能在一个操作里，还会涉及到相关联的一系列操作，那么按照最初的操作来定义请求。

eg：修改出生birthDate，应该是对应的put请求；但是又关联修改年龄age，还新增insert或者删除delete关联的数据(可能不符合逻辑)，此时有是post、delete的请求方式。但是我们方法对应的请求只能有一个，我们会使用最初的put请求，因为后面的相关操作都是在修改birthDate的基础上来展开的，我们只需要给出最初的请求方式即可。

使用事件发射器API

任何实现了事件发射器模式的对象（比如TCP Socket，HTTP 请求等）都实现了下面的一组方法：

```
.addListener和.on —— 为指定类型的事件添加事件监听器
.once —— 为指定类型的事件绑定一个仅执行一次的事件监听器
.removeEventListener —— 删除绑定到指定事件上的某个监听器
.removeAllEventListener —— 删除绑定到指定事件上的所有监听器
```

使用.addListener()或.on()绑定回调函数

readStream.on(“data”, function(data) {
console.log(“got data from file read stream: %j”, data);
});