Why Node JS?

A brief Summary

Me

Kent Archie

kentarchie@gmail.com

Code and slides on GitHub

https://github.com/kentarchie/nodeJS-UniforumTalk

References at the end

Node JS history

Started in 2009 as a command line interpreter for Google V8 JavaScript

Microsoft adds support in 2011

Npm package manager added in 2011

NodeJS Foundation begun in 2015

Many IDEs developed for Node and Node support added to Visual Studio and others

Used by PayPal, Netflix, Uber, New York Times and many others

Node Basics

Node executes Javascript

It can be a web server by running code that implements a web server

It can be used for local applications as well

Written in C++, Node can be linked to existing libraries

- You can now run COBOL and Fortran code in Node

Applications don’t block, even the IO

Supports a Read-Eval-Print loop

Windows Installation

Get installers from nodejs.org

Run them

Notes here

https://docs.npmjs.com/getting-started/installing-node

http://blog.teamtreehouse.com/install-node-js-npm-windows

OS X Installation

Get installers from nodejs.org

Run them

I haven’t tried any of this

Notes here

http://blog.teamtreehouse.com/install-node-js-npm-mac

http://sourabhbajaj.com/mac-setup/Node.js/README.html

http://shapeshed.com/setting-up-nodejs-and-npm-on-mac-osx/

Linux Installation

Simplest process is to use the software install on your distribution. For example:

```
sudo apt-get install nodejs
```

But these can be out of date

So notes here on updating the installation

http://www.hostingadvice.com/how-to/install-nodejs-ubuntu-14-04/

Javascript without a browser

Node runs any Javascript, just like your browser

No DOM, no graphics

Access to local file system

Packages to access databases and other external libraries

REPL Example

```
$ node
> var a=6;

> function bob(foo) {
... console.log("bob: result="+foo+3)
... }

> bob(3);
bob: result=33

>
```

File Search

You can write applications in Node that work solely on the local machine

This example opens a file, reads it line by line and searches each line for a specified string. It then prints lines that match

It uses 3 libraries to access files, parse file paths and process command line parameters.

There are lots of better ways to do this, with and without Node, it’s just an example

File Search code, part 1

```
var fs = require('fs');  // we need the file system library
var path = require('path');  // library to process file paths
var commandLineArgs = require('command-line-args'); // command line processing

// define allowed command line args
var cli = commandLineArgs([
  { name: 'fileName', alias: 'f', type: String }
  ,{ name: 'searchString', alias:'s',type: String}
])

// nothing to do if we don't have a file to search and something to search for
// node searchFile filename searchString
var commandName = path.basename(process.argv[1]);
if(process.argv.length < 4) {
    console.log('missing args: node '+commandName+' datafile searchString');
    process.exit();
}
var options = cli.parse(); // parse command line
```

File Search code, part 2

```
var fileName = options.fileName;
var searchString = options.searchString;
console.log(commandName + ': fileName=' + fileName +
'searchString='+searchString);

fs.readFile(fileName, 'utf8', function (err,data) {
  if (err) {
    return console.log(argv[1] + ':' + err);
  }

  var dataLines = data.split('\n');
  for(var i = 0; i< dataLines.length; ++i) {
    dataLines[i] = dataLines[i].trim(); // remove leading/trailing whitespace
    if(dataLines[i].indexOf(searchString) != -1)
        console.log(dataLines[i]);
  }
  console.log("fs.readFile anonymous function completed");
});
console.log("fs.readFile completed");
```

C Version

This is the C version of the main loop

```
if ( (file = fopen(fileName, "r")) == NULL ) {
    fprintf(stderr, "-f %s failed\n", fileName);
    return 1;
  }

  while ( fgets ( line, sizeof line, file ) != NULL ) {
    if(strstr(line, searchString) != NULL)
        printf("%s\n", line);
  } // while
```

C and Node Comparison

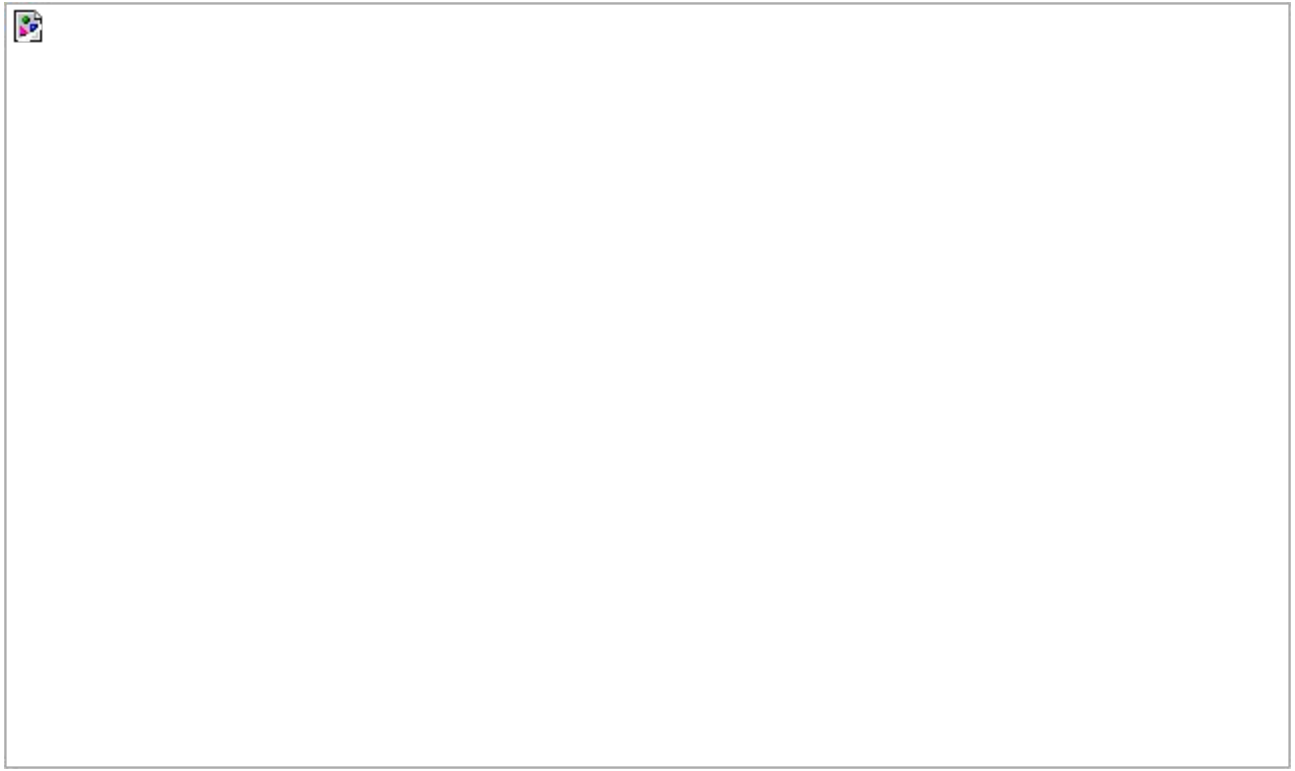
The C version reads and process the lines one by one, reading a line from the file and checking it.

The Node version gets the entire file contents, splits it into lines and checks them one by one.

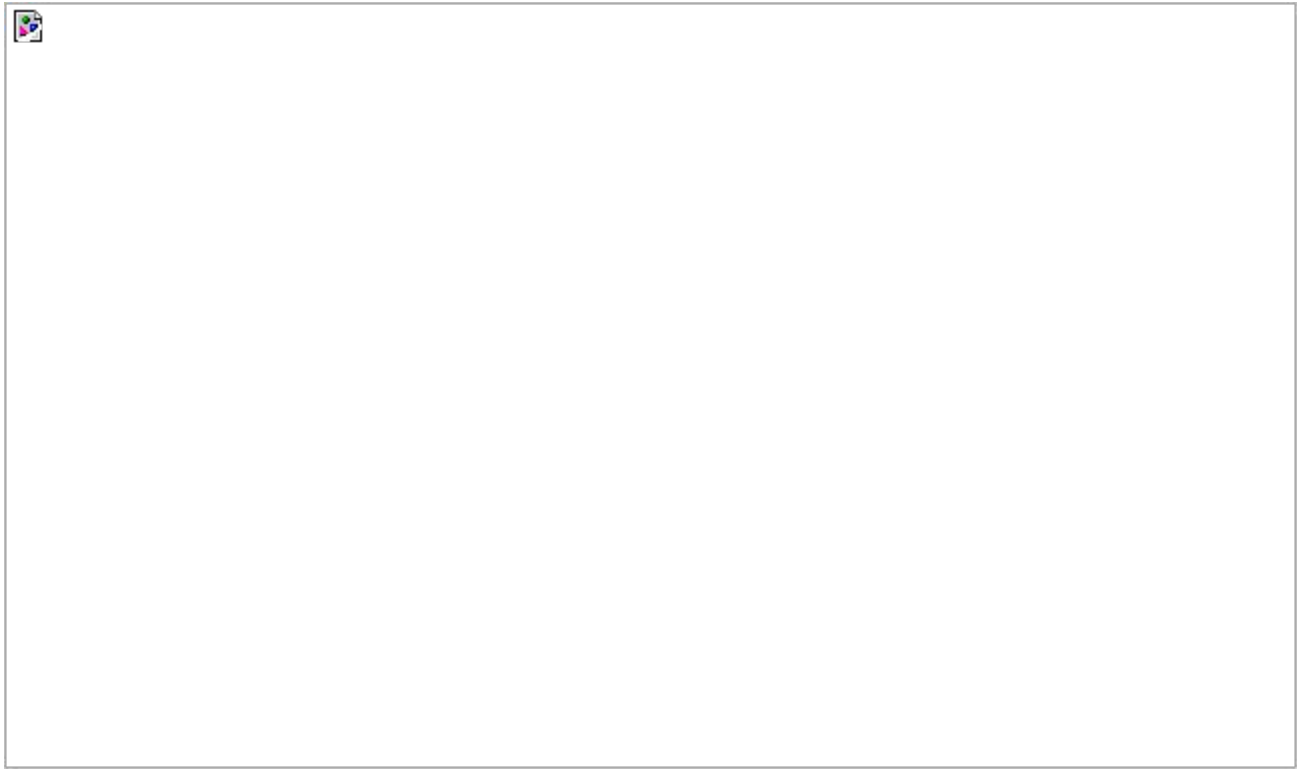
In the C version, the main code pauses while the data is read. In the Node version, the `fs.readFile()` call completes without waiting for the datafile to be read. When the data is available, the function passed to `fs.readFile` is called.

```
searchFile: fileName=testdata.csv searchString=stone
fs.readFile completed
Yellowstone1999,,yes,yes,,,yes
Yellowstone2005,yes,yes,yes,,,yes
fs.readFile anonymous function completed
```

General Web App Structure



NodeJS Web App Structure



WebServer and Node Comparison

The WebServer and the backend code are in different processes (mod_perl, etc are exceptions). Each call to the backend spawns another process.

They are generally not written in the same language.

In Node, there is a single process doing all the work. Each backend call is handled without making a new process.

The webserver code and the backend code are in the same language and the same process.

Howdy Web App

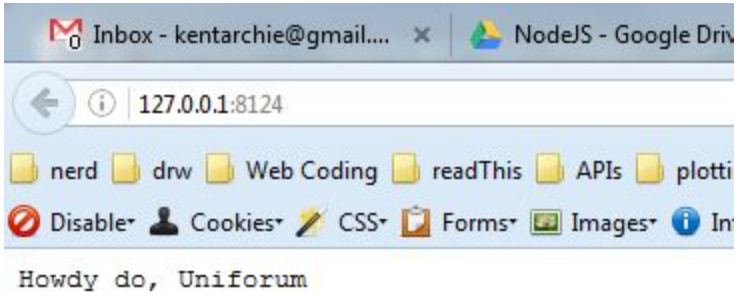
```
var http = require('http');   // need a web server library

function sendHowdy(req, res)
{
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Howdy do, Uniforum\n');
} // sendHowdy

http.createServer(sendHowdy).listen(8124, "127.0.0.1");

console.log('Server running at http://127.0.0.1:8124/');
```

Run it

$ node index.js

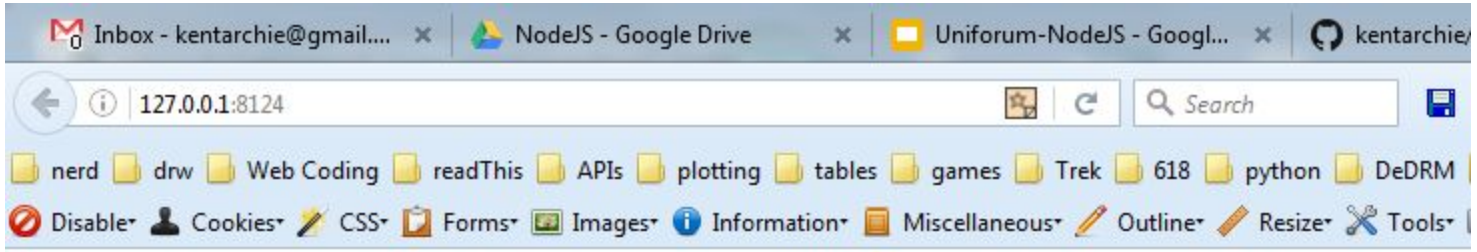
Server running at http://127.0.0.1:8124/

Results


Inbox - kentarchie@gmail....
NodeJS - Google Driv
127.0.0.1:8124
nerd
drw
Web Coding
readThis
APIs
plotti
Disable
Cookies
CSS
Forms
Images
Howdy do, Uniforum

Generate Page

```
var http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<html>');
    res.write('<body>');
    res.write('<h1>Page written by node</h1>');

    res.write('<h4>request headers</h4>');
    res.write('<pre>');
    res.write(JSON.stringify(req.headers,null,'\n'));
    res.write('</pre>');

    res.write('<hr />');
    res.write('</body>');
    res.write('</html>');
    res.end();
}).listen(8124, "127.0.0.1");
console.log('makePage server running at http://127.0.0.1:8124/');
```



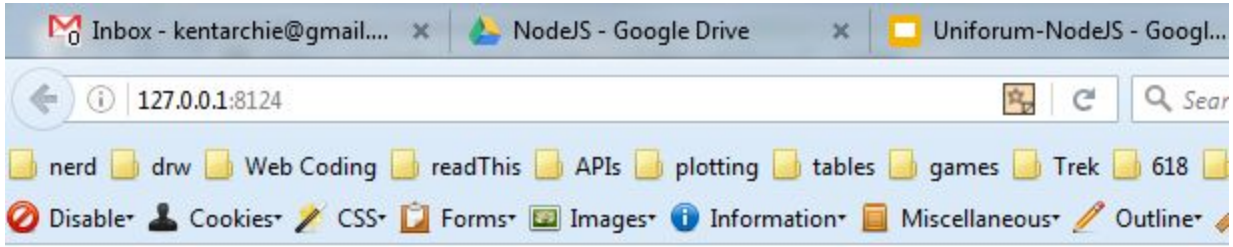
Page written by node

request headers


```
{
"host": "127.0.0.1:8124",
"user-agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:46.0) Gecko/20100101 Firefox/46.0",
"accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
"accept-language": "en-US,en;q=0.5",
"accept-encoding": "gzip, deflate",
"connection": "keep-alive",
"pragma": "no-cache",
"cache-control": "no-cache"
}
```

Page From File

```
var http = require('http');
var fs = require('fs');  // file system

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    fs.readFile('index.html', 'utf8',
        function (err,data) {
        if (err) {
            return console.log(err);
        }
        res.write(data);
        res.end();
        });
}).listen(8124, "127.0.0.1");

console.log('makePage server running at http://127.0.0.1:8124/');
```



Page copied from a file

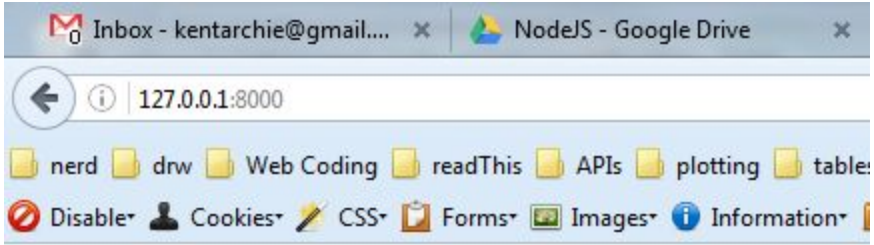
```
var http = require('http');
var fs = require('fs');  // file system

http.createServer(function (req, res) {
      res.writeHead(200, {'Content-Type': 'text/html'});
      fs.readFile('index.html', 'utf8',
         function (err,data) {
            if (err) {
               return console.log(err);
            }
            res.write(data);
            res.end();
         });
}).listen(8124, "127.0.0.1");

console.log('makePage server running at http://127.0.0.1:8124/');
```

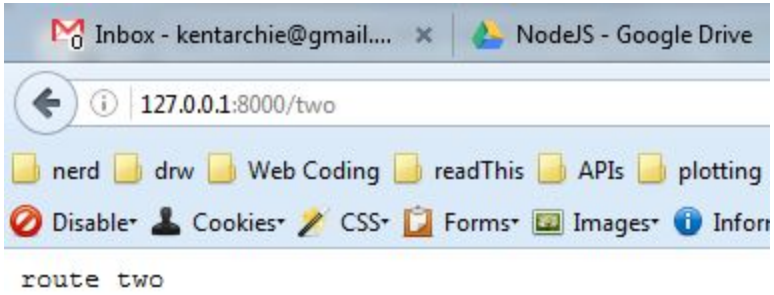
Simple Routing

```
var http = require('http'),   // library to act as web server
url = require('url'); // library to manipulate incoming url

http.createServer(function (req, res) {
  setTimeout(function () {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    var url_parts = url.parse(req.url);
    console.log(url_parts.pathname);
```

Simple Routing 2

```
    switch(url_parts.pathname){
    case '/':
        res.write("display root");
        break;
    case '/one':
        res.write("route one");
        break;
    case '/two':
        res.write("route two");
        break;
    default:
        res.write("oh dear, 404");
    }
    res.end();
  }, 2000);
}).listen(8000);
```


Inbox - kentarchie@gmail....
NodeJS - Google Drive
127.0.0.1:8000
nerd
drw
Web Coding
readThis
APIs
plotting
Disable
Cookies
CSS
Forms
Images
Information
display root


Inbox - kentarchie@gmail....
NodeJS - Google Drive
127.0.0.1:8000/two
nerd
drw
Web Coding
readThis
APIs
plotting
Disable
Cookies
CSS
Forms
Images
route two

Simple Routing log

$ node index.js

/

/favicon.ico

/favicon.ico

/two

Web Application

A typical application has 3 parts

A Node based webserver

Node controlled backend code

Front end web page

Include libraries

Npm manages a large set of Node libraries

Similar to CPAN in Perl, pip in python and gem in Ruby

A bit like `apt-get` in Ubuntu as well

Running it in a project updated the package.json file

`npm install --save express` will download and install the Express library but also update the package.json file

`npm --update` will update all dependencies in package.json

Web App Layout

package.json

```
{
  "name": "web-app-example",
  "version": "1.0.0",
  "description": "Uniforum NodeJS Web App Example",
  "main": "WebAppExample",
  "repository": "https://github.com/kentarchie/nodeJS-UniforumTalk.git",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Kent Archie",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.15.1",
    "express": "^4.13.4",
    "express-handlebars": "^3.0.0",
    "moment": "^2.13.0"
  }
}
```

Live Code

Show the structure and use of a small web app

Run using node WebAppExample.js

Talk about web page views and layouts

Demonstrate template code

Show difference between node code and web age code

References

Ok overview and history

https://en.wikipedia.org/wiki/Node.js

Web Development with Node and Express Todd Brown O'Reilly Press

Node JS Chicago Meetup

Npm notes

https://docs.npmjs.com/getting-started/using-a-package.json

Incomplete but interesting

https://blog.risingstack.com/node-hero-tutorial-getting-started-with-node-js/