Node.js - Tips, Gotchas and Tools

Added by Ragan, Richard, last edited by Ragan, Richard on May 07, 2014

Node.js Gotchas

When you first start with the asynchronous coding style of node, the following are likely places you might trip up. Understand these and bear them in mind, and you will have fewer problems.

- Be suspicious of any normal JavaScript control constructs (if/else, for, foreach) containing asynch code. Flow will likely escape your control construct before the contained asynch logic is executed. For example,

```
  if (test) {
  asynch code
  then do other stuff
}
```

  or even just

```
  asynch code
```

then do other stuff // this will likely execute before the asynch code completes

In general, assume the asynch code will execute much later after you have done “the other stuff”. A for loop tends to run through all the iterations firing off all the aysnch code which only executes much later. If you expect things to have changed after leaving the for loop, it is likely they won’t have.

Remedy: Use the async module: https://github.com/caolan/async#each. It has each and eachSeries for looping in an asynch world, waterfall for executing a series of actions serially passing data between the actions, reduce for doing a reduction operation on an array, and much more.

- Remember to do res.render only when you are done with all processing in setting up the model. Minor point but be careful.
- Instead of doing “callback(null, value);” do “return callback(null, value);”. The return ensures you don’t fall through into other things.
- Be careful of the name **callback**. It becomes a habit to always use this name for a callback but if you have nested code you can end up calling the wrong one:
```
var createTicket = function createTicket (model, callback) {
  …
  var callback = function(response) {…
    http.get(options, callback).end(); }
}
```
If you are doing something like async.waterfall, consider using the variable name "next" in lieu of callback for the stages of the waterfall.
- Remember on any callback that the first value is the error parameter and pass null if things are fine. Pass the Error object if you have an error to report and always check for errors in callbacks.
- Don't invoke your dust partial in a template with a leading slash (e.g. {>"/wallet/addFI" /} will cause you problems in LIVE/Stage. The build process registers template paths starting immediately under the public/templates directory so no leading slash is needed and if you use one, the registered template name won't match the one you wrote when invoking the partial.

Tips

- Use console.dir(obj) to generate readable dumps of objects.
- Try out npm-check-updates module to find out what major/minor versions have changed that you should decide if you want to adopt via manual package.json changes.
https://npmjs.org/package/npm-check-updates

Node Modules of General Interest

- https://github.com/chriso/node-validator - Good for validating form field input values..

Node Remote Debugging on a stage

For something better than adding/removing console.log lines in your code:

- Install node-inspector on your stage (hopefully one day this will just be on all stages automatically) - https://github.com/node-inspector/node-inspector
- Start node-inspector in the background (e.g. node-inspector&) or start it in a separate console window in foreground
- Start your node application with node --debug index.js (or whatever your main starting point is).
- Go to Chrome and enter the url: http://yourStage:8080/debug?port=5858
- You should get a page with a Scripts icon. Click this to view the source of a particular bit of JS where you want to set a breakpoint and get debugging
- If, for some reason, you need to set breakpoints before any execution of your app starts, use --debug-brk instead of --debug on the node command

Node Debugging Tools (beyond node inspector/Webstorm)

- Tracebacks too cryptic from error: longjohn - Longer traces: https://github.com/mattinsler/longjohn
- Longer stack traces: long-stack-traces https://github.com/tlrobinson/long-stack-traces
- Having trouble with various modules of node (like module loading, or http, etc.). Use NODE_DEBUG to get more insight. http://juliengilli.com/2013/05/26/Using-Node.js-NODE_DEBUG-for-fun-and-profit/
- See time spent in each middleware module: https://github.com/guille/express-trace

Node Code Coverage Tool

- Node.js framework team suggests Istanbul for code coverage

Node Unit Test Tool

- Node.js framework team suggests Mocha and JSHint for unit testing

Node Functional Test Tool

- Node.js framework team suggests Nemo for functional testing

Node Task Runner

- Node.js framework team suggests Grunt for running tasks

Node Testing Tools

- Unit testing services with mock data: https://github.paypal.com/gist/jsisk/4f8149bad44cbcd4de4d
- Testing to be sure app is robust if service calls are slow or non-responsive: https://github.com/ql-io/netmorphic

Node Performance and Leak Finding Tools

This is a list of links to tools that may help you with your L&P issues:

- Memory Leaks:
    - Tracking memory leaks in node.js: https://hacks.mozilla.org/2012/11/tracking-down-memory-leaks-in-node-js-a-node-js-holiday-season/
    - Memory Leak Patterns in JavaScript: null those big variables http://www.ibm.com/developerworks/web/library/wa-memleak/
    - Writing Fast, Memory Efficient Javascript: http://coding.smashingmagazine.com/2012/11/05/writing-fast-memory-efficient-javascript/
- DTrace for latency investigations (requires Joyent SmartOS)
    - http://techapostle.blogspot.com/2012/11/smartos-improves-nodejs-debugging.html
    - http://techapostle.blogspot.com/2012/11/smartos-improves-nodejs-debugging_27.html
- Event loop lag - See if you are hogging the event loop?
    - https://hacks.mozilla.org/2013/01/building-a-node-js-server-that-wont-melt-a-node-js-holiday-season-part-5/
- Profiling tool: Look: https://github.com/baryshev/look - Based on nodetime but doesn't send your data to a 3rd party
- Deeper profiling. Start node with --prof option, hit it with load and then post process the v8.log file using https://github.com/sidorares/node-tick
- OS X Instruments.app provides many useful profiling and memory tools.

Page Created by:

Ragan, Richard

Created Date:

Feb 28, 2013 21:41

Last Modified Date:

May 07, 2014 16:27