Accessing parts of the URL

**This is important because in a get request the form/query data is encoded at the end of the URL of the request.**

**Note:** My main file is index.js
 It is a known problem that sometimes app.js will lead to a 'module not found' error – even tho' it is there. *It appears that the issue is naming the folder and the module (i.e. the main file for the entry point) with the same name.*

**Here is the code:**

//This app shows examples of how URLs may be examined in Node.

//Examples and documentation is at https://nodejs.org/docs/latest/api/url.html

//The documentation says "

//The URL constructor is accessible as a property on the global object.

//It can also be imported from the built-in url module:

//The folllowing are equivalent: URL and require('url').URL

//So the next line is not needed:

//const url = require('url');

const chap16url = 'http://web.simmons.edu/~menzin/CS321/Unit\_8\_Server\_Side\_Node/';

const ch16 = **new URL**(chap16url); //this is the chap16url as an object, making use of the url module

console.log("url.href is: " + ch16.href);

console.log("url.host is: " + ch16.host);

console.log("url.origin is: " + ch16.origin);

console.log("url.pathname is: " + ch16.pathname);

console.log("url.port is: " + ch16.port);

console.log("url.search is: " + ch16.search);

console.log("url.toJSON is: " + ch16.toJSON);

//Repeat all this when search at duckduckgo for node tutorial

const ntut = 'https://duckduckgo.com/?q=node+tutorial&t=hd&va=u&ia=web'

const nt = **new URL**(ntut); //this is the chap16url as an object, making use of the url module

console.log("url.href is: " + nt.href);

console.log("url.host is: " + nt.host);

console.log("url.origin is: " + nt.origin);

console.log("url.pathname is: " + nt.pathname);

console.log("url.port is: " + nt.port);

console.log("url.search is: " + nt.search);

console.log("url.searchParams is: " + nt.searchParams);

console.log("url.searchParams.get('q') is: " + nt.searchParams.get('q'));

console.log("url.toJSON is: " + nt.toJSON);

**Here is the run: (//The comments in red are from me, not from the run.)**

**C:\myNode\formData\urlData2>node index**

url.href is: http://web.simmons.edu/~menzin/CS321/Unit\_8\_Server\_Side\_Node/

url.host is: web.simmons.edu

url.origin is: http://web.simmons.edu

url.pathname is: /~menzin/CS321/Unit\_8\_Server\_Side\_Node/

 //Notice index.html is not shown!

url.port is:

url.search is:

url.toJSON is: function toJSON() {

 return this[kFormat]({});

 }

url.href is: https://duckduckgo.com/?q=node+tutorial&t=hd&va=u&ia=web

url.host is: duckduckgo.com

url.origin is: https://duckduckgo.com

url.pathname is: /
 //In our server we also got this as req.url

url.port is:

url.search is: ?q=node+tutorial&t=hd&va=u&ia=web
 //Notice the ? at the start of the query string

url.searchParams is: q=node+tutorial&t=hd&va=u&ia=web
 //Notice there in NO ? at the start of the query string

 //So using searchParams, rather than search, gets rid of the ?
 //But the get method on what searchParams returns allows us
 // to access specific key-value pairs. See next line.

url.searchParams.get('q') is: node tutorial
 // this is a string (with a blank space in the middle)

url.toJSON is: function toJSON() {

 return this[kFormat]({});

 }

**To see this in action**

To see this in action, go to <http://web.simmons.edu/~menzin/CS321/Unit_8_Server_Side_Node/SimpleProjects/> and open <http://web.simmons.edu/~menzin/CS321/Unit_8_Server_Side_Node/SimpleProjects/ConsoleLoggingFrom_GET/PrototypeVerySimpleFormGET_v5.html> the file in your browser.

Now open the app.js file in VS code, in a folder (say name the folder postAccess) and in the terminal navigate to the folder. As usual, in the folder type npm init, and then node app.
Now go back to the form in your browser, fill it out and submit it.

**The code for the app.js is (w/o all the comments)**

var http = require('http');

var fs = require('fs');

var url = require('url'); //may need url-parse module instead

http.createServer(function(req, res) {

 //grab firstName and lastName and passwd from request
 // and put them in vars fName, lName, passGood

 var method = req.method; //can test for GET or POST

 //Should check to see that method is GET -
 //Otherwise need to look at the req body to find these key-value pairs.

 console.log('new version')
 let q = req.url;
 console.log('q is '+q)

 //The next 3 lines use code which is now legacy
 var queryData = url.parse(req.url, true).query;
 //the query string is in the queryData object if it was a GET

 console.log(queryData); //may need to stringify this

 console.log(queryData.firstName)

}).listen(320**);**

**My terminal shows:**

PS C:\myNode\simpleExamples> cd ConsoleLoggingFrom\_GET

PS C:\myNode\simpleExamples\ConsoleLoggingFrom\_GET> node app

new version

**q is /?firstName=Sally&lastName=Simmons&passwd=Please+enter+your+password**

[Object: null prototype] {

 firstName: 'Sally',

 lastName: 'Simmons',

 passwd: 'Please enter your password'

}

Sally

Here I have bolded the interesting part --- namely what req.url is

At this point you have a choice ---

* you could use old fashioned JavaScript to turn this into a set of variables, or an object:
That is, you would use substring() to get rid of the leading **?** for the result of a search or the leading **/ ?** leading, or leading **/someRoute/** **?**  if we start with req.url
We can find the index or the ? in req.url and use substring() to obtain everything after it.
Then you would split() the result on & to get the key-value pairs in an array.
Then for each such pair you would split () it on = to get the key and value and use them to form an object.
Finally you may need to use ***decodeURIComponent ()*** function  to decode any values which have been URI-encoded. (This function leaves alone parts that are already decoded.)
At this point you can write the code, but as you will see soon there are easier ways to do this. If sloth strikes, you can find code at <https://www.youtube.com/watch?v=VQYMXX02EOQ> or <https://www.developerdrive.com/turning-the-querystring-into-a-json-object-using-javascript/> or <https://blog.kevinchisholm.com/object-oriented-javascript/how-to-turn-the-query-string-into-a-javascript-object/> (Some of these start with window.location as string to operate on --- of course it has the name-value pair appended for a get request.)

 Or

* you could join this with a dummy url and then use the methods near the top of this page – for instance
 dummy = <http://something.com> + q;
 ourFakeURL = new URL(dummy);
 let lastName = ourFakeURL.searchParams. get(lastName); //etc
This second method is recommended by Node and by Stackoverflow (altho' they use a real URL in place of dummy).

Or

* you could use the methods of URL searchParams (my recommendation) .

Here is the code, with the important lines bolded:
**const url = require('url');**

const aLocation = new URL("https://nodejs.org/docs/latest/api/url.html#url\_urlsearchparams\_tostring");

**const anOnlineQuery = new URL("https://duckduckgo.com/?q=what+day+is+today&t=hc&va=u&ia=web");**

console.log("The hash for aLocation is " +aLocation.hash);

**//Examples based on https://nodejs.org/docs/latest/api/url.html#url\_class\_urlsearchparams**

//First we log the whole query

console.log("The query from anOnlineQuery is "+anOnlineQuery.searchParams);

//Then we iterate thru all the name-value pairs
**anOnlineQuery.searchParams.forEach((value, name, searchParams) => {
 console.log(name, value,
 anOnlineQuery.searchParams === searchParams);
 });**

//Of course, when you write a server you know the names of the pairs which the application returns

//The get method of searchParams will retrieve the first value associated with a key/name

//If there are multiple values, use getAll() to have them put in an array

console.log("The value of q from our search params is given by")
**console.log(anOnlineQuery.searchParams.get('q') );**
My comment: the forEach method of searchParams will take as may arguments as you wish. Notice, however, that value and name have a specified order and refer to exactly what you want them to.

And here is the run:

The hash for aLocation is #url\_urlsearchparams\_tostring
The query from anOnlineQuery is q=what+day+is+today&t=hc&va=u&ia=web

q what day is today true

t hc true

va u true

ia web true

The value of q from our search params is given by

what day is today

My comment: Notice that the url encoding ( such as a + for a space) has also been taken care of.

Several sites also provide similar code for this – the Node documentation referenced in the code, <https://attacomsian.com/blog/javascript-convert-query-string-to-object> , <https://www.javascripttutorial.net/javascript-bom/javascript-get-query-string/> <https://masteringjs.io/tutorials/fundamentals/query-string> or <https://flaviocopes.com/urlsearchparams/>

Here is the code for ConsoleLoggingFrom\_GET app file:
In it you will see that we use the req.url to make a new fake URL and then we can use seartchParams on it.
This code is also on the web site.

var http = require('http');

var fs = require('fs');

var url = require('url');   //may need url-parse module instead

http.createServer(function(req, res) {

    //grab firstName and lastName and passwd from request

  // and put them in vars fName, lName, passGood

  var method = req.method;   //can test for GET or POST

  //Should check to see that method is GET - otherwise need to look at the req body to find these key-vale pairs.

  let q = req.url;

  console.log('q is '+q);

  fakeQ = new URL("http://dummy.com"+q);

  //Strictly speaking we should use the path module and

  //fakeQ = path.join("http://dummy.com", q)

  console.log("fakeQ is "+fakeQ);

  console.log('Our searchParams object is ' + fakeQ.searchParams)

  console.log('Now to use these values');

  let fName = fakeQ.searchParams.get('firstName');

  let lName = fakeQ.searchParams.get('lastName');

  console.log("We have ", fName, lName);

}).listen(320);