CSCI-UA.0480-001 FA16 Midterm v1 - Evergreen

Jessica M Donahue

TOTAL POINTS

81 / 118

QUESTION 1

1 1 7 / 9

- 0 Correct
- 0.5 1 - Request
- 1 1 - GET /pies/special HTTP/1.1
- 0.5 2 - Response
- 1 2 - HTTP/1.1 301 Moved Permanently
- **1 2 - Location: /extra/pineapple**
- 0.5 3 - Request
- 1 3 - GET /extra/pineapple HTTP/1.1
- 0.5 4 - Response
- 1 4 - HTTP/1.1 200 OK
- **1 4 - Content-Type: text/html**
- 1 4 - <h2>you haz a pineapple</h2>
- 0.5 4 - empty line ... then body / body is not a header

QUESTION 2

2 2 3.5 / 5

- 0 Correct
- 1 A) set max-age or expires cookie options for Set-Cookie header in http response from server
- 0.5 A) ^^^ partially correct answer (setting a new cookie doesn't actually delete, no delete option for Set-Cookie, doesn't mention headers, etc.)
- 1 B) prevents client side JavaScript from reading cookie values (when you don't want 3rd party scripts to read cookies)
- 0.5 B) ^^^ partially correct answer
- 1 C) Sends cookies back on every request for that domain using the Cookie header
- **0.5 C) ^^^ partially correct answer**
- 1 When no cookies are sent, no session id in cookies sent or session id is not in session store. Some situations include... after clearing cookies, switching to a different browser, etc.
- 0.5 D) ^^^ partially correct answer
- **1 E) can be "stolen" easily (shoulder surfing, logs, bookmarks), looks like it may be sequential, easily guessable**
- 5 blank

QUESTION 3

3 3 13.5 / 20

- 0 Correct
- 2 a) undefined
cactus
undefined
undefined
- **0.5 a) 1 incorrect**
- 1 a) 2 incorrect
- 1.5 a) 3 incorrect
- 0.5 a) incorrect order
- 2 b) playerInfo.apply(player,[true]);
- **0.5 b) mentions array, but array not passed in as 2nd argument / 1st arg passed in as array instead of 2nd**
- 1 b) partial credit - true not in Array or this treated as Array
- 2 c) #3 #5 #7 #9
- 1 c) ^^^ partially correct
- **2 d) function,**
object
- 1 d) 1 out of 2 types correct
- 2 e)
error, format isn't a
function (declaration
is hoisted, but
definition is not)
- 2 f)
var MediaTypes = {

css:'text',
jpg:'image'
};
type = mediaTypes[ext];
- 1 f) attempt to use ternary or switch, but does not actually achieve same functionality as code in 1st column
- 2 g)
3
undefined
false
- 0.5 **g) 1 incorrect**
- 1 g) more than 1 incorrect
- 2 h)
Function.prototype
Object.prototype
- 1 **h) 1st object is incorrect, but 2nd is prototype of 1st**
- 1 h) missing .prototype for each
- 2 i)
browser, curl, nc, client libraries like request / net module / http module,
etc.
- 1 i) only 1 of 2 correct
- 2 **j) var s = net.createServer(**
this.fn.bind(this)
)
- 1 j) use bind instead of call
- 1 j) ^^^ partially correct
- 0.5 j) used bind in correct line, but usage not correct

QUESTION 4

4 4 **4 / 4**

- 0 **Correct**
- 0.5 GET
- 0.5 POST
- 0.5 GET - url / query
- 0.5 POST - request body
- 0.5 GET - req.query
- 0.5 POST - req.body
- 1 POST to create, GET to read, GET for bookmark-able urls
- 0.5 ^^^ partially correct or not enough detail (for example, "hiding" isn't quite accurate because data can still be seen in request body)
- 4 blank / incorrect

QUESTION 5

5 5 **2 / 4**

- 0 Correct
- 1 document: name value pairs / an "object"
- 0.5 ^^^ document: partial credit
- 1 collection: a group of documents
- 0.5 ^^^ collection: partial credit
- 1 **database: a group of collections**
- 0.5 ^^^ database: partial credit
- 1 **local file system, cloud, on client via cookies, in-memory**
- 0.5 ^^^ 1 / 2 correct

QUESTION 6

6 6 **3 / 4**

- 0 Correct
- 1 called res.render in callback
- 0.5 1st arg is name of template
- 0.5 **second arg is present and is an object**
- 0.5 **2nd arg prop name is used as variable in template**
- 0.5 has surrounding markup
- 0.5 #each
- 0.5 prints cat's name and lives using: this.name and this.lives or just name and lives

QUESTION 7

7 7 **0 / 18**

- 0 Correct
- 18 **DO NOT GRADE**
- 18 BLANK
- 2 app.use
- 2 res/req, next in signature
- 1 missing next in signature
- 1 path includes "static"
- 1 path uses req.path
- 2 readfile callback exists

- 1 readfile callback doesn't have correct parameters
- 2 call next if error
- 2 extract extension / get content type
- 2 set content type on response
- 2 use send to send back data
- 1 wrote data, but did not send response

QUESTION 8

8 8 18 / 18

- 0 Correct
- 18 DO NOT GRADE
- 2 mapWith: returns new function
- 2 mapWith: does not create new array
- 2 mapWith: new function should return an array
- 2 mapWith: new Array should consist of all values created from calling old function on incoming array
- 2 any: does not correctly determine if test passes for at least one element (ex: resetting eventual return val to false while iterating)
- 2 any: does not return boolean
- 1 both: did not use 2nd hof / repeated 1st one
- 2 both: did not use any hof

QUESTION 9

9 9 12 / 18

- 0 Correct
- 18 DO NOT GRADE
- 1 function Request (constructor)
- 1 signature correct
- 2 parse out domain
- 2 parse out path
- 1 path and domain: find index of / ... then slice
- 1 path and domain parsing - handle no trailing slash
- **1 does not work with more than one nested directory**
- 2 all other props are set
- 2 define send on prototype
- 2 makes connection
- 1 does not use object's props to connect
- **1 connect should be in send**
- **2 should create a valid http request in connect and write it**
- **1 write should occur on connection**
- 2 does'nt handle “this” in connect callback function
- 1 attempted to use bind or "capture" this to fix this problem, but solution not entirely correct
- 2 logs out response / does not point to client code that listens on data
- **1 Point adjustment**
  - -1: has on data, but doesn't log out response

QUESTION 10

10 10 18 / 18

- 0 Correct
- 18 BLANK
- 18 DO NOT GRADE
- 1 +1 global array of lines
- 0.5 ^^^ initialize storage to empty list or strings
- 0.5 ^^^ method missing or incorrect (should be POST)
- 1 +1 app.get /poem
- 2 +2 renders index with lines
- 1 +1 app.post /poem
- 1 +1 checks for req.body adverb and verb
- 2 +2 add both to global array
- 2 +2 redirect back to GET /poem
- 2 +2 otherwise, re-render with error
- 1 ^^^ error implementation only partially correct or error is global, so same error will show up for all users (store error in session or re-render page with error in post instead)
- 2 +2 template should iterate through poem lines
- 1 +1 template displays error message
- 1 +1 template: form with correct action and method
- 0.5 ^^^ form should be post
- 1 +1 template: form element 1 w/ name
- 1 +1 template: form element 2 w/ name
- 1 ^^^ missing name for both inputs
- 0 minor errors (names are the same, attribute name should be action for form, used send instead of render, compare form to null or undefined instead of empty string, extra lines after render, etc.)
- 4 post logic in get handler; should be moved into

post, form should be post

great work!

Net ID: JMD732 Name: Jessica Donahue

CSCI-UA.0480-001 – Applied Internet Technology

Midterm Exam

October 20th, 2016

Instructor: Joseph Versoza

Keep this test booklet closed until the class is prompted to begin the exam

- Computers, calculators, phones, textbooks or notebooks are **not allowed** during the exam
- Please turn off your phone to avoid disrupting others during the exam
- Some questions allow you to make assumptions about what's already written or to use **shorthand for html by omitting close tag**, so read the instructions carefully
- Tear off the last page – it can be used as scrap/scratch paper and as a reference
- Choose to skip one of the last 4 questions (7 through 10) by marking it as DO NOT GRADE

1. Write out the full **HTTP interaction** involved in the following scenario. (9 points)

   a) A **user enters** the following **URL in the URL bar** of their **browser**: http://asliceof.pizza/pies/special.
   b) The browser ends up displaying a page, but the page that it goes to is: http://asliceof.pizza/extra/pineapple.
   c) The resulting page only has the following html: <h2>you haz a pineapple!</h2> (!?)
   d) (The browser behaves as if the **server told** the **browser** that it should **go to a page other than the one requested**).

   In the tables below, write out the HTTP requests and responses (there are 4 total) that transpired:

   - Specify "**request**" or "**response**" in the **first row**, and write the **entire HTTP** request or response **below** it
   - Write the **requests** and **response** in the **order that they occur** starting off with the initial request from the browser (note that the table columns are numbered)
   - Assume version HTTP/1.1
   - **Ignore** any requests/responses related "static" resources, such as **favicons**, **css**, **images**, etc. ...
   - If there are multiple valid status codes, any of those codes can be used
   - Only add the headers that are necessary for the interaction specified above

Request or Response?	1 request	2 response
Entire Source of Request or Response	GET /pies/special HTTP/1.1	HTTP/1.1 301 Redirect you haz a pineapple

Request or Response?	3 request	4 response
Entire Source of Request or Response	GET /pineapple HTTP/1.1	HTTP/1.1 200 OK <h2> you haz a pineapple

2. Give a short, one-sentence answer to the following questions about cookies and session management. (5 points)

   a) How can the server inform the browser that it would like to delete cookies (be as specific as possible; include any references necessary for HTTP requests/responses)?
   It can set an expiration date for the cookie or it can kill the server.

   b) Why would you use the `HttpOnly` option for a cookie?
   HTTPOnly means the cookie can only be read in the HTTP request object and not in JavaScript.

   c) Once a cookie is created by the browser for a particular domain, what happens on every subsequent request to that domain from that browser (be as specific as possible; include any references necessary for HTTP requests/responses)?
   Once a cookie is created it sets that cookie onto that HTTP request for that domain. Then for every subsequent request it checks to see if the request has a cookie. If the request doesn't have a cookie, it creates a cookie for that request and sets it in the request

   d) Describe two situations where the server may create a new session id and send it back to the client:
   (1) if the client doesn't have a session id
   (2) if the session id expired

   e) Imagine that you saw ?session_id=12 in the url of your browser. What makes this a terrible idea for transmitting session id?
   This is incredibly insecure because then any client can steal your session id which could be held in a session store management system that holds private information like passwords and credentials

3. **Read the code** in the left column. **Answer the questions** about the code in the right columns. Show your work if possible (for partial credit) (20 points)

Code	Question #1	Question #2

`function playerInfo(includeScore) {`
`  console.log(this.name);`
`  if(includeScore) {`
`  console.log(this.score);`
`  }`
`}`
 	
`var player = {name: 'cactus'};`
`playerInfo(false);`
`playerInfo.call(player, true);`
`var info = new playerInfo(false);`
	a) What is the output of this program (error or nothing are possible, explain why if error)?
	cactus
	undefined
	undefined
		b) **Rewrite** the **2nd to last line** (`playerInfo.call`) in the space below so that it uses **apply** instead of `call`.
		playerInfo.apply(
		[player], true)

`var makeCounter = function() {`
`  var count = 1;`
`  return {`
`  inc: function() {`
`  count += 2;`
`  },`
`  print: function() {`
`  this.inc();`
`  console.log('#' + count);`
`  }`
`  };`
`};`
`var counter = makeCounter();`
`for(var i = 0;i < 4; i++) {`
`  counter.print();`
`}`
	c) What is the output of this program (error or nothing are possible, explain why if error)?
	3
	5
	7
	9
		d) What would that output of these two lines be if they were added to the end of the code in the first column?
		`console.log(typeof makeCounter);`
		`console.log(typeof counter);`
		function
		function

`function extractMediaType(f) {`
`  var i = f.lastIndexOf('.');`
`  var ext = f.substring(i + 1);`
`  var type;`
`  if(ext === 'css') {`
`  type = 'text';`
`  } else if (ext === 'jpg'){`
`  type = 'image';`
`  }`
`  console.log(format(type, ext));`
 	
`  var format = function (t, s) {`
`  return t + '/' + s;`
`  };`
`}`
`extractMediaType('wavey.css');`
	e) What is the output of this program (error or nothing are possible, explain why if error)?
	text/css
		f) Rewrite the code that sets the variable, `type`, (based on an extension) so that it **doesn't** use an if / else if statement.
		var types = {'css': 'text',
		'jpg': 'image'}
		var type = types[ext]

`var arr1 = [1, 2, 3];`
 	
`Array.prototype.last = function() {`
`  if(this.length > 0) {`
`  var i = this.length - 1;`
`  return this[i];`
`  }`
`};`
`console.log(arr1.last());`
 	
`var arr2 = [];`
`console.log(arr2.last());`
 	
`var r = arr2.hasOwnProperty('last');`
`console.log(r);`
	g) What is the output of this program (error or nothing are possible, explain why if error)?
	3
	false
		h) What is the **[[prototype]]** of `Array.prototype.last`? What is the **[[prototype]] of the [[prototype]]** of `Array.prototype.last`?
		1. Object.Prototype
		2. null

`function App(port, connectMsg) {`
`  this.msg = connectMsg;`
`  var net = require('net');`
`  var s = net.createServer(this.fn);`
`  console.log('started');`
`  s.listen(port);`
`}`
`App.prototype.fn = function(sock) {`
`  console.log(this.msg);`
`};`
`var app = new App(3000, 'connected');`
	i) Name 2 ways to connect to this running server (that is, name 2 *clients* you can use to communicate with this running server).
	Chrome (or any browser)
	and net cat
		j) When a client connects to this server, the server should log out `'connected'`. However, it logs out `undefined` instead. **Draw an arrow** to the line that needs to be rewritten, and **rewrite** it below:
		var s = net.createServer
		(app.fn);

4. What are the two HTTP request methods that we've used to send data to the server? When a form is submitted with either method, specify where in the *actual* HTTP request the data from the form elements is located. On the server side, what object / property in the Express framework contains that data? Lastly, when would you use one request method over the other? (4 points)

Method Name	Where in HTTP Request is It Located?	Name of Obj /Prop in Express API
GET	the query string of the URI	req.query
POST	the body of the request	req.body

When would we use one request method over the other?	We would use post if we wanted the data that is being sent to be a little more secure since it is parsed from the body opposed to being visible in the URI.

5. Answer the following database related questions. (4 points)

a) In the context of MongoDB, describe the following: **document**, **collection** and **database**

A document is a single object that is a bunch of key value pairs. A collection is a group of documents. MongoDB is a non relational database organized by document stores. It is a semi-structured database that allows for fluid data.

b) Aside from a database, name two other potential mechanisms for persistent data storage (solutions that persist data only while the application are running are also acceptable):

6. You have an **Express application** that stores cat names. The path, `/cats`, should display a list of all of the cat names, along with their number of lives, that are stored in the database. The page that is displayed should look like this:

Cats

- Paw Newman - 7
- Kitty Purry - 9
- Chairman Meow - 3
- Bill Furry - 5

Finish the **route handler** and **template** below for `/cats`. (4 points)

a) the mongoose schema is defined in the 1st column (`db.js`)
b) **add the line to send back a response based on a template in the 2nd column** (`app.js`)
c) **finish the template so that it displays an unordered list of all of the cat names and their number of lives that are in the database** (`views/cats.hbs`)
d) you can assume that a all of the other required files and code (`app.listen(...)`, `layout.hbs`, etc.) are present

```
// db.js
// ----------

var Cat = new
mongoose.Schema({
 name: String,
 lives: Number
});

mongoose.model('Cat', Cat);
```

```
// app.js
// ----------

var Cat = mongoose.model('Cat');

app.get('/cats', function(req, res){

  var cb = function(err, cats, count){
      res.render('cats',
  }


  Cat.find({}, cb);


});
```

```
// views/cats.hbs
// ----------
<h1>Cats</h1>

<ul>
  {{#each cats}}
  <li> {{this.name}} -
       {{this.lives}}
  </li>
  {{/each}}
</ul>
```

Choose 3 out of the next 4 questions to answer. Mark the question you want to skip by writing DO NOT GRADE at the top of the question.

7. You have a directory called **`static`** in your Express project, and it contains html and css files. You'd like to implement a feature that lets you serve the files in that `static` folder without specifying explicit paths for each file. (18 points)

   Here's what your app will do:

   a) send the contents of files in the `static` folder back as http response bodies based on the url that was requested
   b) for example requesting http://your.site/baz.html would read the contents of `static/baz.html` and send it back as part of the response without having to define a route handler for `/baz.html`
   c) if the path specified in the url does not match a file in your static folder, then let the rest of your application handle the request (for example, if the original request was for `'/foo'`, but `foo` doesn't exist in the `static` folder, your app may still be able to handle that path through a regular route handler like `app.get('/foo', ...)`
   d) to implement this:
      - for every incoming request, regardless of path...
      - try to find the resource that the request is asking for in your app's static folder
      - you can assume that if the path doesn't have an extension of html or css, that the file is not in `static`
      - if the file exists, serve up the file's content (which can be html or css)
      - make sure the browser knows how to handle the contents of the request body!
      - if there's **any issue opening the file** (for example, if it doesn't exist)
      - just **allow the request to continue being handled** by the *rest* of your application
      - you **cannot use `express-static`** (this will essentially replicate the functionality of express static) or **`res.sendFile`**
      - **hints**:
        - you'll probably need to figure out the extension of the file... when you extract the extension, make sure that the dot (.) is not included, and also make sure to handle the case where there's no extension!
        - you'll need to use the `fs` module to open a file; this should be familiar from homework...
        - make sure you check the error object; it will let you know if there was an error reading the file:
        - fs module usage summary / refresher:

```
var fs = require('fs');
fs.readFile(fileName, callbackFunction);
// callbackFunction has two parameters:
// err – undefined/null if read worked, an object if there was an error
// for example, if they file were not found err would be an object
// data - the contents of the file on a successful read
```

Write code that does this below. **Assume that all the Express setup requirements are already taken care of. Just add the code to implement this feature below.**

DO NOT GRADE

8. Implement two higher order functions, **`mapWith`** and **`any`**. **You must use two of the following built-in Array methods to implement these functions**: `forEach`, `filter`, `reduce`, or `map`. You can only use each one once. Remember that `break` does not work with any of these methods. (18 points)

a) **`mapWith(fn)`**
- `mapWith` **has a single parameter**, `fn`, which is a function that takes in a single argument and returns a value
- `mapWith` **returns a function** that:
  - has 1 parameter, an `Array`
  - returns a new `Array` where every element is the result of calling `fn` on elements from the incoming `Array`

This is different from *regular* map. The regular version of `map` immediately calls the callback function on every element in an Array to return a new `Array`. `mapWith`, on the other hand, gives back a function rather than executing the callback immediately (think of the difference between `bind` and `call`/`apply`). `mapWith` is basically a function that turns another function into a *mapping* function (a function that works on `Arrays`). Here's how it works:

```
function square(n) {return n * n;}      // original square function that works on Numbers
mapWithSquare = mapWith(square);        // create a 'mapped' version of the square function
console.log(mapWithSquare([1, 2, 3]));  // now square can work on Arrays of Numbers!

mapWithParseInt = mapWith(parseInt);
console.log(mapWithParseInt([' 123', '45', '67   ']));
```

b) **`any(arr, func)`** – returns true if *any* (at least one) of the elements in the Array, `arr`, pass the test, `func`. The test, `func`, is a function that takes a single argument, and it returns either true or false. Here's an example:

```
// prints out false (no elements pass the test function)
console.log(any([1, 2, 3, 4], function(num) {
    return num > 100;
}));

// prints out true (at least one of the elements passes the test function)
console.log(any([{name: 'joe'}, {color: 'green'}, {x: 42}], function(obj) {
    return obj.hasOwnProperty('x');
}));
```

a)
```
function mapWith(fn) {

    return function (array) {
        var newArray = [];
        array.forEach( function (element) {
            newArray.push( fn(element) );

        return newArray;
    });
```

b)
```
function any (arr, func) {

    arr.filter( function (element) {
        func(element)
    if (arr.length >= 1)
        return true;
});
```

9. Write a constructor that makes **`Request` objects**. Unlike our homework, this object **will not parse** an HTTP Request string. Instead, its **properties will be set** through **individual arguments** passed to the constructor. Additionally, it will have the **ability to send an actual http request** and **print** out the resulting response! (18 points)

   You will do this by using the net module as a client (see the example client code in the first column below) and by examining the example usage of a `Request` object in the second column.

`Client Code with net Module Example`

```
var net = require('net');
var client = new net.Socket();

// connect to port and domain
client.connect(3000, 'localhost', function() {

  // this code is only executed when this
  // client successfully connects to a server
  console.log('Connected');

  // write sends data to the server connected
  // to
  client.write('Example being sent');
});

client.on('data', function(data) {
  // this code is executed when the server
  // sends
  // data back to the client
  console.log('Received: ' + data);
  client.end();
});
```

`Example Usage of Request Constructor and Object`

```
// create a new Request object with constructor
var req = new Request('GET', 'www.google.com', 80);

// resulting object has these properties
console.log(req.method);   // GET
console.log(req.domain);   // www.google.com
console.log(req.port);     // 80
console.log(req.path);     // /

// the properties are defined on the object itself
console.log(req.hasOwnProperty('method')); // true

// the method, send, is defined elsewhere???
console.log(req.hasOwnProperty('send'));   // false

// send the request
req.send()

// prints out response from server
// HTTP/1.1 200 OK … etc.
```

a) Note that the resulting `Request` object has the following **properties**: **`method`**, **`domain`**, **`port`** and **`path`**
b) These properties are set from the **constructor's parameters**: **`method`**, **`url`** and **`port`**
c) `domain` and `path` are taken from `url` – you can assume that no protocol is included (no http://), everything before the 1st / is the `domain`, and everything after is the `path`. For example: foo.bar/baz/qux → `domain` is `foo.bar` and `path` is `/baz/qux`
d) If the domain doesn't contain a /, assume the whole url is the `domain`, and the `path` is /
e) The `Request` object's **`send`** method **connects to the server** specified by the `domain` and `port`, **sends a valid http** request (no need to add a Host header), and **prints out the response** when it receives data
f) You can copy most parts of the client code wholesale for your implementation, but you will have to make a few modifications
g) In particular, **watch out for the value of `this`** when using callbacks (either bind this to another name in the closure, make the variables you need available in the closure or use bind).

```
function Request(method, url, port)
  var method, url, port;
  if (url.indexOf('/') === -1)
     this.domain = url
     this.path = '/';
  else{
     var strings = url.split('/')
     if (strings.length > 2)
        this.domain = strings[0]
        var paths = strings.slice(1, strings.length + 1)
        this.path = paths;
     else
        this.domain = strings[0]
        this.path = string[1];
  }
}
this.method = method;
this.port = port;
```

[{ "adverb": ____ "verb": ____ },]

10. **Write the route handlers and the templates for a small Express application.** The application is a collaborative poem where users enter an adverb and a verb combination. Every combination entered by every user is displayed. (18 points)

A Poem **Add to the poem:** adverb: casually verb: cook Submit	**A Poem** casually cook **Add to the poem:** adverb: verb: Submit	**A Poem** casually cook elatedly eat soundly sleep **Add to the poem:** adverb: verb: Submit	must not leave fields blank adverb: verb:
The path /poem displays all of the lines of the poem (starting with no lines), along with a form to add a new line.	Submitting the form adds a line to the poem and takes you back to the original page (showing your new line added).	Lines can continually submitted to to construct a poem.	Leaving a field blank when submitting a form results in the form being re-rendered with an error message.

a) Assume that all required modules and configuration is already present (body-parser, hbs, etc.); you only have to write the routes, templates and any globals you'll need for persisting data
b) You can also assume that a views directory is setup with a layout.hbs file
c) When you write your templates, you can omit closing tags as a shorthand way of writing html
d) There's only one page in the entire application: /poem
e) You should not be able to refresh the page and cause a form submission to resubmit (with the exception of errors – see part g)
f) The poem looks the same for all users (that is, if I add a line on my browser, it will show up when you view the page on your browser). It's ok if all of the lines of the poem disappear when the server restarts
g) If either of the fields are left blank, display an error message above the form (in this case, it's ok if a refresh causes a resubmit)

```
// var poemList = []

app.get('/poem', function(req, res) {
    res.render('poem', {"poemList": poemList});

app.post('/poem', function(req, res) {
    var adverb = req.body.adverb
    var verb = req.body.adverb
    var poem = {"adverb": adverb, "verb": verb}
    poemList.push(poem);

    if (adverb === undefined || verb === undefined)
        var error = "must not leave field blank";
        res.render('poem', {"poemList": poemList, "error": error})
    else {
        var poem = {"adverb": adverb, "verb": verb}
        poemList.push(poem)
        var error = ""
        res.render('poem', {"poemList": poemList, "error": error})
```

#10

<!-- views/poem.html -->
<h1> A Poem </h1>

{{#each poemList }}
<h4> {{this.adverb}} {{this.verb}} </h4>
{{/each}}

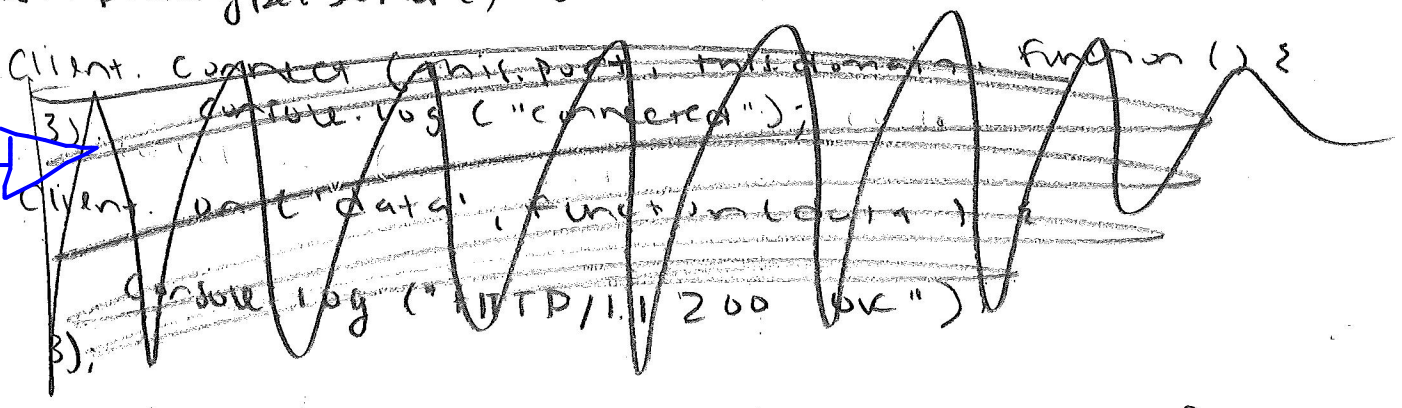
<h2> Add to the poem: </h2>
{{error}}

<form type="POST" method='/poem'>
adverb: <input type="text" name="adverb">
verb: <input type="text" name="verb">
<input type="submit">
</form>

#9

= function

Request.prototype.send() {



console.log(this.method, this.path, "HTTP/1.1")

};

client.connect(this.port, this.domain, function() {
req.send();
})

client.on('data', function(data) {
console.log("HTTP/1.1 200 OK")
}

Name Jessica Donahue Net ID JMD732

Objects / Methods Reference

Array

properties

length

methods

pop()

reverse()

sort([compareFunction])

splice(index, howMany[, element1[, …[, elementN]]])

slice(index, howMany[, element1[, …[, elementN]]])

join([separator = ','])

concat(value1[, value2[, ...[, valueN]]])

indexOf(searchElement[, fromIndex = 0])

forEach(callback[, thisArg])

map(callback[, thisArg])

filter(callback[, thisArg])

reduce(callback[, initialValue])

some(callback[, thisArg])

every(callback[, thisArg])

String

properties

length

methods

split([separator][, limit])

toUpperCase()

slice(beginSlice[, endSlice])

replace(regexp|substr, newSubStr|function[, flags])

substring(begin[, end])

indexOf(ch)

lastIndexOf(ch)

Object / Object.prototype

Object.getPrototypeOf(obj)

Object.prototype.hasOwnProperty(prop)

Request Object

properties

body

headers

method

path

query

session

url

methods

get

Response Object

methods

append

end

redirect

render

send

sendFile

set

status

writeHead