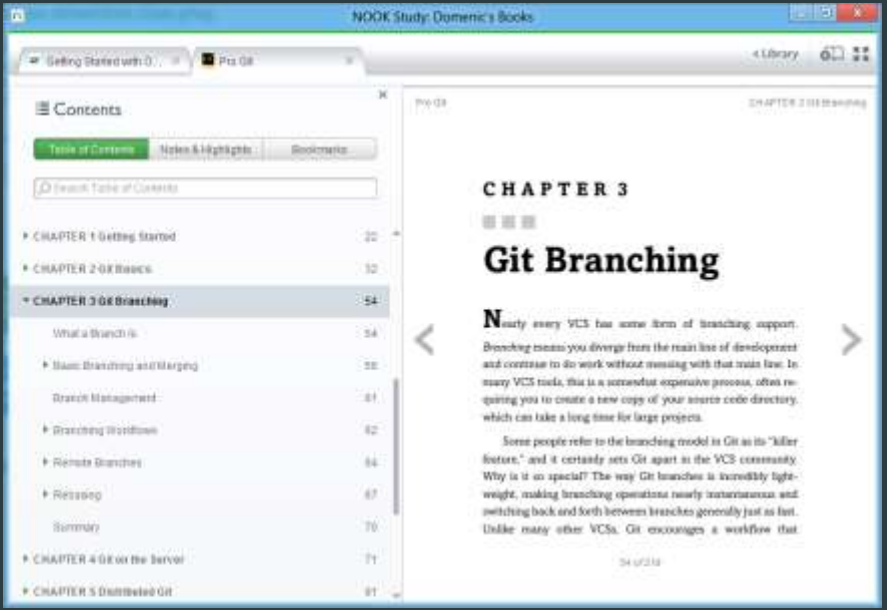
understanding the node.js platform

Domenic Denicola
http://domenicdenicola.com
@domenic

story time



NOOK Study: Domenic's Books

Getting Started with O... | Pro Git | ‹ Library

Contents

Table of Contents | Notes & Highlights | Bookmarks

Search Table of Contents

CHAPTER 1 Getting Started	22
CHAPTER 2 Git Basics	32
CHAPTER 3 Git Branching	54
What a Branch Is	54
Basic Branching and Merging	55
Branch Management	61
Branching Workflows	62
Remote Branches	64
Rebasing	67
Summary	70
CHAPTER 4 Git on the Server	71
CHAPTER 5 Distributed Git	81

Pro Git — CHAPTER 3 Git Branching

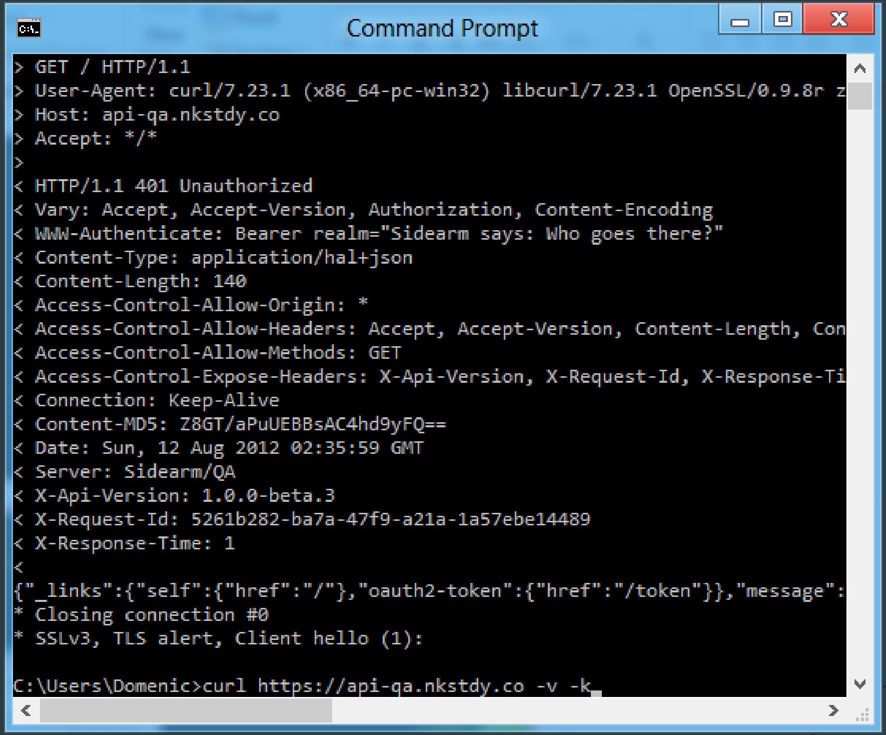
CHAPTER 3

Git Branching

Nearly every VCS has some form of branching support. Branching means you diverge from the main line of development and continue to do work without messing with that main line. In many VCS tools, this is a somewhat expensive process, often requiring you to create a new copy of your source code directory, which can take a long time for large projects.

Some people refer to the branching model in Git as its "killer feature," and it certainly sets Git apart in the VCS community. Why is it so special? The way Git branches is incredibly lightweight, making branching operations nearly instantaneous and switching back and forth between branches generally just as fast. Unlike many other VCSs, Git encourages a workflow that

54 of 218



Command Prompt

```
> GET / HTTP/1.1
> User-Agent: curl/7.23.1 (x86_64-pc-win32) libcurl/7.23.1 OpenSSL/0.9.8r z
> Host: api-qa.nkstdy.co
> Accept: */*
>
< HTTP/1.1 401 Unauthorized
< Vary: Accept, Accept-Version, Authorization, Content-Encoding
< WWW-Authenticate: Bearer realm="Sidearm says: Who goes there?"
< Content-Type: application/hal+json
< Content-Length: 140
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Accept, Accept-Version, Content-Length, Con
< Access-Control-Allow-Methods: GET
< Access-Control-Expose-Headers: X-Api-Version, X-Request-Id, X-Response-Ti
< Connection: Keep-Alive
< Content-MD5: Z8GT/aPuUEBBsAC4hd9yFQ==
< Date: Sun, 12 Aug 2012 02:35:59 GMT
< Server: Sidearm/QA
< X-Api-Version: 1.0.0-beta.3
< X-Request-Id: 5261b282-ba7a-47f9-a21a-1a57ebe14489
< X-Response-Time: 1
<
{"_links":{"self":{"href":"/"},"oauth2-token":{"href":"/token"}},"message":
* Closing connection #0
* SSLv3, TLS alert, Client hello (1):

C:\Users\Domenic>curl https://api-qa.nkstdy.co -v -k
```

@domenic

Domenic Denicola

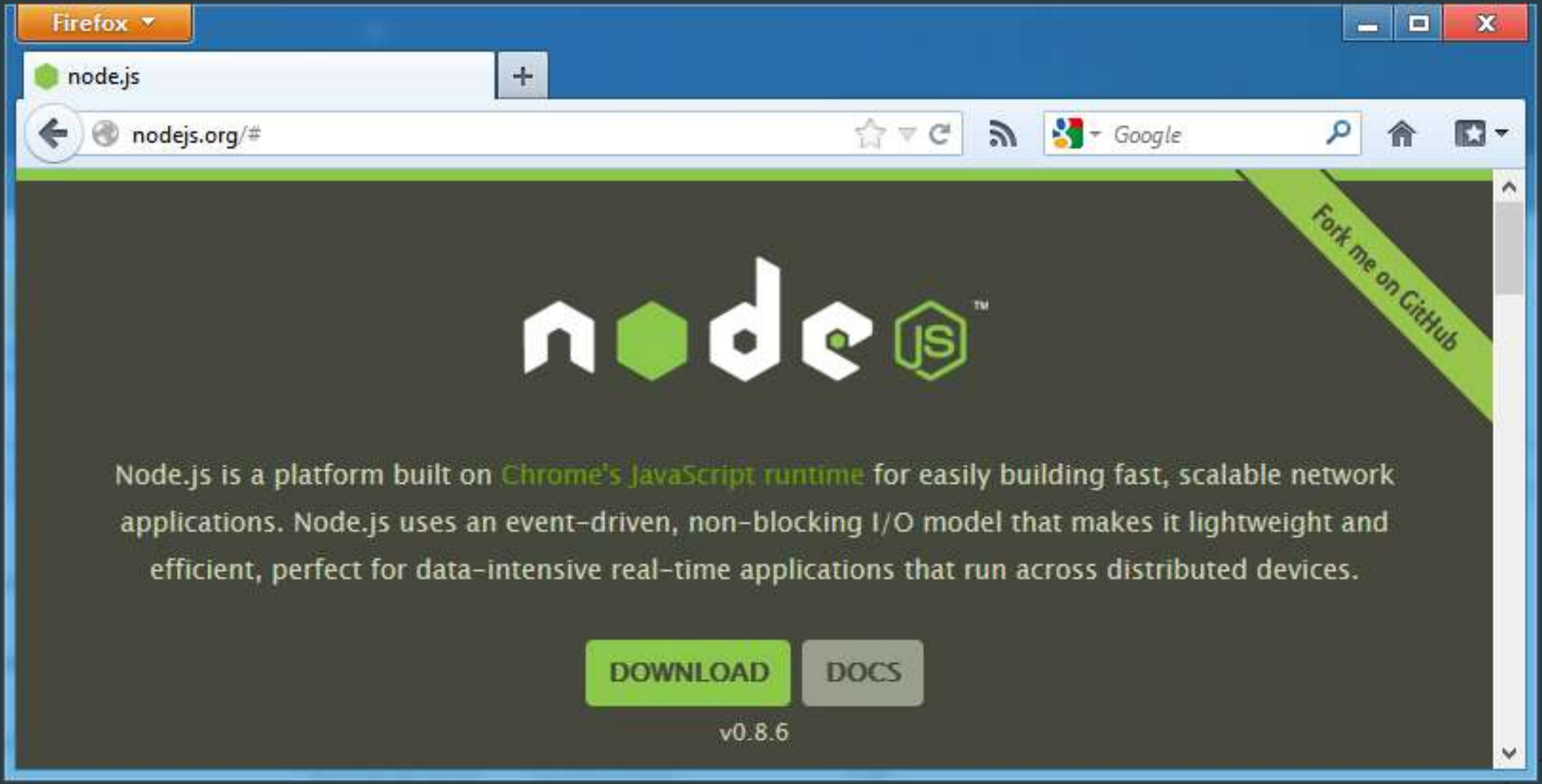
@domenic

https://npmjs.org/profile/domenicdenicola
http://github.com/domenic
http://github.com/NobleJS

agenda

- how to node
- why to node
- coding time

how to node

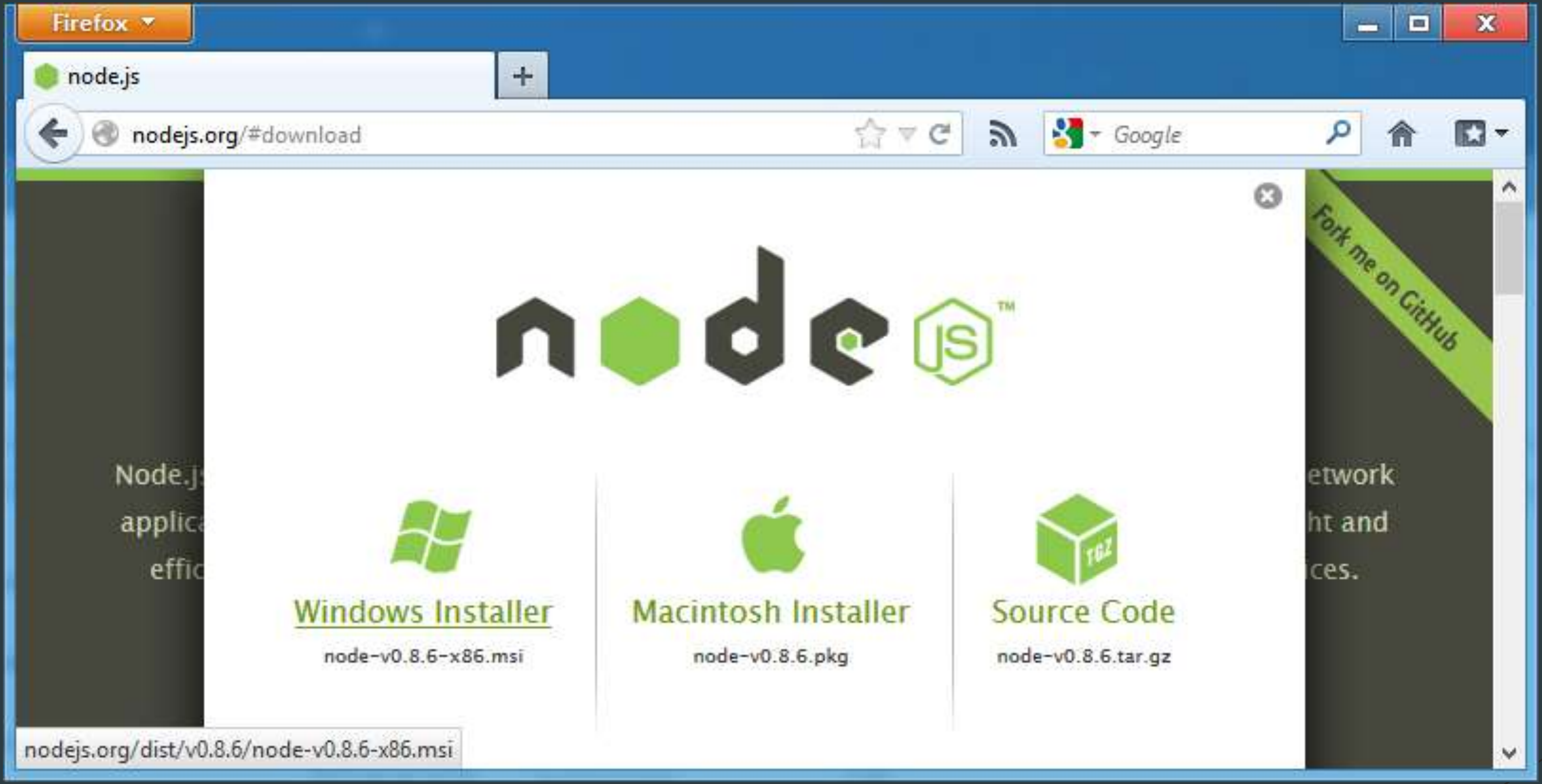


Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

v0.8.6

@domenic

how to node



Windows Installer
node-v0.8.6-x86.msi

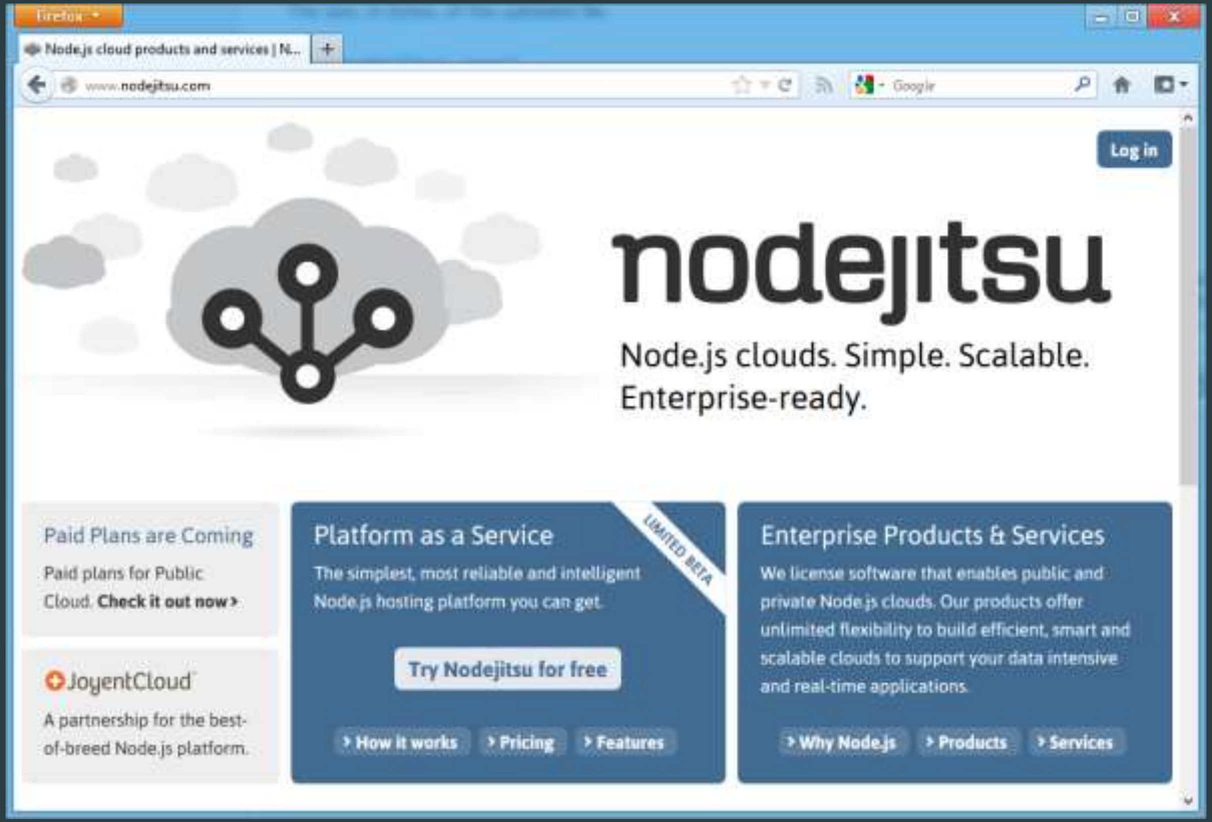
Macintosh Installer
node-v0.8.6.pkg

Source Code
node-v0.8.6.tar.gz

nodejs.org/dist/v0.8.6/node-v0.8.6-x86.msi

@domenic

how to node


Node.js cloud products and services | N...

www.nodejitsu.com

Log in

nodejitsu

Node.js clouds. Simple. Scalable. Enterprise-ready.

Paid Plans are Coming

Paid plans for Public Cloud. **Check it out now ›**

Platform as a Service

LIMITED BETA

The simplest, most reliable and intelligent Node.js hosting platform you can get.

Try Nodejitsu for free

› How it works › Pricing › Features

Enterprise Products & Services

We license software that enables public and private Node.js clouds. Our products offer unlimited flexibility to build efficient, smart and scalable clouds to support your data intensive and real-time applications.

› Why Node.js › Products › Services

JoyentCloud

A partnership for the best-of-breed Node.js platform.

@domenic

why to node

- new and shiny
- fast
- scalable
- low-level
- community

@domenic

new and shiny

let’s look at the most-used node.js packages.

@domenic

- socket.io: used by 306 other packages
- redis: 259 (hiredis: 70)
- stylus: 148 (less: 134)
- mongodb: 144 (mongoose: 135)

fast

@domenic

New HTTP Parser

I've implemented a new HTTP/1.1 request and response parser by hand. (My previous parser was written with the help of Ragel.) It requires 124 bytes per HTTP connection, makes zero allocations, has no dependencies, is nearly optimal in its use of CPU instructions, interruptible on any character, has extensive tests, and is MIT licensed.

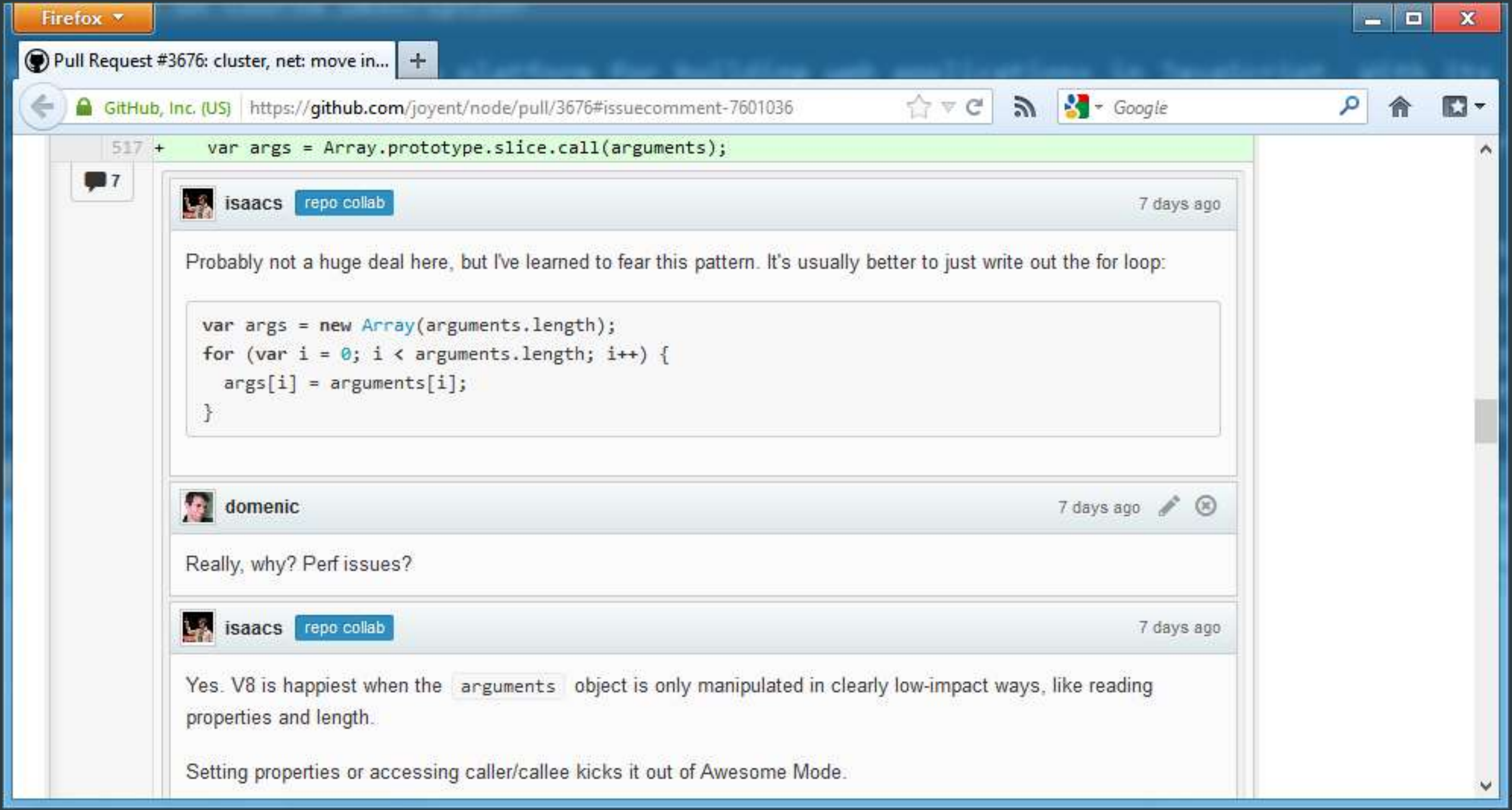
README

http_parser.h

http_parser.c

(Only one user at the moment: I've just merged it into Node.)

http://four.livejournal.com/1033160.html


Firefox
Pull Request #3676: cluster, net: move in...
GitHub, Inc. (US)
https://github.com/joyent/node/pull/3676#issuecomment-7601036
Google

```
517 +    var args = Array.prototype.slice.call(arguments);
```

isaacs repo collab 7 days ago

Probably not a huge deal here, but I've learned to fear this pattern. It's usually better to just write out the for loop:

```
var args = new Array(arguments.length);
for (var i = 0; i < arguments.length; i++) {
  args[i] = arguments[i];
}
```

domenic 7 days ago

Really, why? Perf issues?

isaacs repo collab 7 days ago

Yes. V8 is happiest when the `arguments` object is only manipulated in clearly low-impact ways, like reading properties and length.

Setting properties or accessing caller/callee kicks it out of Awesome Mode.

@domenic

scalable

q: what do web servers actually do?

a: i/o

@domenic

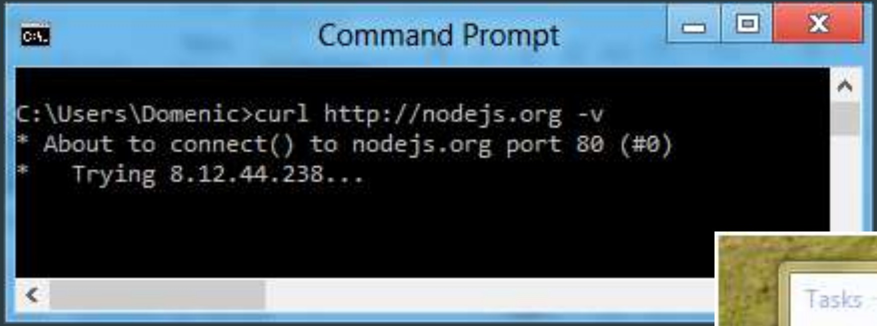
```
Response.Write("hello, world!");
```

```
move_uploaded_file(
    $_FILES['userphoto']['tmp_name'],
    "/var/www/userphotos/" . $_POST['username']
);
```

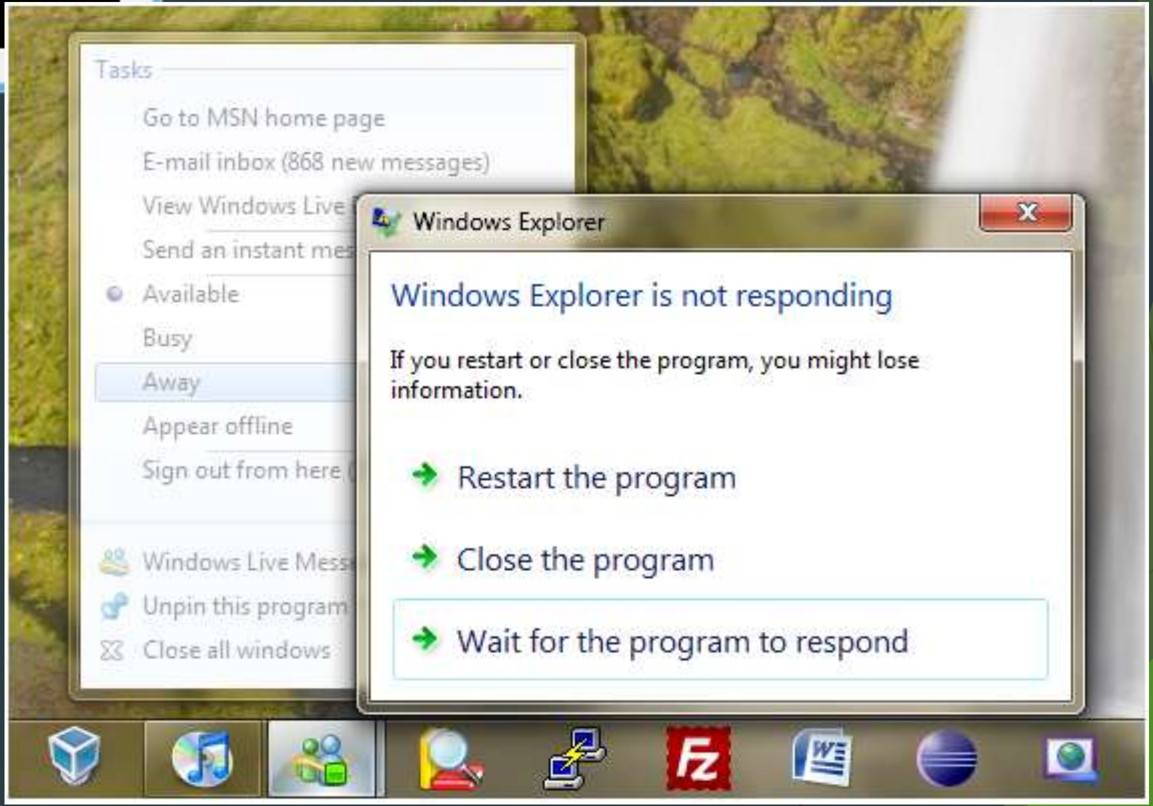
```
import memcache

mc = memcache.Client(['127.0.0.1:11211'])

mc.set("heavily_used_data", "data")
value = mc.get("more_data")
```

```
class Post < ActiveRecord::Base
    attr_accessible :content, :name, :title

    validates :name,  :presence => true
    validates :title, :presence => true
end
```

move this stuff out of the context of the web for a moment



```
C:\Users\Domenic>curl http://nodejs.org -v
* About to connect() to nodejs.org port 80 (#0)
*   Trying 8.12.44.238...
```



Windows Explorer

Windows Explorer is not responding

If you restart or close the program, you might lose information.

Restart the program

Close the program

Wait for the program to respond

@domenic

q: how do last-generation web servers fix this problem?

a: threads

let’s talk about *threads*.

they suck.

the end.

@domenic

q: how does node solve this problem?

a: javascript

My next project

I'm going to write a special thin web server tied to the V8 javascript interpreter

The web server will execute javascripts in response to requests in real-time.

The beautiful thing about this is that I can lock the users in an evented box where they have no choice but to be fast. They cannot touch the disk. They cannot access a database with some stupid library that blocks because they have no access to blocking I/O. They cannot resize an image by linking imagemagick into the web server process (and thus slowing down/blocking the entire thing). They cannot crash it because they are very limited in what they can do.

Why javascript?

- because its bare and does not come with I/O APIs
- web developers use it already
- DOM API is event-based. Everyone is already used to running without threads and on an event loop already.

I think this design will be extremely efficient and support very high loads. Web requests are transformed into mysql, memcached, AMQP requests and then return to the event loop.

Web developers need this sort of environment where it is not possible for them to do stupid things. Ruby, python, C++, PHP are all terrible languages for web development because they allow too much freedom.

http://four.livejournal.com/963421.html

@domenic

javascript has never had blocking i/o

javascript has never had more than one thread

instead we use *callbacks* and *events*

```
$.get("http://nodejs.org", function (data) {
    document.body.innerHTML = data;
});
```

```
var dbReq = indexedDB.open("my-database");

dbReq.addEventListener("success", function () {
    var dbConnection = dbReq.result;
});
```

it’s the same in node

```
fs.readFile("/etc/passwd", function (err, data) {
    console.log(data);
});
```

@domenic

```
request.on("data", function (chunk) {
    response.write(chunk);
});
```

```
db.users.find({ name: "domenic" }, function (err, users) {
    users.forEach(function (user) {
        response.write(user);
    });
});
```

```
io.sockets.on("connection", function (socket) {
    socket.emit("news", { hello: "world" });
    socket.on("my other event", function (data) {
      console.log(data);
    });
});
```

low-level

▶ STDIO ▶ Timers ▶ Process ▶ Utilities
▶ Events ▶ Domain ▶ Buffer ▶ Stream
▶ Crypto ▶ TLS/SSL ▶ String Decoder
▶ File System ▶ Path ▶ Net ▶ UDP/Datagram
▶ DNS ▶ HTTP ▶ HTTPS ▶ URL ▶ Query Strings
▶ Punycode ▶ Readline ▶ REPL ▶ VM
▶ Child Processes ▶ Assertion Testing ▶ TTY
▶ ZLIB ▶ OS ▶ Cluster

http://nodejs.org/docs/latest/api/

that’s it.

that’s all you get.

community

npm

github

SOCIAL CODING

@domenic

codingTime();

https://github.com/domenic/understanding-node

@domenic