**URL 类**

**new URL(input[, base])**

* input：要解析的 [绝对] / [相对] 的输入网址。如果 input 是相对的，则需要 base，如果 input 是绝对的，则忽略 base
* base：如果 input 不是绝对的，则为要解析的基本网址

const myURL = new URL('/Superman\_H', 'https://blog.csdn.net');

console.log(myURL.href); // https://blog.csdn.net/Superman\_H

**url.href**

获取、设置网址

const myURL = new URL('https://blog.csdn.net/Superman\_H?name=superman');

console.log(myURL.href); // https://blog.csdn.net/Superman\_H?name=superman

myURL.href = 'https://blog.csdn.net/Superman\_H';

console.log(myURL.href); // https://blog.csdn.net/Superman\_H

**url.hash**

获取和设置网址的片段部分

const myURL = new URL('https://blog.csdn.net/Superman\_H#superman');

console.log(myURL.hash); // #superman

myURL.hash = 'superCoder';

console.log(myURL.href); // https://blog.csdn.net/Superman\_H#superCoder

**url.search**

获取、设置网址的查询部分

const myURL = new URL('https://blog.csdn.net/Superman\_H?name=superman');

console.log(myURL.search); // ?name=superman

myURL.search = 'name=superCoder&age=21';

console.log(myURL.href); // https://blog.csdn.net/Superman\_H?name=superCoder&age=21

**url.searchParams**

此属性是只读的，但它提供的 URLSearchParams 对象可用于管理 URL 参数

**URLSearchParams 类**

**new URLSearchParams(param)**

**param 为 string**

let params;

// params = new URLSearchParams('?user=superman&age=21'); // 即前面有没有 ? 都 ok

params = new URLSearchParams('user=superman&age=21');

console.log(params); // URLSearchParams { 'user' => 'superman', 'age' => '21' }

console.log(params.get('user')); // superman

console.log(params.toString()); // user=superman&age=21

**param 为 obj**

const params = new URLSearchParams({

user: 'superman',

arr: ['First', 'Second']

});

console.log(params.getAll('arr')); // [ 'First,Second' ]

console.log(params.toString()); // user=superman&arr=First%2CSecond

**params.get(key)**

对于 URL 参数，通过 key 获取 value

const myURL = new URL('https://blog.csdn.net/Superman\_H?name=superman');

console.log(myURL.searchParams.get('name')); // superman

**params.append(key, value)**

添加 URL 参数

const myURL = new URL('https://blog.csdn.net/Superman\_H?name=superman');

myURL.searchParams.append('age', 21);

console.log(myURL.href); // https://blog.csdn.net/Superman\_H?name=superman&age=21

**params.delete(key)**

删除指定 URL 参数

const myURL = new URL('https://blog.csdn.net/Superman\_H?name=superman');

myURL.searchParams.delete('name');

console.log(myURL.href); // https://blog.csdn.net/Superman\_H

**params.set(key, value)**

设置指定 URL 参数的 value 值

const myURL = new URL('https://blog.csdn.net/Superman\_H?name=superman');

myURL.searchParams.set('name', 'superCoder');

console.log(myURL.href); // https://blog.csdn.net/Superman\_H?name=superCoder

**url 模块 (旧版)**

parse 方法：解析 URL 地址，返回该 URL 地址的信息对象。可通过信息对象的 query 属性获取 URL 参数信息

接收两个参数：URL 地址、解析格式  
解析格式：接收布尔值，false → URL 参数信息为 String；true → URL 参数信息为 Object

const url = require('url');

let strUrl = 'http://class/index?name=张三&age=18';

console.log(url.parse(strUrl).query); // name=张三&age=18

console.log(url.parse(strUrl, true).query); // { name: '张三', age: '18' }

**querystring 模块 (旧版)**

**String** 形式的数据 ↔ **Object** 形式的数据：  
① parse 方法：String → Object  
② stringify 方法：Object → String

const querystring = require("querystring");

let strUrl = "name=superman&password=123";

let obj = querystring.parse(strUrl);

console.log(obj); // { name: 'superman', password: '123' }

let str = querystring.stringify(obj);

console.log(str); // name=superman&password=123

**面试题**

**获取网址的 URL 参数**

let strUrl = 'https://blog.csdn.net/Superman\_H?name=superman';

**node.js 方法**

1. URL → 参数映射 → 参数值

const myURL = new URL(strUrl);

let paramsMap = myURL.searchParams;

let name = paramsMap.get('name')

// ---------------------------------------

console.log("paramsMap", paramsMap); // paramsMap { 'name' => 'superman' }

console.log("name", name); // name superman

1. URL → 参数字符串 → 参数映射 → 参数值

const myURL = new URL(strUrl);

let paramsStr = myURL.search;

let paramsMap = new URLSearchParams(paramsStr);

let name = paramsMap.get('name');

// -----------------------------------------

console.log("paramsStr", paramsStr); // paramsStr ?name=superman

console.log("paramsMap", paramsMap); // paramsMap { 'name' => 'superman' }

console.log("name", name); // name superman

1. URL → 参数对象 → 参数值

const url = require('url');

let paramsObj = url.parse(strUrl, true).query;

let name = paramsObj.name;

// -----------------------------------------

console.log("paramsObj", paramsObj); // { name: 'superman' }

console.log("name", name); // superman

1. URL → 参数字符串 → 参数对象 → 参数值

const querystring = require('querystring');

const myURL = new URL(strUrl);

let paramsStr = myURL.search.slice(1);

let paramsObj = querystring.parse(paramsStr);

let name = paramsObj.name;

// -----------------------------------------

console.log("paramsStr", paramsStr); // paramsStr name=superman

console.log("paramsObj", paramsObj); // paramsObj { name: 'superman' }

console.log("name", name); // name superman

1. URL → 参数字符串 → 参数对象 → 参数值

const querystring = require('querystring');

const url = require('url');

let paramsStr = url.parse(strUrl).query;

let paramsObj = querystring.parse(paramsStr);

let name = paramsObj.name;

// -----------------------------------------

console.log("paramsStr", paramsStr); // paramsStr name=superman

console.log("paramsObj", paramsObj); // paramsObj { name: 'superman' }

console.log("name", name); // name superman

**原生 JS**

let strUrl = 'https://blog.csdn.net/Superman\_H?name=superman';

1. 使用 split() 方法

function GetQueryValue(strUrl, queryName) {

let query = strUrl.split('?')[1];

let paramsArr = query.split("&");

let value = null;

paramsArr.forEach(item => {

let keyValue = item.split('=');

if (keyValue[0] == queryName) {

value = keyValue[1];

}

});

return value;

}

console.log(GetQueryValue(strUrl, "name"));

1. 使用正则：

function GetQueryValue(strUrl, queryName) {

let params = strUrl.split('?')[1];

let reg = new RegExp("(^|&)" + queryName + "=([^&]\*)(&|$)");

let result = params.match(reg);

console.log("result", result);

return result ? result[2] : null;

}

console.log(GetQueryValue(strUrl, "name"));

(^|&) 意思是从头开始匹配字符 &，=([^&]\*) 意思是匹配 = 后面 0~n 个不是&的字符，直至碰到第一个 & 为止  
在正则表达式中，增加一个 () 代表着匹配数组中增加一个值，因此代码中的正则匹配后数组中应包含 4 个值

result [ 'name=superman', '', 'superman', '', index: 0, input: 'name=superman', groups: undefined ]

* 1