**How we access data from a post**

The best (but rather sophisticated) explanation of what happens with a post is at <http://book.mixu.net/node/ch10.html> (although it was written before url.parse() became a legacy.)

In it, Mixnu explains that when you set up a server the Request Object ( = what you get back from the server = the first parameter in our callback function = aka the IncomingMessage) is a ***readable stream***. We'll talk more about streams soon, but here the important items are: think of a stream as data on the move; files are extensions of streams; streams come with built-in events – especially the ***data*** and ***end*** events. This means that we may attach event handlers to those events.

When you are handling a post request, the common code looks like:  
  
function parsePost(req, callback) {

var data = '';

req.on('data', function(chunk) {

data += chunk;

});

req.on('end', function() {

callback(data);

});

}

(Lots of folk use body as the name of the var which will hold what is sent back in the POST. It will hold the body of the POST message. ) The callback(data) is what you will do with the data once you have it all --- i.e. extract the key-value pairs.

You can find a similar example at <https://www.tutorialspoint.com/parsing-request-body-in-node> (I don't know why they chose to put the data in an array and then concatenate all the pieces.) In the next to the last black box at <https://codezup.com/handle-process-http-post-request-data-node-js-tutorial/> you will see an example (a little odd) where they used a server to write a page that has a form. The form is posted back to the server, the server puts all the data together and then writes it back to the same page. When you look at that page you can see that what was written back (i.e. what was accumulated in the body variable) is the key-value pairs from the POSTing. Of couse, at this point you can split is on the &s to obtain the key-value pairs, etc.

To see this in action, go to <http://web.simmons.edu/~menzin/CS321/Unit_8_Server_Side_Node/SimpleProjects/> and then <http://web.simmons.edu/~menzin/CS321/Unit_8_Server_Side_Node/SimpleProjects/ConsoleLoggingFrom_POST/> . Then open the index2.js file <http://web.simmons.edu/~menzin/CS321/Unit_8_Server_Side_Node/SimpleProjects/ConsoleLoggingFrom_POST/index2.html> 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 app2** (This is app2 and it works with the form at index2 b/c I was originally going to put them all in the same project. Also app2 listens on a diffferent port from the other get app)

const http = require('http');  
const fs = require('fs');  
const url = require('url');

http.createServer(function(req, res) {  
 //grab firstName and lastName and passwd from request  
 // and put them in vars fName, lName, passGood

let method = req.method; //can test for GET or POST  
 //Should check to see that method is POST - otherwise need to look at the   
 // URL to find these key-value pairs.

//We now accumulate the data sent to us in the POST

let body ="";  
 req.on('data', (data) => {  
 body += data;});  
 req.on('end', () => {console.log(body)})

}).listen(321);

**And here is what my terminal says:**PS C:\myNode\simpleExamples\ConsoleLoggingFrom\_POST> node app2

**firstName=Happy&lastName=Student&passwdGood=%7BpassGood\_from\_Form\_1%7D**

You can see that body now holes the key-value pairs (the last one is URI encoded and may need to be run thru the decodeURIComponent function, which is a standard JavaScript function. (The %7B's and %7D come from the { } that are lingering in the form inputs, for no good reason.)  
  
We are now back exactly where we were with the get value pairs --- either use string manipulation to extract the key-value pairs or make up a dummy URL, append this with a ? as the query string, run it through **new URL**, and then use searchParams and searchParams. get(*some key*) to find the values.

**Warnin**g **body was defined inside the createServer function --- if you need it outside that function then declare body = "" beforehand.**