Getting Started with node.js

Beardsley (“B”) Ruml
b@ruml.com
http://ruml.com

probably the most exciting new idea since Ruby on Rails in 2004

node.js
2009
Ryan Dahl

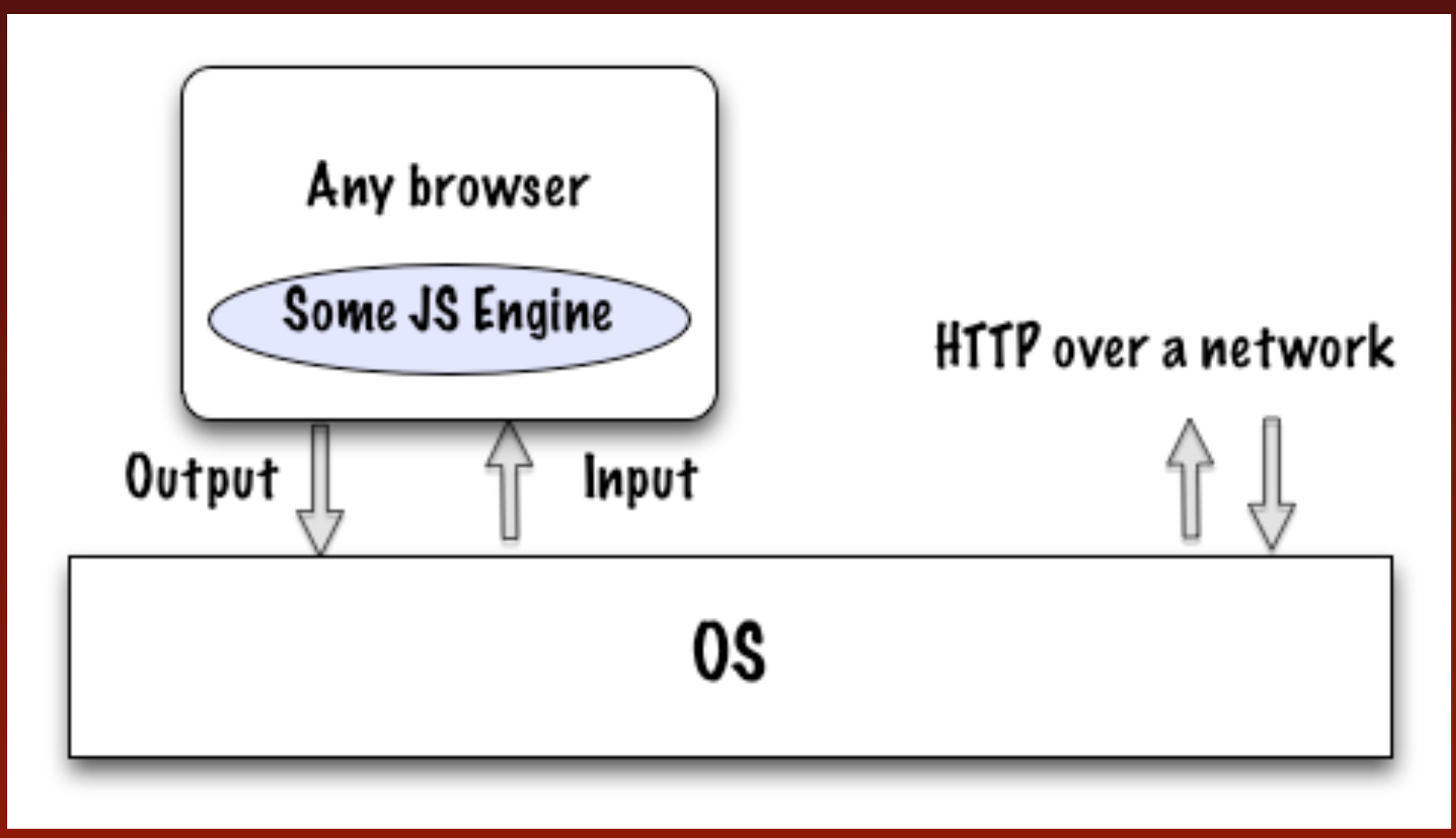
What is node.js?

What is node.js?

a program execution environment for JS

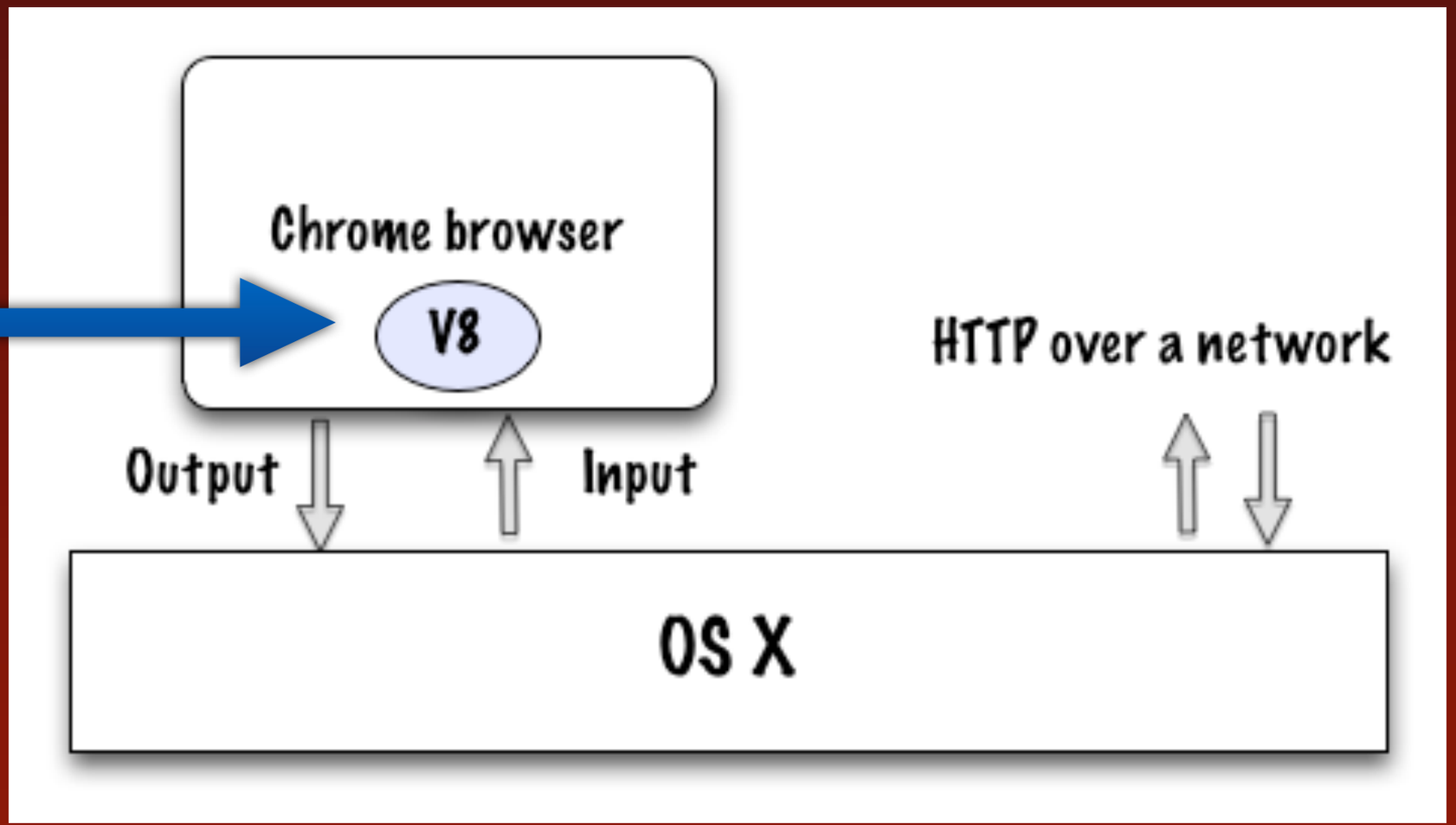
A browser is

a program execution environment for JS



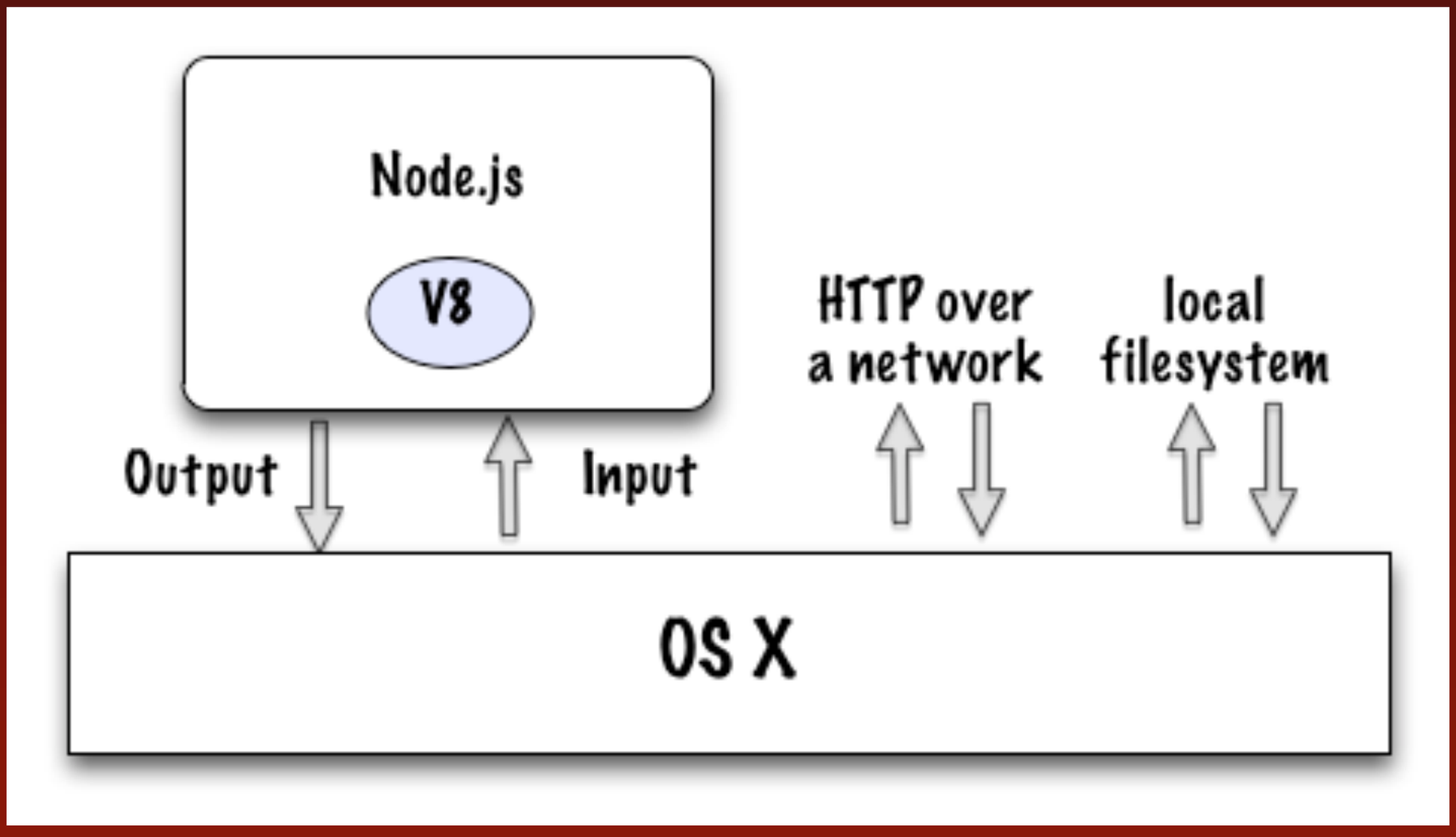
The Chrome browser is

a program execution environment for JS



node.js is

a program execution environment for JS



with access to the local filesystem

What is node.js?

a program execution environment for JS with event-driven, non-blocking I/O

event-driven:

On launch, the program sits in an event-loop waiting for GUI events to occur; when they do, each event is placed at the end of an event-queue.

event-driven(2):

The program removes each event from the head of the queue and "handles" it by invoking the event-handler functions which have been bound to the event.

“event-handler”

==

“callback”

The program “calls back” when the event occurs.

non-blocking I/O:

When input is requested (e.g., disk-access), the program doesn't wait for the data from disk. It continues on with "something else."

non-blocking I/O:

The code to be executed when the disk data is finally available is placed in a "callback function" and that function is invoked when the data-available event occurs.

What is node.js?

a program execution environment for JS with event-driven, non-blocking I/O written in C++ and incorporating Google's V8 JavaScript engine

[also a set of core modules]

Why is it so fast as a network server?

In CGI, each request spawns a new thread with a separate instance of the application: new interpreter, new initializaton, etc.

In node.js, each request triggers a callback within a single thread (small heap memory allocation)which provides an environment which can save state.

Why is node.js “the future” of web applications?

Web evolution:

- **a series of linked static pages;**
- **pages which were dynamically generated from a database (templating);**
- **pages which send significant amounts of new data and can ask for updates for parts of a page without refreshing the page (Ajax);**
- **pages which need constant communication; updates without asking; many browsers talking to each other (chat, backchannel);**

node.js core modules

- http
- net
- child_process
- fs
- os
- sys
- url
- util
- . . .

require(<module>)

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

[convention: use the same name as module]

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});
server.listen(1337);
console.log('Server running on port 1337');
```

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});
server.listen(1337);
console.log('Server running on port 1337');
```

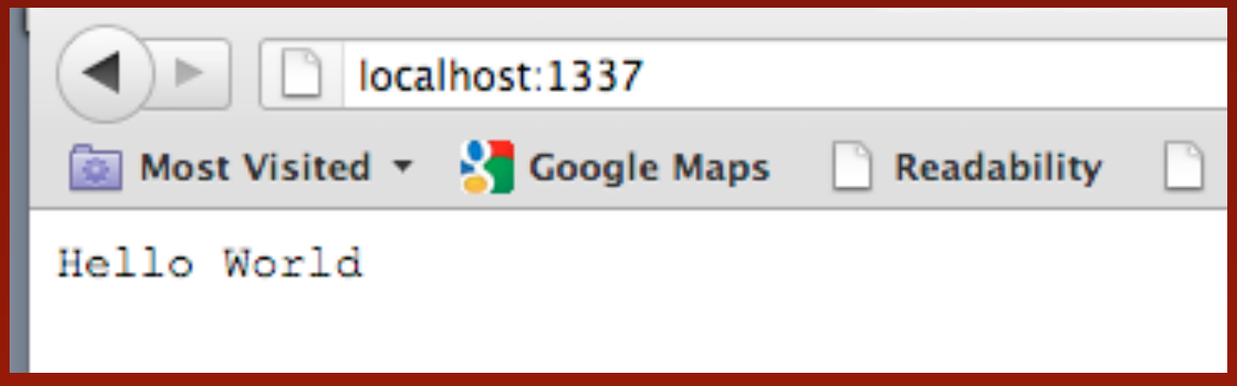
```
[ruml@TheBoss-784 code]$ node helloWorldServer.js
Server running on port 1337
```

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});
server.listen(1337);
console.log('Server running on port 1337');
```

```
[ruml@TheBoss-784 code]$ node helloWorldServer.js
Server running on port 1337
```



what does the call to createServer() do?

```
server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});
```

what does the call to createServer() do?

```
server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});
```

It binds the event-handler function (passed as the parameter) with the incoming-HTTP-request event.

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});
server.listen(1337);
console.log('Server running on port 1337');
```

but we can chain our calls!

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(1337);
console.log('Server running on port 1337');
```

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(1337);
console.log('Server running on port 1337');
```

what's wrong here?

a web server in node:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(1337, function() {
  console.log('Server running on port 1337');
});
```

the serverResponse object:

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

server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(1337, function() {
  console.log('Server running on port 1337');
});
```

the serverRequest object:

- request.method
- request.url
- request.headers
- request.trailers
- request.httpVersion
- request.setEncoding(encoding=null)
- request.pause()
- request.resume()
- request.connection

let's log the requests:

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

function logRequest(request) {
  console.log("REQUEST: " + request.method + " HTTP " +
              request.httpVersion + " " + request.url);
  console.dir(request.headers);
}


server = http.createServer(function (request, response) {
  logRequest(request);
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(1337, function() {
  console.log('Server running on port 1337');
});
```

let's log the requests:

```
^C[ruml@TheBoss-784 code]$ node helloWorldServer.js
Server running on port 1337
REQUEST: GET HTTP 1.1 /
{ host: 'localhost:1337',
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X
  accept: 'text/html,application/xhtml+xml,application/x
  'accept-language': 'en-us,en;q=0.5',
  'accept-encoding': 'gzip, deflate',
  'accept-charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.7',
  connection: 'keep-alive',
  'cache-control': 'max-age=0' }
```

let's serve a static file:

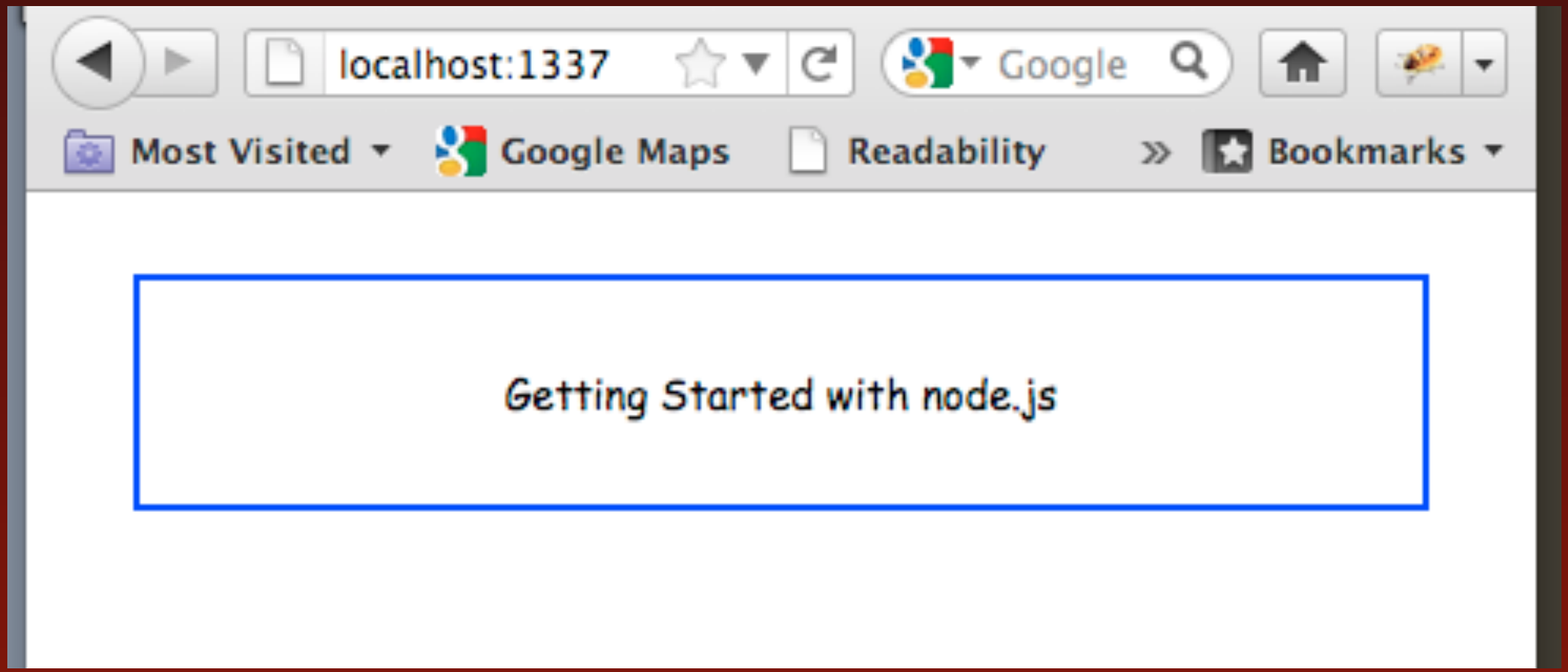
```
var http = require('http');
var fs   = require('fs');

server = http.createServer(function (request, response) {
  fs.readFile('./home.html', function(err, data) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end(data);
  });
}).listen(1337, function() {
  console.log('Server running on port 1337');
});
```

let's serve a static file:

```
<html>
<head>
  <title>Getting Started with node.js</title>
</head>
<body>
  <div id="container"
       style="margin: 2em;
              border: 2px solid blue;
              padding: 2em;
              text-align: center;">
    Getting Started with node.js
  </div>
</body>
</html>
```

let's serve a static file:



Web App Framework: Express

```
var app = require("express").createServer();

app.get('/', function(req, res){
    res.send('Hello World');
});

app.listen(3000);
```

A Chat Application

(the “demo app” of node.js)

How does chat work?

Joe, Sue and Bob all go to the same URL.
They navigate to the same “chat room.”

How does chat work?

How does chat work?

1. When anyone sends a message, the sender's ID and the message are displayed in the browsers of all. (In the sender's browser, the sender is "you.")

How does chat work?

1. When anyone sends a message, the sender's ID and the message are displayed in the browsers of all. (In the sender's browser, the sender is "you.")

2. When a new person enters (or leaves) the room, everyone else is notified.

How does chat work?

1. When anyone sends a message, the sender's ID and the message are displayed in the browsers of all. (In the sender's browser, the sender is "you.")

2. When a new person enters (or leaves) the room, everyone else is notified.

3. The newcomer is sent the most recent messages.

This isn't typical client-server!

This requires a: permanently-open bidirectional channel.

Before HTML5: hacks!

long polling
flash

After HTML5:

websockets!

(a permanently-open bidirectional channel)

websockets for node.js:

socket.io!

5 different transports
(including long polling and flash)

JavaScript object syntax:

```
{
   firstName: “John”,
   lastName: “Brown”,
   age: 23,
   children: [ “Sue”, “Bob”],
   birthdate: [ 1994, 12, 25 ]
}
```

back to the chat app:

What messages do we need to send?

What messages do we need to send?

1. When anyone sends a message, the sender's ID and the message are displayed in the browsers of all. (In the sender's browser, the sender is "you.")

Message from the sender's browser:

```
<body>
```

Message from the server to others:

```
{ message: [<author>, <body>] }
```

What messages do we need to send?

2. When a new person enters (or leaves) the room, everyone else is notified.

Message from the server to others:

```
{ announcement: <text> }
```

What messages do we need to send?

3. The newcomer is sent the most recent messages.

Message from the server to newcomer:

```
{ buffer: [ <msg>, <msg>, ... ] }
```

Chat app messages summary:

Messages from the browser:

```
<body>
```

Messages from the server:

```
{ buffer: [ <msg>, <msg>, ... ] }

{ announcement: <text> }

{ message: [<author>, <body>] }
```

What has to happen on the server?

1. Listen for attempts to connect.
2. When a connection occurs:
   - create a client object and assign an ID;
   - bind event-handlers to the events of the client: 1) receipt-of-a-message and 2) disconnecton
3. The message event-handler needs to:
   - add the client's ID as author;
   - package author and body into a message object;
   - sent the message object to everyone else.

Let's look at the server code:

```
var serverListener = require('socket.io').listen(server);
var buffer = [];

serverListener.on('connection', function(client){
  // send the buffer to this client;
  client.send({ buffer: buffer });
  // send an announcement to all other clients;
  client.broadcast({ announcement: '=> ' + client.sessionId + ' connected' });

  client.on('message', function(message){
    var msg = { message: [client.sessionId, message] };
    buffer.push(msg);
    // maximum of 15 messages in the buffer at once;
    if (buffer.length > 15) buffer.shift();
    client.broadcast(msg);
  });

  client.on('disconnect', function(){
    client.broadcast({ announcement: client.sessionId + ' disconnected' });
  });
});
```

What has to happen on the browser?

1. Create a socket object and send a connection request to the server.

2. Bind event-handler to receipt-of-message event.

3. The message event-handler needs to:
   - for buffer objects: add to chat window in a loop;
   - otherwise just add to chat window.

Let's look at the browser code:

```
<script src="/js/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost:8888');
  socket.on('message', function(obj){
    if ('buffer' in obj){
      for (var i in obj.buffer) { addParaToChat(obj.buffer[i]); }
    } else {
      addParaToChat(obj);
    }
  });

  socket.on('connect', function(){
    addParaToChat({ message: ['System', 'Connected']})
  });
  socket.on('disconnect', function(){
    addParaToChat({ message: ['System', 'Disconnected']})
  });
</script>
```

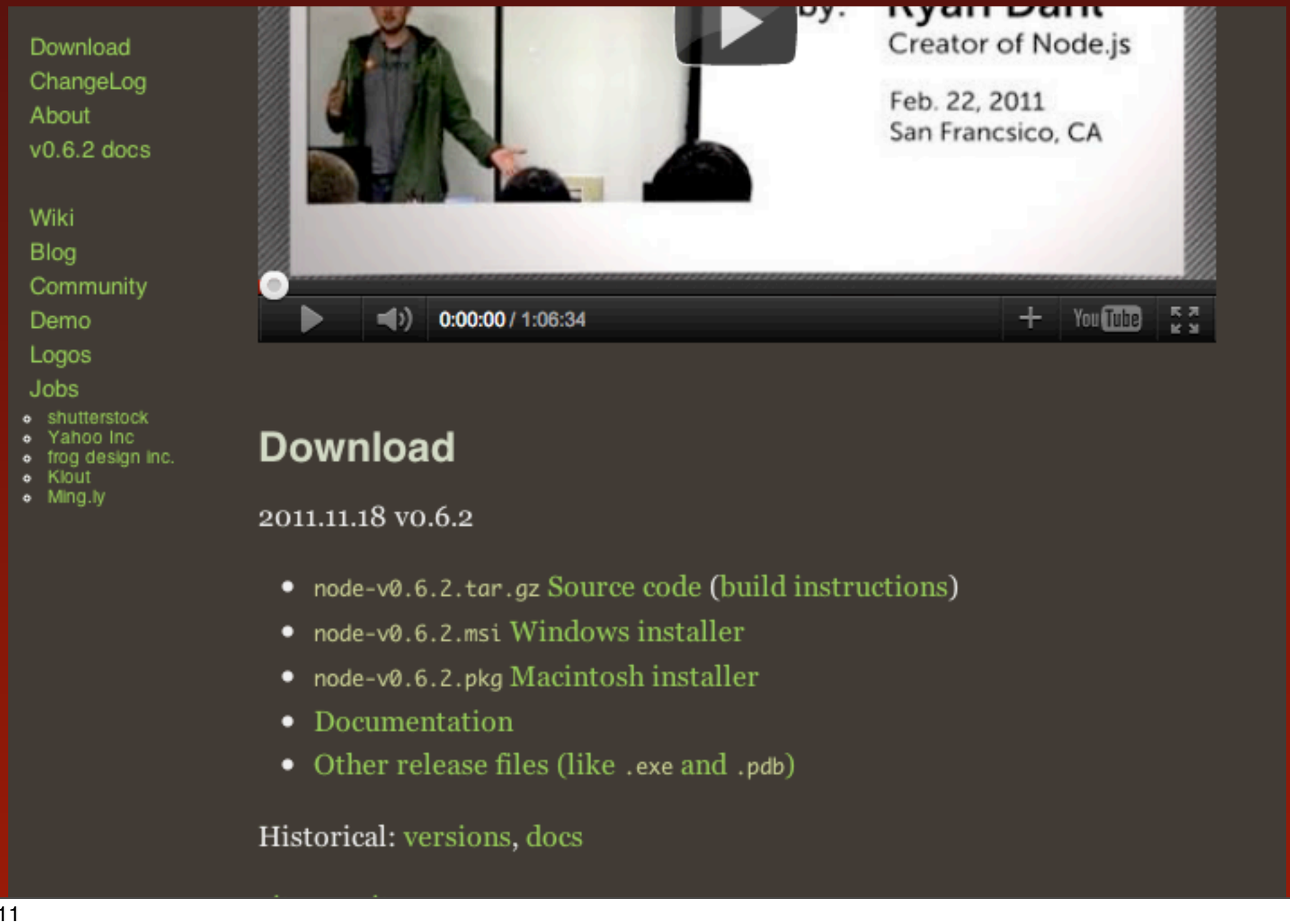
backchannel

A variation on chat: posts (typically questions for a lecturer) are voted on by all and can be displayed in order of popularity.

http://github.com/bruml2/backchannel

node.js installation

http://nodejs.org



node.js documentation

http://nodejs.org/docs/v0.6.2/api/all.html

Node.js v0.6.2 Manual & Documentation

Index | View on single page

Table of Contents

node.js directory

nodecloud.org

NodeCloud is a resource directory gathering sites related to Node.js and ordering them by their Alexa traffic, allowing to evaluate relative popularity of a project. Screenshots are generated locally using PhantomJS.

To suggest new sites to be added to the listing, you can reach me through my site or on twitter.

Note to programmers : you might as well be interested in my Ascii Codes reference chart.

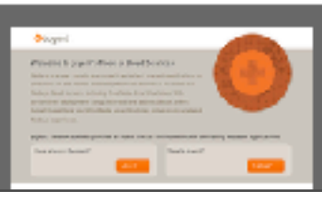
BREAKING NEWS : check out Echo Linux, a social news site dedicated to Linux and related topics!

Share this site :

Node.js

Home of the Node.js project, a network application framework written on top of Google V8 JavaScript engine.

NodeCloud Ranking : 1
Alexa Traffic Rank : 19039



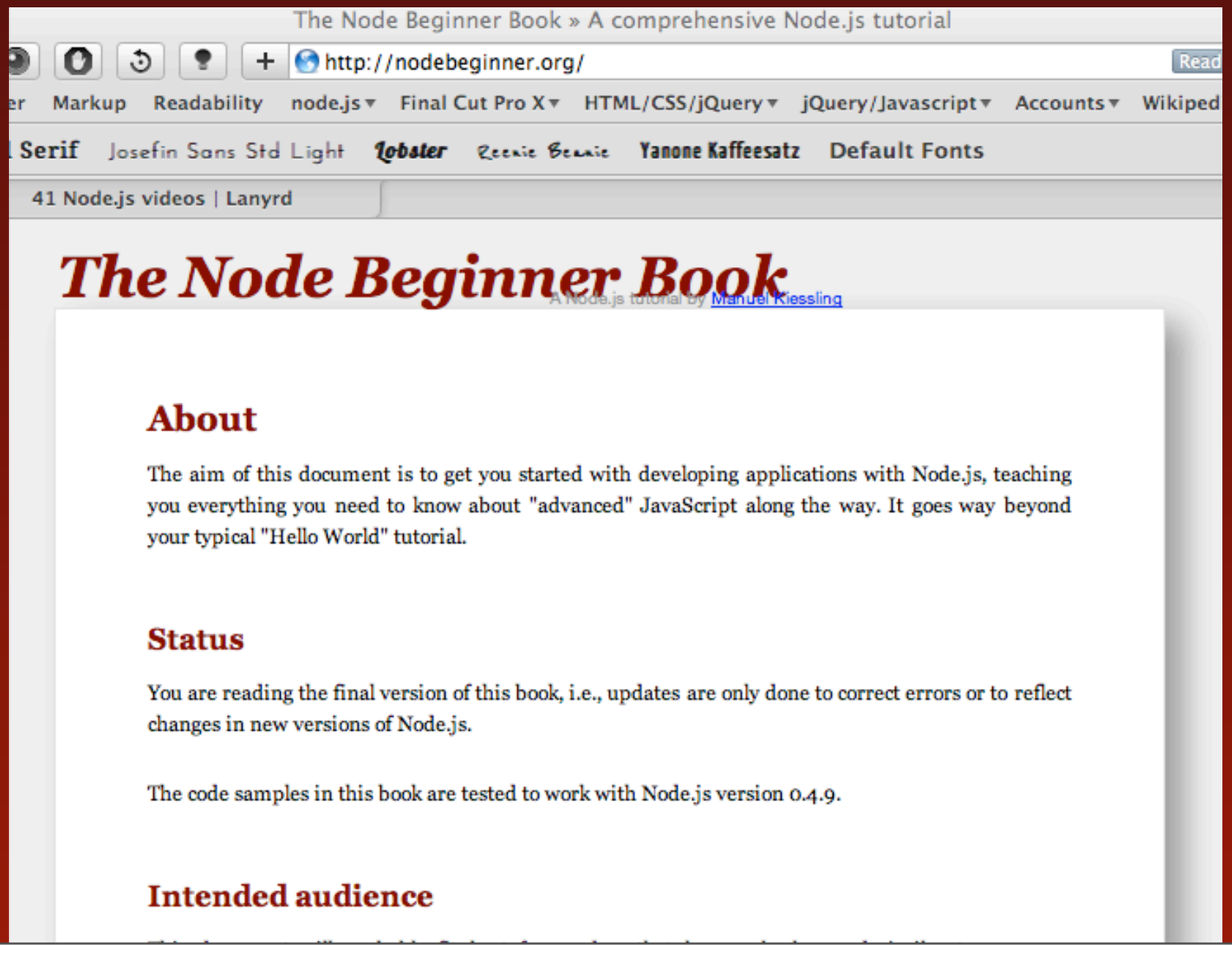
Node SmartMachines

Node.js cloud hosting services by Joyent, offering free Node SmartMachines.

node.js tutorials

The Node Beginner Book

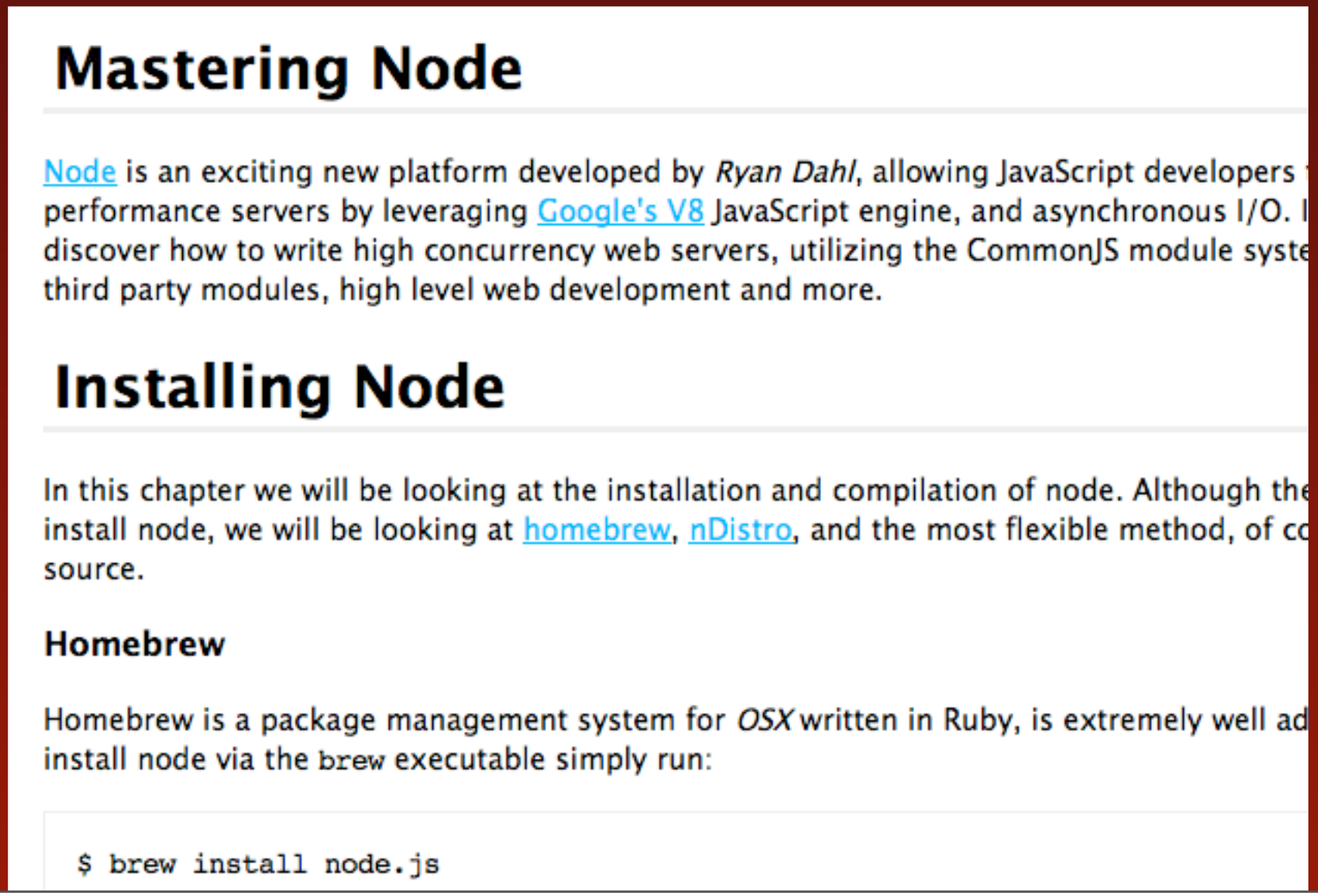
http://nodebeginner.org



node.js tutorials

Mastering Node

http://visionmedia.github.com/masteringnode/book.html



Mastering Node

Node is an exciting new platform developed by *Ryan Dahl*, allowing JavaScript developers
performance servers by leveraging Google's V8 JavaScript engine, and asynchronous I/O.
discover how to write high concurrency web servers, utilizing the CommonJS module syste
third party modules, high level web development and more.

Installing Node

In this chapter we will be looking at the installation and compilation of node. Although the
install node, we will be looking at homebrew, nDistro, and the most flexible method, of co
source.

Homebrew

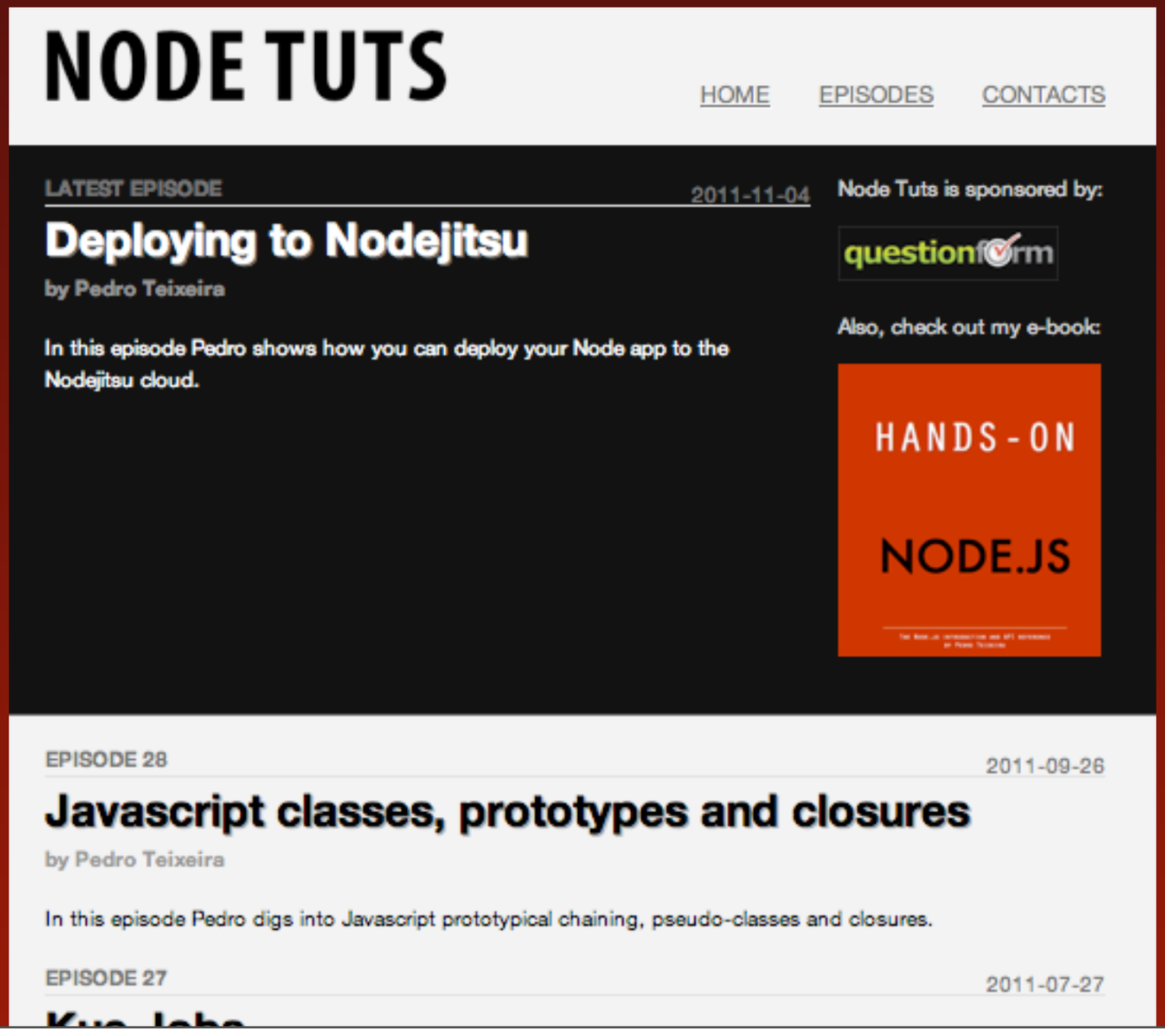
Homebrew is a package management system for *OSX* written in Ruby, is extremely well ad
install node via the `brew` executable simply run:

```
$ brew install node.js
```

node.js tutorials

Node Tuts screencasts

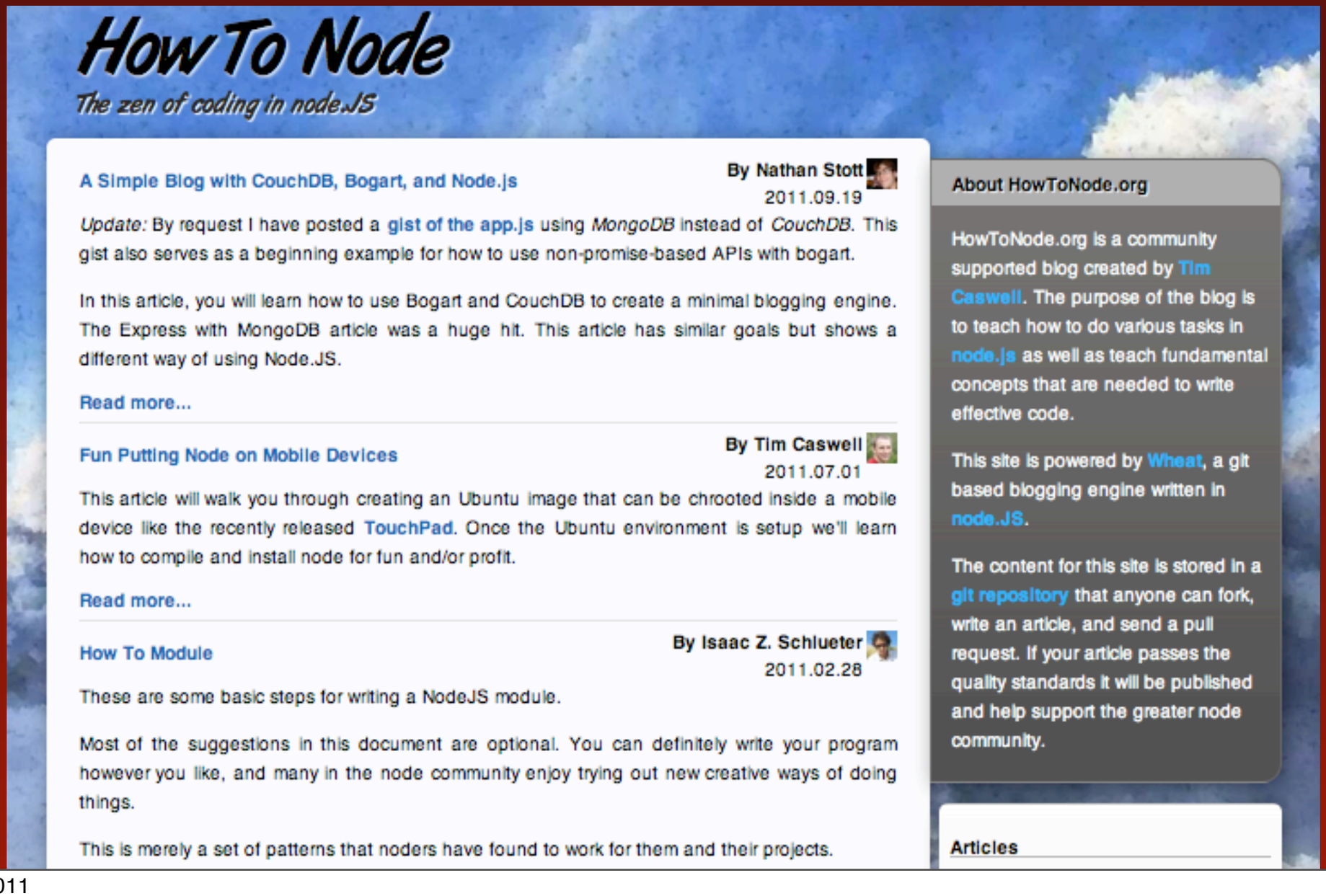
http://nodetuts.com



node.js tutorials

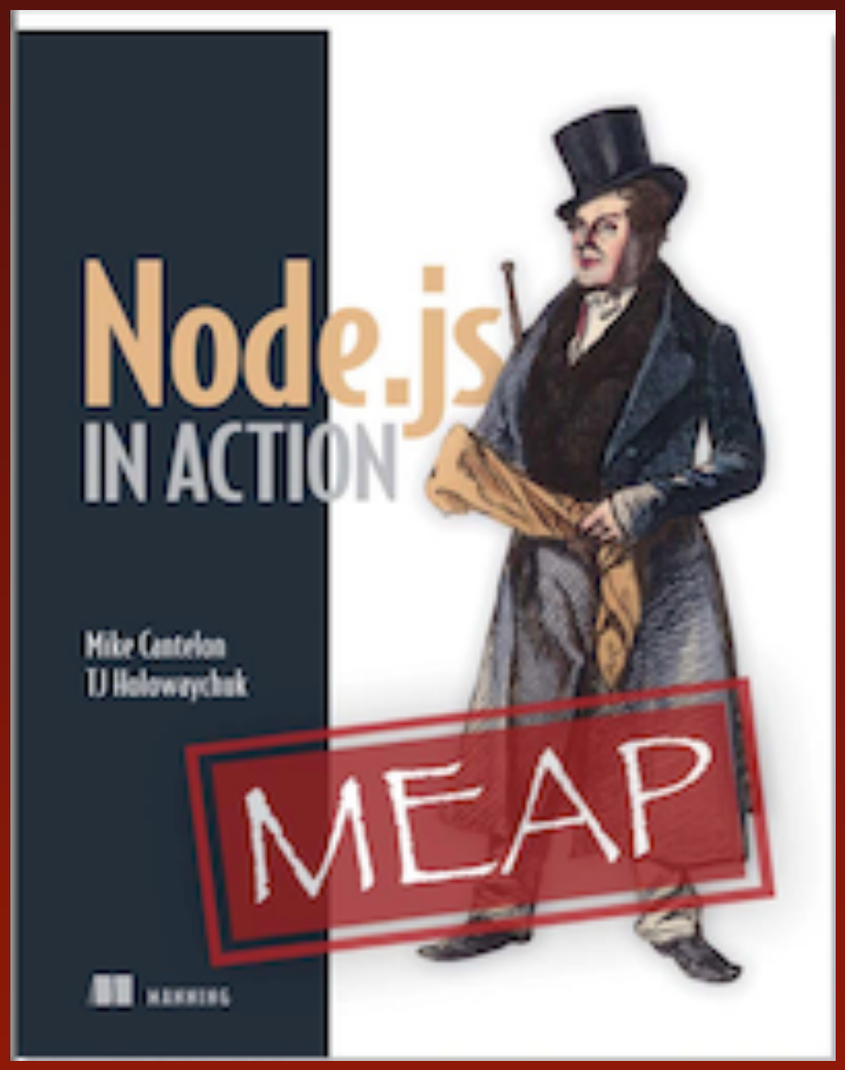
How To Node

http://howtonode.org



node.js books

Node.js in Action (Manning)

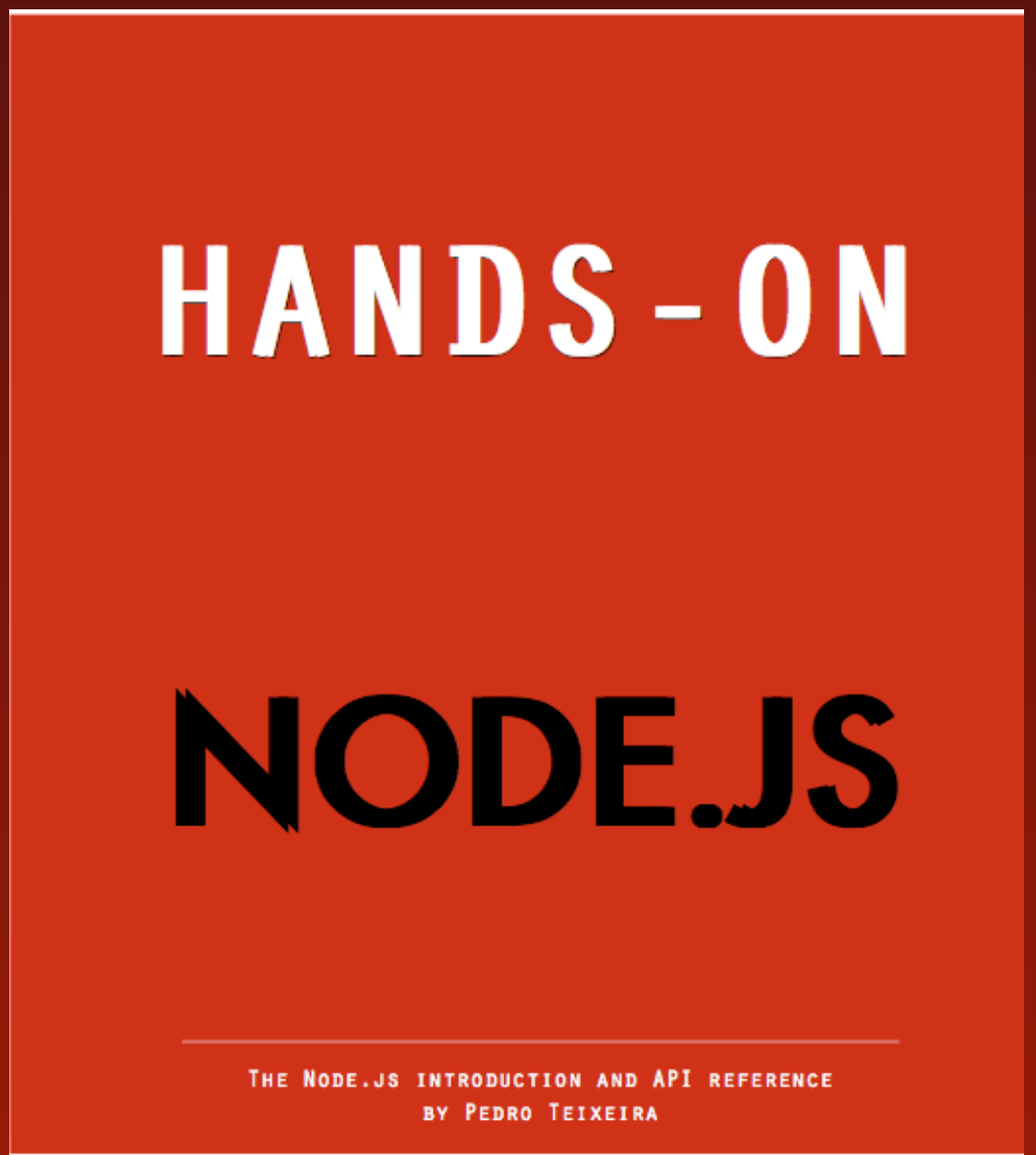


node.js books

Hands-on node.js

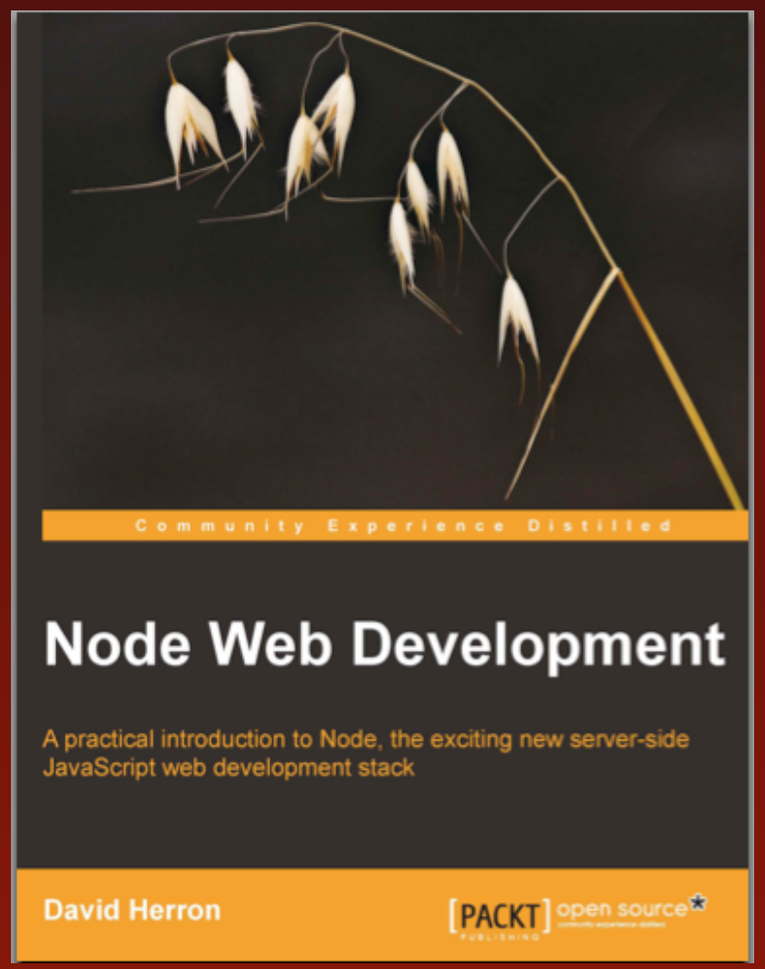
60 pages (of 118) free

http://nodetuts.com



node.js books

Node Web Development (Packt)



not so good!

node.js Hosting

Joyent (no.de)
Nodejitsu
Nodester
Heroku

easy; cheap;

Thanks for listening!