Feedback — HW 2 Quiz: SSJS Interactive Tutorial

You submitted this quiz on **Sun 30 Jun 2013 3:06 PM SGT (UTC +0800)**. You got a score of **10.00** out of **10.00**.

HW2 Quiz: SSJS Interactive Tutorial

Background

We're going to learn a bit more about server-side Javascript (SSJS) via `node` by going through a slightly more complex script than we did last time and answering some questions in the process. Many of these answers will be self-evident as you do the problems; others may require a little bit of Googling, common sense, or (in a pinch) skimming of the Node Beginner Book. As usual, we assume these commands are being executed on AWS on a EC2 t1.micro instance running Ubuntu 12.04.2 LTS, unless otherwise specified. Note that part of the goal here is to teach you to read code; there are many little subtleties that one can only really glean by reading source code and being resourceful.

Setup

We're going to be working with `market-research.js` , a simple script which one can make progressively more sophisticated to do some basic market research on public companies. Over time we'll modify this code to help analyze sectors and industries that are good places to start companies. But right now we're going to use it to look at the profits (aka earnings) of some public technology companies by programmatically accessing the Yahoo Finance API.

First, execute the following commands to download and run the script.

```
  # Download and make executable
  wget https://spark-public.s3.amazonaws.com/startup/code/market-researc
h.js
  wget https://spark-public.s3.amazonaws.com/startup/code/market-researc
h-wrapper.js

  # Install npm dependencies. This will create a node_modules directory
in
  # the current working directory. Don't cd into other directories right
  # now; later we'll show how to install modules globally.
  npm install restler csv accounting

  # As a script
  node market-research.js
  node market-research.js FB ORCL
```

```
  # As an executable
  chmod 777 market-research.js
  ./market-research.js
  ./market-research.js GOOG CRM

  # As a module, through another program invoked as a script
  node market-research-wrapper.js

  # As a module, through another program being invoked as an executable
  chmod 777 market-research-wrapper.js
  ./market-research-wrapper.js

  # Also as a module - but with the external code being input at the com
mand line
  # via the -e flag
  node -e "require('./market-research.js')"
  node -e "var mr = require('./market-research.js'); mr.marketResearch()
;"
  node -e "var mr = require('./market-research.js'); mr.marketResearch([
\"FB\",\"ORCL\"]);"
```

This illustrates several different ways to use the code: as a script explicitly with `node`, as a standalone command-line executable, and most interestingly as a module in another piece of code (`market-research-wrapper.js`). This is a very useful technique: develop and debug your code independently as a standalone script, with an eye towards externalizing one (or perhaps a few) key routines for invocation as a library by the larger codebase.

Here is the code itself:

market-research.js

```
#!/usr/bin/env node
/*
Use the Yahoo Finance CSV API to do some basic market research calculati
ons.

 - Background: http://greenido.wordpress.com/2009/12/22/yahoo-finance-hi
dden-api/
 - Example URL: http://finance.yahoo.com/d/quotes.csv?s=GOOG+FB+AAPL&f=s
nj1pr
    s:  Symbol
    n:  Name
    j1: Market Capitalization (in billions)
    p:  Price-per-share (at previous close)
    r:  Price to Earnings Ratio

Further references.

 - https://github.com/danwrong/restler
```

```
 - https://github.com/wdavidw/node-csv
 - http://josscrowcroft.github.io/accounting.js
 - http://stackoverflow.com/questions/4981891/node-js-equivalent-of-pyth
ons-if-name-main
 - http://nodejs.org/docs/latest/api/util.html#util_util_format_format

*/

var util = require('util');
var fs = require('fs');
var rest = require('restler');
var csv = require('csv');
var accounting = require('accounting');
var CSVFILE_DEFAULT = "market-research.csv";
var SYMBOLS_DEFAULT = ["GOOG", "FB", "AAPL", "YHOO", "MSFT", "LNKD", "CR
M"];
var COLUMNS_DEFAULT = 'snj1pr'; // http://greenido.wordpress.com/2009/12
/22/yahoo-finance-hidden-api
var HEADERS_DEFAULT = ["Symbol", "Name", "Market Cap", "Previous Close P
rice",
                       "P/E Ratio", "Shares", "EPS", "Earnings"];

var financeurl = function(symbols, columns) {
    return util.format(
        'http://finance.yahoo.com/d/quotes.csv?s=%s&f=%s',
        symbols.join('+'),
        columns);
};

var marketCapFloat = function(marketCapString) {
    return parseFloat(marketCapString.split('B')[0]) * 1e9;
};

var csv2console = function(csvfile, headers) {
    console.log(headers.join("\t"));
    csv()
    .from.path(csvfile)
    .on('record', function(row, index) {
        var shares = Math.round(marketCapFloat(row[2])/row[3], 0);
        var eps = (row[3]/row[4]).toFixed(3);
        var earnings = accounting.formatMoney(eps * shares);
        outrow = row.concat([shares, eps, earnings]);
        console.log(outrow.join("\t"));
    });
};

var buildfn = function(csvfile, headers) {
    var response2console = function(result, response) {
        if (result instanceof Error) {
            console.error('Error: ' + util.format(response.message));
```

```
        } else {
            console.error("Wrote %s", csvfile);
            fs.writeFileSync(csvfile, result);
            csv2console(csvfile, headers);
        }
    };
    return response2console;
};

var marketResearch = function(symbols, columns, csvfile, headers) {
    symbols = symbols || SYMBOLS_DEFAULT;
    columns = columns || COLUMNS_DEFAULT;
    csvfile = csvfile || CSVFILE_DEFAULT;
    headers = headers || HEADERS_DEFAULT;
    var apiurl = financeurl(symbols, columns);
    var response2console = buildfn(csvfile, headers);
    rest.get(apiurl).on('complete', response2console);
};

if(require.main == module) {
    console.error('Invoked at command line.');
    var symbols = process.argv;
    if(symbols.length > 2) {
        symbols = symbols.slice(2, symbols.length);
    } else {
        symbols = undefined;
    }
    marketResearch(symbols);
} else {
    console.error('Invoked via library call');
}

exports.marketResearch = marketResearch;
```

And here is a wrapper that shows how to invoke this as a module. We're able to do so because of the last line in `market-research.js` (namely `exports.marketResearch = marketResearch;`):

market-research-wrapper.js

```
#!/usr/bin/env node
// Example of using market-research.js as a module
var mr = require('./market-research.js');
mr.marketResearch(["FB", "ORCL"]);
```

The best way to understand what the code does is to download it, edit it, change the arguments around, and so on. The quiz questions will walk you through the properties of individual functions.

Hints and Tips

Understand what the code is doing

We're hitting the Yahoo Finance API, pulling down data on stock symbols, and computing a few derived quantities (the number of shares, the earnings per share, and the total earnings or profit). This blog post gives more details. Note that some large public companies have no profits or minimal profits; this will result in some "NaN" values when calculating things like earnings-per-share.

First-Class Functions in JS

Note that the `market-research.js` script is *not* fully idiomatic node code and has been simplified in some key respects to minimize the use of asynchronous programming. That said, we weren't able to fully avoid the use of first-class functions, so to understand what's going on with `buildfn` and `response2console`, you'll want to read about Functions as first-class objects in Javascript. Again, if you have a CS106 or equivalent background, you should have already been exposed to this (perhaps under the name of "function pointers"), but it might be worthwhile to open up a separate buffer (using `nano` or `emacs`) and make a few test scripts to check your intuition.

Read the Library Documentation

One of the most important skills when doing a startup is learning how to learn. My friend Jawed Karim has a saying: "There is no expert. You are the expert." This means that the moment you leave the confines of academia and try to do something no one has ever done before (e.g. start a technology company), the answers are no longer in the back of the book and you'll have to become ever more resourceful and self-directed. Towards that end, it is a useful exercise to read the documentation for the restler, csv, and accounting modules, as well as a bit about process.argv and then refer back to market-research.js to compare to the corresponding module invocations therein. It's a useful exercise to mess around with these invocations to add more parameters, change the behavior, and otherwise modify the script.

Question 1

Which of the following statements is true about our use of `npm` and `chmod` before executing `./market-research.js`?

Your Answer	Score	Explanation
○ Using `chmod` to change executability is essential to invoking the script at the command line, but using `npm` to install the libraries is not.		
○ Installing the libraries via `npm` is essential to invoking the script at the command line, but changing executability via `chmod` (or a		

similar command) is not.		
We need to both install necessary libraries and make the file executable before executing it.	✔	1.00
Neither is essential, but doing this a useful practice.		
Total		1.00 / 1.00

Question 2

After looking at the source of `market-research.js`, what variable holds the arguments sent to the script when invoked at the command line?

Your Answer		Score	Explanation
`console.log`			
`columns`			
`process.argv`	✔	1.00	
`require.main`			
Total		1.00 / 1.00	

Question 3

What do the lines like `symbols = symbols || SYMBOLS_DEFAULT;` do?

Your Answer		Score	Explanation
This is a JS idiom for setting `symbols` to `undefined`.			
This is a JS idiom for overriding user inputted arguments.			
This is a JS idiom for setting default arguments.	✔	1.00	
This is a JS idiom for checking that arguments are within a valid range.			
Total		1.00 / 1.00	

Question 4

Which of the following are standard `node` modules, included with the default install?

Your Answer		Score	Explanation
☑ `util`	✔	0.20	
☐ `csv`	✔	0.20	
☐ `restler`	✔	0.20	
☑ `fs`	✔	0.20	
☐ `accounting`	✔	0.20	
Total		1.00 / 1.00	

Question 5

What is the most likely bug in the `marketCapFloat` function?

Your Answer		Score	Explanation
◉ The function currently assumes dollar values in the billions.	✔	1.00	
○ It incorrectly assumes that the return value of the function should be a floating point.			
○ No bug in this function.			
○ `parseFloat` should not be used here.			
Total		1.00 / 1.00	

Question 6

What does `csv2console` do?

Your Answer		Score	Explanation
Reads in data from `csvfile`, asserts that it is within range, and prints to STDOUT.			
Reads in data from the Yahoo Finance API directly.			
Reads in data from `csvfile` and prints it out unchanged to STDOUT.			
Reads in data from `csvfile`, computes and formats derived quantities, and prints these out to STDOUT.	✔	1.00	
Total		1.00 / 1.00	

Question 7

What does `buildfn` do? (Hint: read this and this if you've never heard of closures.)

Your Answer		Score	Explanation
Creates and returns a function named `response2console`, which is parametrized with the values of `result` and `response` from the enclosing scope.			
Prints out an error message if the HTTP response returns an error.			
Creates and returns a function named `response2console`, which is parametrized with the values of `csvfile` and `headers` from the enclosing scope.	✔	1.00	
Writes data from the Yahoo Finance API to `csvfile`.			
Total		1.00 / 1.00	

Question 8

What does the `marketResearch` function do?

Your Answer	Score	Explanation

Constructs a Yahoo Finance API call and exits		
Resolves errors when invoked at the command line		
Sets several default values and then exits.		
Constructs a Yahoo Finance API call, and invokes code that first writes the results of that API call to disk and then reads/prints/processes said file	✔	1.00
Total		1.00 / 1.00

Question 9

What does the `if(require.main == module)` conditional do? Check all that apply. (Hint: see here and here (halfway down the page) if you can't figure it out from context).

Your Answer		Score	Explanation
Confirm that the code is only being invoked as a library and shut down with an error otherwise.	✔	0.25	
Confirm that the code is only being invoked as a command line app and shut down with an error otherwise.	✔	0.25	
Triggers two different blocks of code, one that executes when `market-research` is invoked from the command line and one that executes when invoked as a module via `require`.	✔	0.25	
Parse the stock symbols input at the command line, starting with the `process.argv` variable and producing the `symbols` variable.	✔	0.25	
Total		1.00 / 1.00	

Question 10

Why do we use `console.error` in addition to `console.log` (Hint: see here)?

Your Answer	Score	Explanation
`console.error` is the same as `console.log`, we just do it		

for the sake of variety.

`console.error` will force the program to exit.		
`console.error` prints to STDERR, while `console.log` prints to STDOUT. This allows us to print both metadata/error messages and actual output at the same time.	✔	1.00
`console.error` is more robust.		
Total		1.00 / 1.00