前面我们学习了模型，事件驱动，今天我们来看一下在nodejs中路径的相关操作方法。在我们开发过程中路径主要有:URL路径，在浏览器中进行使用，当然这个也包含查询字符串QueryString的相关操作；而另一种是磁盘路径，主要用于文件等等进行操作，我们称为Path，所以我将它们放在一起进行总结学习。

**URL：**

要使用URL模型，则可以用require('url')进行引入，我们都知道URL包含了很多部分，这个在JavaScript中也有相关的操作即是location对象。下面将他的各个属性进行一番记载，首先给个URL地址：'http://user:pass@host.com:8080/p/a/t/h?query=string#hash'，然后我们将对该地址进行分解个解析，这样可以更好的理解他们。

* ***href:***整个URL地址字符串，不过在nodejs中已经将协议字符串和域名字符串转换为小写。如：'http://user:pass@host.com:8080/p/a/t/h?query=string#hash'；
* ***protocol:***请求协议字符串，如："http:"，小写字符串；
* ***host:***主机域名，包含端口的小写字符串，如： 'host.com:8080'；
* ***auth:*** 验证信息，如：'user:pass'；
* ***hostname:*** 主机域名，小写字符串，如： 'host.com'；
* ***port:*** 端口，如： '8080'；
* ***pathname:***文件路径，查询字符串之前，域名端口之后的字符串，以“/”开头，如： '/p/a/t/h'；
* ***search:***查询字符串，包含“?”，如： '?query=string'；
* ***path:***包含文件路径和查询字符串，如：'/p/a/t/h?query=string'；
* ***query:***查询字符串键值对字符串或JSON对象，如：'query=string' or {'query':'string'}；
* ***hash:*** 锚点标记，如：'#hash'；

在URL对象中除了上面的属性外，还提供了一些有用的方法，具体如下：

***url.parse(urlStr, [parseQueryString], [slashesDenoteHost])：***

将字符串转换为URL对象。

第二个参数表示是否也将查询字符串通过QueryString模型进行分析，默认为false；

第三个参数表示是否将“//”作为协议前缀进行分析，如："//foo/bar"，将被分析为：{host:'foo', pathname:'/bar'}而不是：{pathname:'//foo/bar'}，默认设置为false。

***url.format(urlObj)：***

将URL对象，根据格式化的对象返回一个URL字符串。

* href属性将被忽略；
* protocol不同协议处理的方法是不一样的：
  1. 协议http, https, ftp, gopher, file将以“ :// ”结尾；
  2. 其它协议如：mailto, xmpp, aim, sftp, foo等将以“:”结尾；

***url.resolve(from, to)：***

URL地址转换，可以看一下如下代码：

url.resolve('/one/two/three', 'four') // '/one/two/four'

url.resolve('http://example.com/', '/one') // 'http://example.com/one'

url.resolve('http://example.com/one', '/two') // 'http://example.com/two'

**Query String：**

查询字符串主要由两个方法和内置格式化方法组成，一个是将对象转换为字符串，一个则是相反，将字符串转换为对象：

***querystring.stringify(obj, [sep], [eq])：***

将JSON对象格式化为查询字符串格式的字符串，默认的分隔符为：“&”和“=”，具体可以看一下以下代码：

querystring.stringify({ foo: 'bar', baz: ['qux', 'quux'], corge: '' })

// returns

'foo=bar&baz=qux&baz=quux&corge='

querystring.stringify({foo: 'bar', baz: 'qux'}, ';', ':')

// returns

'foo:bar;baz:qux'

***querystring.parse(str, [sep], [eq], [options])：***

根据“&”和“=”将字符串进行分割，反序列化为JSON对象，而options包含的maxKeys默认设置为1000，如果将其设置为0则表示没这个限制。

querystring.parse('foo=bar&baz=qux&baz=quux&corge')

// returns

{ foo: 'bar', baz: ['qux', 'quux'], corge: '' }

***querystring.escape，querystring.unescape：***

这两个内置方法，分别在上述两个方法的内置使用，如果有需要分别格式化和解码URL字符串。

**Path：**

路径主要提供一些有用的方法，来转换文件路径的操作，并且可以提供各种方法验证路径是否合法，如果要使用Path，则使用require('path')引入。

***path.normalize(p)：***

讲一个路径正常化，如果路径包含“.”或包含“..”，和两个“//”都将被格式化成正确的路径，可以看如下代码：

path.normalize('/foo/bar//baz/asdf/quux/..')

// returns

'/foo/bar/baz/asdf'

***path.join([path1], [path2], [...])：***

拼接路径的各个部分。

path.join('/foo', 'bar', 'baz/asdf', 'quux', '..')

// returns

'/foo/bar/baz/asdf'

path.join('foo', {}, 'bar')

// returns

'foo/bar'

***path.resolve([from ...], to)：***

这个命令和join不一样，而相当于命令汉中的cd命令，例如：

path.resolve('foo/bar', '/tmp/file/', '..', 'a/../subfile')

以上代码和在命令行执行如下代码是相似的：

cd foo/bar

cd /tmp/file/

cd ..

cd a/../subfile

pwd

不同的是，所拼接的路径可以不一定存在，而命令行中的路径一定要存在。

path.resolve('/foo/bar', './baz')

// returns

'/foo/bar/baz'

path.resolve('/foo/bar', '/tmp/file/')

// returns

'/tmp/file'

path.resolve('wwwroot', 'static\_files/png/', '../gif/image.gif')

// if currently in /home/myself/node, it returns

'/home/myself/node/wwwroot/static\_files/gif/image.gif'

***path.relative(from, to)：***

转换相对路径，即to相对于from的路径，这个方法是将路径转换为绝对路径的反向路径，有如下等式：

path.resolve(from, path.relative(from, to)) == path.resolve(to)

例如执行如下代码后将得到相应的结果：

path.relative('C:\\orandea\\test\\aaa', 'C:\\orandea\\impl\\bbb')

// returns

'..\\..\\impl\\bbb'

path.relative('/data/orandea/test/aaa', '/data/orandea/impl/bbb')

// returns

'../../impl/bbb'

***path.dirname(p)：***

获取路径的文件夹路径。

***path.basename(p, [ext])：***

获取路径的文件名，其中[ext]为扩展名，结果将文件名移除扩展名后的字符串，可以看一下如下代码，就更明白是怎么回事了。

path.basename('/foo/bar/baz/asdf/quux.html')

// returns

'quux.html'

path.basename('/foo/bar/baz/asdf/quux.html', '.html')

// returns

'quux'

***path.extname(p)：***

获取扩展名，即文件名“.”之后包含“.”的字符串，如果“.”不存在，则返回“..”，如果存在“.”则返回“.”，代码如下：

path.extname('index.html')

// returns

'.html'

path.extname('index.')

// returns

'.'

path.extname('index')

// returns

''

***path.sep：***

表示路径的分隔符：“/”或“\”。

***path.delimiter：***

路径分隔符：“:”和“;”，如下代码可以看到是怎么个回事：

console.log(process.env.PATH)

// '/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin'

process.env.PATH.split(path.delimiter)

// returns

['/usr/bin', '/bin', '/usr/sbin', '/sbin', '/usr/local/bin']

在windows下的代码将是：

console.log(process.env.PATH)

// 'C:\Windows\system32;C:\Windows;C:\Program Files\nodejs\'

process.env.PATH.split(path.delimiter)

// returns

['C:\Windows\system32', 'C:\Windows', 'C:\Program Files\nodejs\']

本文将路径URL，Query String以及Path的一些方法或属性，这样对我解析文件或其他的就会有所了解了

参考文献：

<http://www.mozlite.com/nodejs-url-querystring-path>