Ex3: Internet 2016/2017



* Read node.js stuff
  + the tutorial: <http://nodebeginner.org/> and <http://www.devshed.com/c/a/JavaScript/JavaScript-Exception-Handling/> and <http://net.tutsplus.com/tutorials/javascript-ajax/introduction-to-express/>
  + Go over the docs: <http://nodejs.org/api/>
    - specifically <http://nodejs.org/docs/latest/api/net.html> and <http://nodejs.org/api/fs.html>
  + How to write module: <https://www.sitepoint.com/understanding-module-exports-exports-node-js/> or <https://www.gitbook.com/book/kevinchisholm/njsc-creating-your-first-node-js-module/details>
  + Watch the video: <http://goo.gl/asGxZ>
* Understand HTTP, you should support the entire protocol, but you should understand the basic syntax, first line, headers, body and what separates them apart.
  + <http://www.jmarshall.com/easy/http/>
  + <http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol>
    - Review the static request part of this peregraph <http://en.wikipedia.org/wiki/Web_server#Path_translation>
* In this exercise you will develop a **HTTP(version 1.1) server** **module** on top of node.js, simpler version of ExpressJS. **Module** is a node.js component that other developers can ‘import’ and use (in our case they will use it to start their own HTTP server).
* **HTTP server** is a software that receives HTTP request, run one or more functions on the request object that represents the HTTP request text and reply with HTTP response.
  + You should build a new node.js module <https://nodejs.org/api/modules.html>
    - The module ‘hujiwebserver’ exposes:
      * .*start*(port,callback(err)) method returns serverObj which starts the server and upon readiness to accept HTTP requests call the callback function.
        + In case the server could not start it should execute the callback with your string err argument that contains the error reason.
        + *serverObj* should have a

.*stop*() function that stops listening

*port* property (number)

* + - * .*use*(command,middleware(request,response,next)) method:
        + Return the hujiwebserver object, it only means that the use function should ‘return this;’ <--new instruction
        + *command* is the prefix of the URL command/resource (the URL portion that can be found right after the domain, highlighted here: [http://www.x.com**/a/b/g**?dfdf=dd](http://www.x.com/a/b/g?dfdf=dd)) that you would like to handle via the middleware

for example the command ‘/x/y’ means that you should handle any *domain/x/y\** resources

commands can be parameterized, the parameters must be between two ‘/’s or at the end of the command. the parameter name starts with the colon symbol

e.g. the commands /x/:y/z handles resources that looks like /x/\*/z including /x/dsfsdg/z/sfdgfg and /x/e/z

the commands argument of the use method is **optional,** in case the .use function receives only one argument, you will set the commands to be ‘/’ i.e. it **should handle any request**.

e.g. .use(function(rq,rs,nxt){...}) should handle any request.

* + - * + *middleware*(request,response,next) is a function that receives 3 arguments, request, response and next

*request* is the object that represents the HTTP request

it should have the following properties. params, query, method, cookies, path, host, protocol, get(), param() and is() . the description of each property can be found here: <http://expressjs.com/api.html#req.params>

Note: ExpressJS named ‘host’ as ‘hostname’

in addition to a .body property that consists of the http request body, in case there is no body, it should be set to null.

You should support GET/POST/PUT/DELETE/OPTIONS/TRACE HTTP methods

*response* is the object that represents the HTTP response

it should have the following properties. set(), status(), get() , cookie(), send() and json() . the description of each property can be found here: <http://expressjs.com/api.html#res.status>

regarding res.send() you do not have to support Buffer as an argument

*next*() is a function that hints your module to lookup for the next relevant middleware.

* + - * In general upon calling the .use() function, your server job is to register the command+middleware in some data structure ( e.g. array)
        + upon HTTP request you should walk the registered middlewares one by one by the order of their insertion. the first handler that matches the real request URL to the middleware’s command should get executed with the relevant request, response and next args. E.g. mw(rq,rs,n)

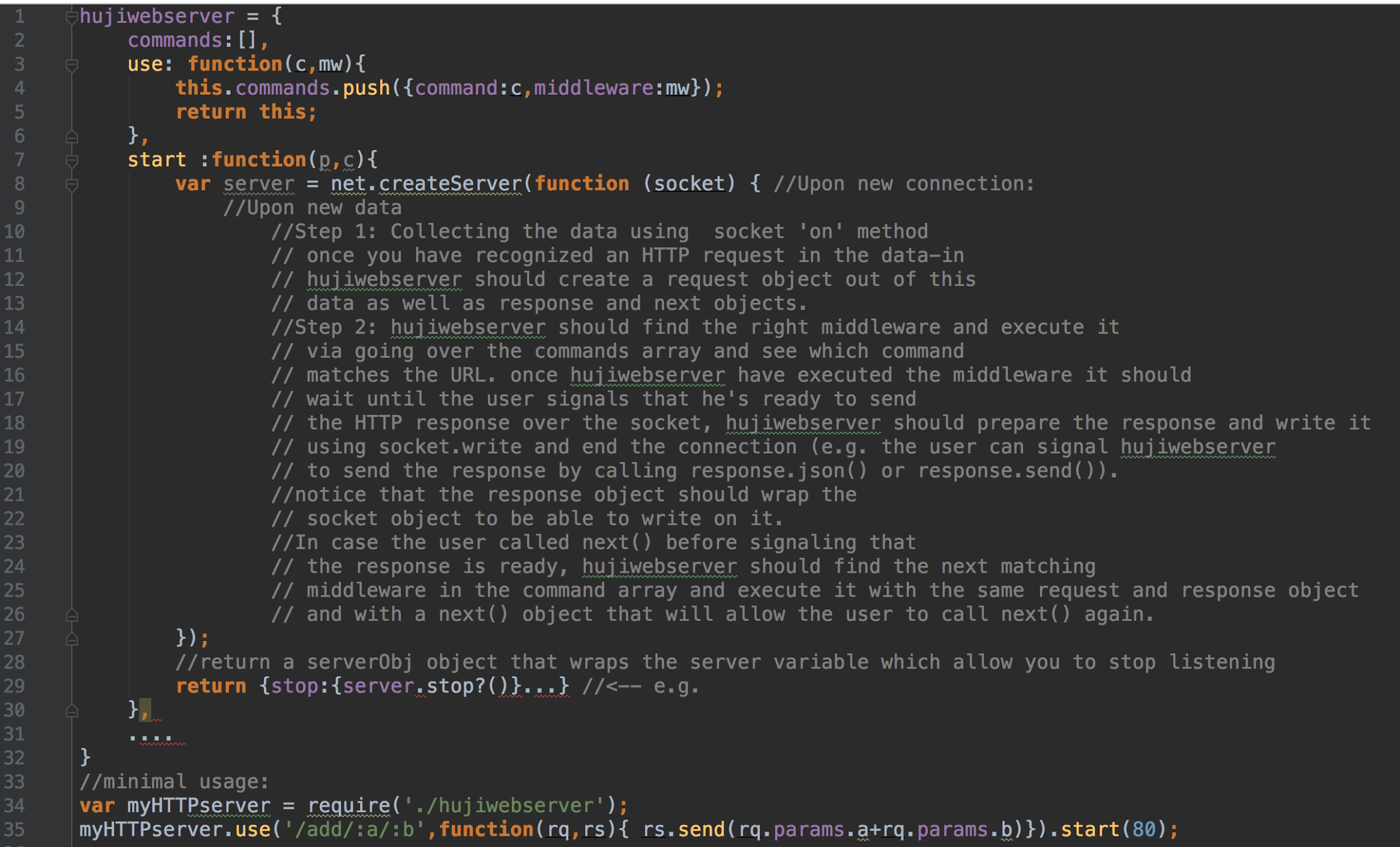
while executing the first middleware, only if the next() function has been called you should start walking again on the middlewares data structure in order to find the next middleware to execute if there is any.

In case the user called send() or json() you should write the HTTP response to the socket and end the connection <https://nodejs.org/api/net.html#net_socket_end_data_encoding>

in case none of the middlewares match the URL or none of them call .send() or .json() after a 10 seconds timeout the response you should return is 404

in case of any exception that is not treated or any other error you should return 500

* + You can add any methods to any of those modules or additional modules.
  + You MUST end the connection after sending the HTTP response socket.end() will close the connection or after a 25 seconds timeout
    - you should **not** support HTTP persistence (keep-alive mechanism):
      * Read: <http://en.wikipedia.org/wiki/HTTP_persistent_connection> to learn more
  + You can assume that when an http request contains a body, it contains a content-length header that specifies the length of the body.
  + Security instructions
    - Make sure that if something bad happen to the processing of one request it won’t crash the entire server. (DOS <https://en.wikipedia.org/wiki/Denial-of-service_attack>)
* Create a tester **test.js** (and add it to the EX zip ) that calls your module and starts it on port 8080. It registers the following commands/middlewares
  + /hello/world should return string hello world with content-type text/plain
  + /add/:n/:m should return the json: {result:n\*m} with content-type application/json
  + /filez/\* should return the file \* from the *filez* folder (that you should add to the EX zip).
    - \* can be recursive e.g. /dgsd/dfdf/sfsfs.html <- in this case you should read the <your main js file path>/filez/dgsd/dfdf/sfsfs.html
    - You should only support .html .css and .js files. You should return the right content-type for those :
      * \*.js: JavaScript : application/javascript
      * \*.html: HTML: text/html
      * \*.css: CSS: text/css
* Additional **important** instructions:
  + You are allowed to **create additional node modules** and to have as many files as you wish.
  + You are **not** allowed to utilize the ‘http’ module, you should use the [‘net’](http://nodejs.org/docs/v0.6.2/api/net.html)  and the ‘fs’ module as your infrastructure.
  + You are **not** allowed to use any external node.js files nor plug-ins without asking us first.
  + Try to **minimize** the number of **global** javascript variables as possible.
  + Consider **hoisting** and write you code hoisting-ready (variables should be located at the top of their scope)
  + Keep in mind that this is a web-server, it should be able to serve thousands of concurrent requests. Pay attention to performance issues.
    - **Don’t** do any I/O operation in a **consequential** manner.
  + **You are allowed to do this exercise in pairs or solo.**
  + **Each student should submit his/her own zip file**
  + Compress all your files and submit **fullName.ID9digists.ex3.zip**
    - Add a partner.<partnerID-firstNameEng-lastNameEng/NoPartner).txt file to the zip
    - Add a readme.txt file to the zip the describes (1) What was hard in this ex? (2) What was fun in this ex? (We won’t reduce points in case this part is empty) (3) how did you test your server and include as details result as much as possible
    - We will test your code utilizing Node.js v7.4.0, [download here](https://nodejs.org/en/)
* **Submission date: 16/1/2017 23:55 pm**
* **Take a look at this very high level code, this is just a bootstrap example:**



* Extra reading
  + <http://en.wikipedia.org/wiki/Nodejs>
  + <http://en.wikipedia.org/wiki/C10k_problem>
  + <http://oreilly.com/openbook/webclient/ch03.html>
* Good luck

