## 1.url模块

nodejs中用户url格式化和反格式化模块  
用于url解析、处理等操作的解决方案

#### 1.url.parse(urlString[, parseQueryString[, slashesDenoteHost]])

* urlString [<string>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FData_structures%23String_type) 要解析的 URL 字符串。
* parseQueryString [<boolean>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FData_structures%23Boolean_type) 如果为 true，则 query 属性总会通过 [querystring](https://link.jianshu.com?t=http%3A%2F%2Fnodejs.cn%2Fapi%2Fquerystring.html) 模块的 parse() 方法生成一个对象。 如果为 false，则返回的 URL 对象上的 query 属性会是一个未解析、未解码的字符串。 默认为 false。
* slashesDenoteHost [<boolean>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FData_structures%23Boolean_type) 如果为 true，则 // 之后至下一个 / 之前的字符串会被解析作为 host。 例如，//foo/bar 会被解析为 {host: 'foo', pathname: '/bar'} 而不是 {pathname: '//foo/bar'}。 默认为 false。  
  url.parse() 方法会解析一个 URL 字符串并返回一个 URL 对象。

如果urlString不是字符串将会抛出TypeError。

如果auth属性存在但无法编码则抛出URIError。

示例1：

var url = require("url")

var myurl="http://www.nodejs.org/some/url/?with=query&param=that#about"

parsedUrl=url.parse(myurl)

结果

{ protocol: 'http:',

slashes: true,

auth: null,

host: 'www.nodejs.org',

port: null,

hostname: 'www.nodejs.org',

hash: '#about',

search: '?with=query&param=that',

query: 'with=query&param=that',

pathname: '/some/url/',

path: '/some/url/?with=query&param=that',

href: 'http://www.nodejs.org/some/url/?with=query&param=that#about'

}

当parse方法第二个参数为true时，结果如下

parsedUrl=url.parse(myurl,true)

{ protocol: 'http:',

slashes: true,

auth: null,

host: 'www.nodejs.org',

port: null,

hostname: 'www.nodejs.org',

hash: '#about',

search: '?with=query&param=that',

query: { with: 'query', param: 'that' },

pathname: '/some/url/',

path: '/some/url/?with=query&param=that',

href: 'http://www.nodejs.org/some/url/?with=query&param=that#about' }

#### 2.url.format(urlObject)

* urlObject [<Object>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FReference%2FGlobal_Objects%2FObject) | [<string>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FData_structures%23String_type) 一个 URL 对象（就像 url.parse() 返回的）。 如果是一个字符串，则通过 url.parse() 转换为一个对象。

url.format() 方法返回一个从 urlObject 格式化后的 URL 字符串。

如果 urlObject 不是一个对象或字符串，则 url.format() 抛出 [TypeError](https://link.jianshu.com?t=http%3A%2F%2Fnodejs.cn%2Fapi%2Ferrors.html%23errors_class_typeerror)。

示例

var url=require('url');

var obj1={ protocol: 'http:',

slashes: true,

auth: null,

host: 'calc.gongjuji.net',

port: null,

hostname: 'calc.gongjuji.net',

hash: '#one#two',

search: '?name=zhangsan&age=18',

query: 'name=zhangsan&age=18',

pathname: '/byte/',

path: '/byte/?name=zhangsan&age=18',

href: 'http://calc.gongjuji.net/byte/?name=zhangsan&age=18#one#two'

};

var url1=url.format(obj1);

console.log(url1);//http://calc.gongjuji.net/byte/?name=zhangsan&age=18#one#two

//请求参数为为json对象

var obj2={ protocol: 'http:',

slashes: true,

auth: null,

host: 'calc.gongjuji.net',

port: null,

hostname: 'calc.gongjuji.net',

hash: '#one#two',

search: '?name=zhangsan&age=18',

query: { name: 'zhangsan', age: '18' }, //页面参数部分，已经解析成对象了

pathname: '/byte/',

path: '/byte/?name=zhangsan&age=18',

href: 'http://calc.gongjuji.net/byte/?name=zhangsan&age=18#one#two' };

var url2=url.format(obj2);

console.log(url2); //http://calc.gongjuji.net/byte/?name=zhangsan&age=18#one#two

//缺少参数的情况

var obj3={ protocol: null,

slashes: true,

auth: null,

host: 'www.gongjuji.net',

port: null,

hostname: 'www.gongjuji.net',

hash: '#one',

search: '?name=zhangsan',

query: { name: 'zhangsan' },

pathname: '/byte/',

path: '/byte/?name=zhangsan',

href: '//www.gongjuji.net/byte/?name=zhangsan#one' };

var url3=url.format(obj3);

console.log(url3);//www.gongjuji.net/byte/?name=zhangsan#one

#### 3.url.resolve(from, to)

* from [<string>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FData_structures%23String_type) 解析时相对的基本 URL。
* to [<string>](https://link.jianshu.com?t=https%3A%2F%2Fdeveloper.mozilla.org%2Fen-US%2Fdocs%2FWeb%2FJavaScript%2FData_structures%23String_type) 要解析的超链接 URL。

url.resolve() 方法会以一种 Web 浏览器解析超链接的方式把一个目标 URL 解析成相对于一个基础 URL。

例子

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'

## 注意：为了方便参考日志，可以安装一个日志模块叫做log4js

### 安装方法：在项目的根目录下面执行

|  |
| --- |
| npm install log4js |

#### 然后可以用这个组件来记录日志，它可以把日志记录一个文件里面

## urldemo.js

|  |
| --- |
| **let url = require('url') let log4js = require('log4js')  // 添加配置信息 log4js.configure({  appenders: { cheese: { type: "file", filename: "cheese.log" } },  // categories: { default: { appenders: ["cheese"], level: "error" } },  categories: { default: { appenders: ["cheese"], level: "info" } },  // categories: { default: { appenders: ["cheese"], level: "debug" } }, }); let logger = log4js.getLogger("cheese"); // let logger = log4js.getLogger(); // logger.level = "debug"   //1.parse方法，把一个网址解析为一个有很多属性的对象 // let urlString = "http://www.nodejs.org/some/url/?with=query&param=that#about" // console.log(url.parse(urlString)) // logger.error(url.parse(urlString)) // logger.info(url.parse(urlString)) //2 format方法，和上面的方法刚好反过来  jsonObj = {  protocol: 'http:',  slashes: true,  auth: null,  host: 'calc.gongjuji.net',  port: null,  hostname: 'calc.gongjuji.net',  hash: '#one#two',  search: '?name=zhangsan&age=18',  query: 'name=zhangsan&age=18',  pathname: '/byte/',  path: '/byte/?name=zhangsan&age=18',  href: 'http://calc.gongjuji.net/byte/?name=zhangsan&age=18#one#two' }  // console.log(url.format(jsonObj)) // http://calc.gongjuji.net/byte/?name=zhangsan&age=18#one#two  //3.resolve方法，有点想url拼接，但是如果base里面有路径，这个路径被覆盖 // console.log(url.resolve("http://www.httpbin.org",'get')) //http://www.httpbin.org/get // console.log(url.resolve("http://www.httpbin.org/get",'post')) //http://www.httpbin.org/post (/get被/post覆盖) // logger.debug(url.resolve("http://www.httpbin.org/get",'post'))   // 4.URLSearchPatams对象 let urlString = 'https://www.sogou.com/web?query=%E5%A5%B3%E5%B8%9D&\_asf=www.sogou.com'  // let urlString = "https://www.bing.com/images/search?q=car&qs=n&form=QBIDMH&first=1"  // let urlString = 'https://www.baidu.com:443/path/index.html?id=2#tag=3'  let urlParams = new URLSearchParams(url.parse(urlString).search)  logger.level='debug'  // logger.debug(urlParams)  logger.debug(urlParams.get('query'))  // logger.debug(urlParams.get('id'))  // logger.debug(urlParams.get('q'))   //URL类的使用  // let myurl = new URL('https://www.baidu.com:443/path/index.html?id=2#tag=3')  // console.log(myurl.searchParams.get('id')) // 2** |

##### 注意URLSearchParams对象的使用

## 日志文件cheese.log

|  |
| --- |
| [2023-10-11T18:33:32.685] [INFO] chsses - Url {  protocol: 'http:',  slashes: true,  auth: null,  host: 'www.nodejs.org',  port: null,  hostname: 'www.nodejs.org',  hash: '#about',  search: '?with=query&param=that',  query: 'with=query&param=that',  pathname: '/some/url/',  path: '/some/url/?with=query&param=that',  href: 'http://www.nodejs.org/some/url/?with=query&param=that#about' } [2023-10-11T18:48:22.894] [DEBUG] chsses - URLSearchParams { 'http://www.nodejs.org/some/url/?with' => 'query', 'param' => 'that#about' } [2023-10-11T18:52:02.649] [DEBUG] chsses - URLSearchParams { 'https://www.baidu.com:443/path/index.html?id' => '2#tag=3' } [2023-10-11T18:53:28.870] [DEBUG] cheese - URLSearchParams { 'https://www.baidu.com:443/path/index.html?id' => '2#tag=3' } [2023-10-11T18:53:28.875] [DEBUG] cheese - null [2023-10-11T19:04:52.019] [DEBUG] cheese - 2  [2023-10-11T19:11:34.155] [DEBUG] cheese - car [2023-10-11T19:14:09.094] [DEBUG] cheese - 女帝 |