3.– 6. September 2012
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

# Jetzt auch auf einem Server in Ihrer Nähe

JavaScript & node.js

Ralph Winzinger

Senacor Technologies AG

```
        <a href="#" data-role="button" data-theme="c" data-rel="back">No</a>
    </div>
</div>

<script id="personTmpl" type="text/x-jquery-tmpl">
    <li>
        <a href="javascript:addPerson('${username}')"><img src="${iconLink}"
        <div class="ui-li-count">${numFollowers}</div>
    </li>
</script>

<script type="text/javascript">
    // dialogs
    function areYouSure(text1, text2, button, callback) {
        $("#sure .sure-1").text(text1);
        $("#sure .sure-2").text(text2);
        $("#sure .sure-do").text(button).on("click.sure", function() {
            callback();
            $(this).off("click.sure");
        });
        $.mobile.changePage("#sure", {role: 'dialog'})
}
```

Jetzt auch auf einem Server in Ihrer Naehe!

# Wer bin ich
# und was tu' ich hier?

- Ralph Winzinger
  Architekt bei Senacor Technologies, Nbg
- JavaScript
- node.js
- putting stuff together

# History, Part 1
# JavaScript

Brendan Eich, 1995, Netscape

# History, Part 1
# JavaScript

Brendan Eich, 1995, Netscape

Netscape Communicator

# History, Part 1
# JavaScript

Brendan Eich, 1995, Netscape

Netscape Communicator

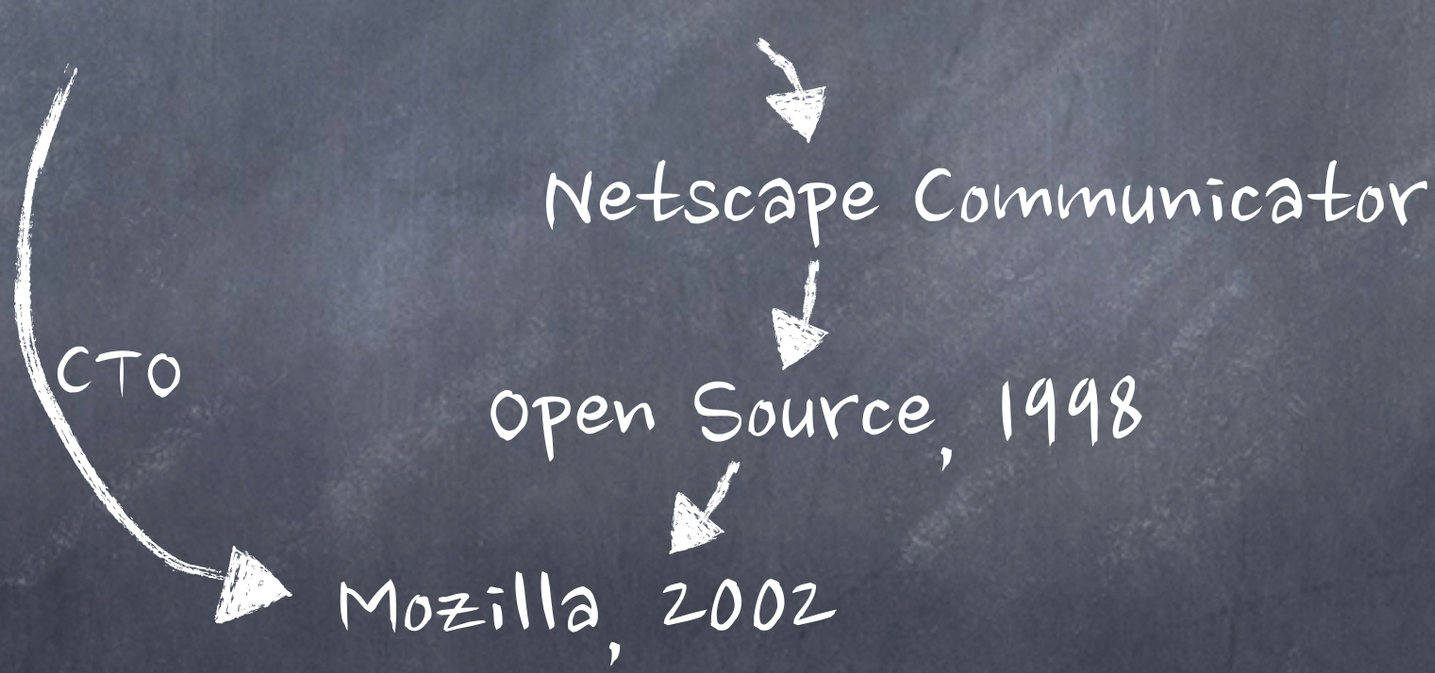
Open Source, 1998

# History, Part 1
# JavaScript

Brendan Eich, 1995, Netscape

# History, Part 1
# JavaScript

Brendan Eich, 1995, Netscape



# History, Part 1
# JavaScript

# History, Part 1

# JavaScript

Microsoft ⇔ Netscape

C++ ⇔ Java

VB ⇔

# History, Part 1

# JavaScript

Microsoft ⇔ Netscape

C++ ⇔ Java

VB ⇔ JS

# History, Part 1

# JavaScript

Microsoft ⇔ Netscape

C++ ⇔ Java

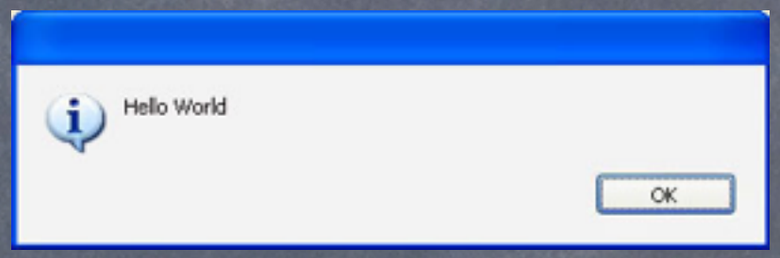
VB ⇔ JS

„Plus, it had to be done in ten days

or something worse had happend“

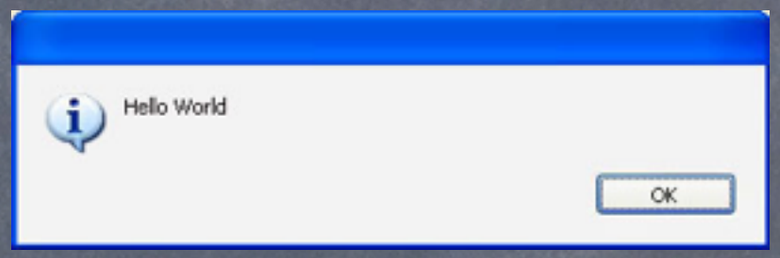
# Was ist JavaScript?

# Was ist JavaScript?




```
alert(„Hello World“);
```

# Was ist JavaScript?




```
alert(„Hello World“);
```

Y2k

Security Flaws

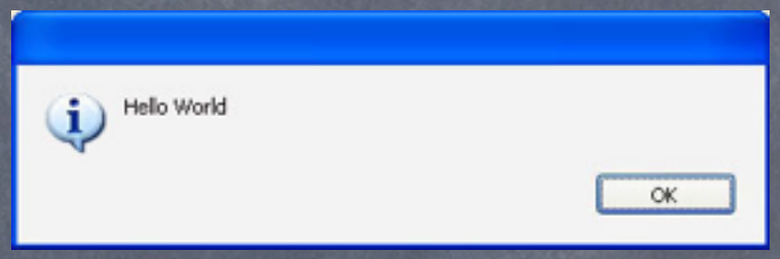
XSS

CPU-Load

Schlechte Erweiterungen

fuer HTML-Seiten

# Was ist JavaScript?




```
alert(„Hello World“);
```

Y2k

Security Flaws

XSS

CPU-Load

Schlechte Erweiterungen

fuer HTML-Seiten

AJAX

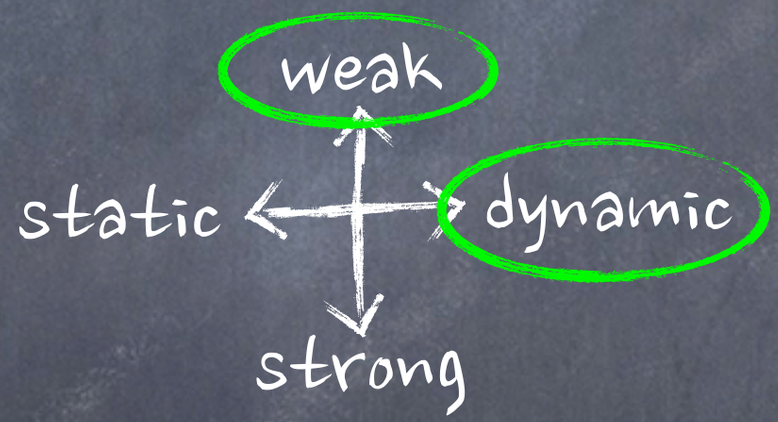
# Was ist JavaScript wirklich?

- eine (fast) ganz normale Sprache
- Typen: Number, String, Boolean, Object, Array


- Operatoren, Kontrollstrukturen, RegExp
- NaN, null, undefined
- Funktionen

# Was ist JavaScript wirklich?

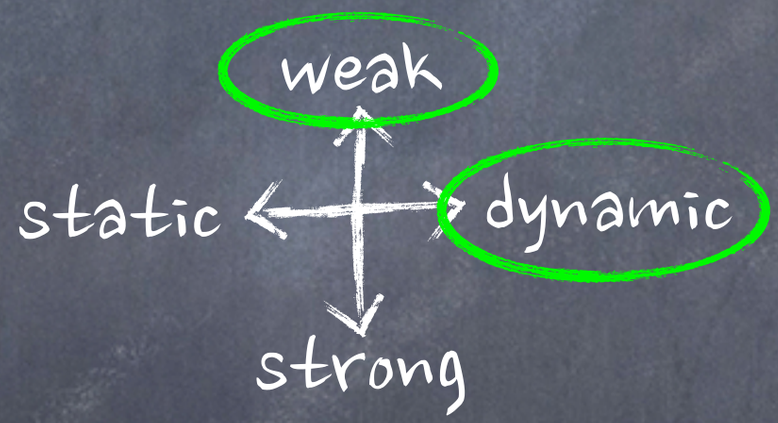
- eine (fast) ganz normale Sprache
- Typen: Number, String, Boolean, Object, Array




- Operatoren, Kontrollstrukturen, RegExp
- NaN, null, undefined
- Funktionen

# Was ist JavaScript wirklich?

- eine (fast) ganz normale Sprache
- Typen: Number, String, Boolean, Object, Array




- Operatoren, Kontrollstrukturen, RegExp
- NaN, null, undefined
- Funktionen

http://is.gd/hc2012

# Ein paar Eigenheiten

- == ↔ ===
- for (name in object)
- Scoping
- var ↔ (nicht var)
- this & that
- Funktionen sind first-class citizens
- object linkage

# Objekte

Im Wesentlichen Key-Value-Pairs

```
rw = {
  firstname:"ralph",
  lastname:"winzinger"
};
```

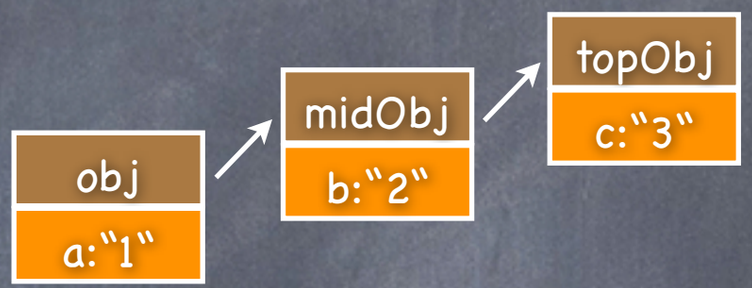
```
console.log(rw.firstname);
console.log(rw["lastname"]);
```

```
rw.title = "architect";
console.log(rw);
```

```
rw.dump = function() {
  console.log(this);
}
```

# Objekte

## Linkage & Prototype

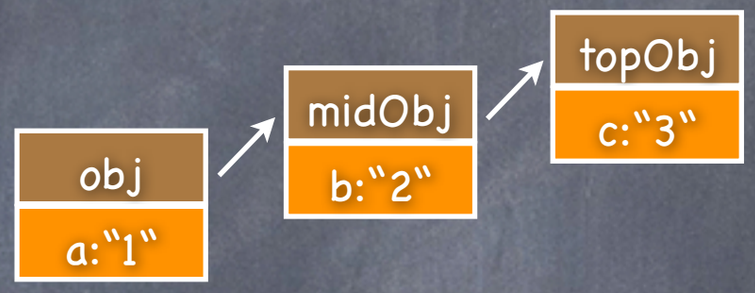


# Objekte

## Linkage & Prototype

```
obj = {a:"1"}
midObj = {b:"2"}
topObj = {c:"3"}

obj.__proto__ = midObj
midObj.__proto__ = topObj
```

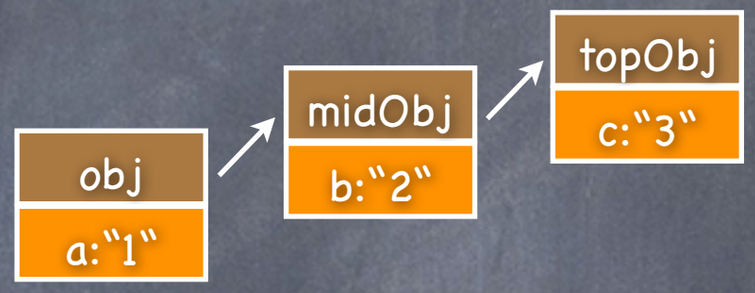


# Objekte

## Linkage & Prototype

```
obj = {a:"1"}
midObj = {b:"2"}
topObj = {c:"3"}

obj.__proto__ = midObj
midObj.__proto__ = topObj
```




## normalerweise via Konstruktor ...

```
// http://javascript.crockford.com/prototypal.html
var derive = function (o) {
    function F() {};
    F.prototype = o;
    return new F();
};
```

# Funktionen

## First-Class citizens, lambdas & closures

```
var greeter = function(name) {
  console.log(„hello “+name);
}
greeter(„ralph“);
```

# Funktionen

## First-Class citizens, lambdas & closures

```
var greeter = function(name) {
  console.log(„hello “+name);
}
greeter(„ralph“);
```

```
var obj = {a:“1“, foo:greeter};
obj.foo(„ralph“);
```

# Funktionen

## First-Class citizens, lambdas & closures

```
var greeter = function(name) {
  console.log(„hello “+name);
}
greeter(„ralph“);
```

first-class

```
var obj = {a:“1“, foo:greeter};
obj.foo(„ralph“);
```

# Funktionen

## First-Class citizens, lambdas & closures

```
var greeter = function(name) {
  console.log(„hello “+name);
}
greeter(„ralph“);
```

first-class

```
var obj = {a:“1“, foo:greeter};
obj.foo(„ralph“);
```

```
setTimeout(1000, function() {
                   console.log(„hello“);
                 });
```

# Funktionen

## First-Class citizens, lambdas & closures

```
var greeter = function(name) {
  console.log(„hello “+name);
}
greeter(„ralph“);
```

first-class

```
var obj = {a:“1“, foo:greeter};
obj.foo(„ralph“);
```

```
setTimeout(1000, function() {
                   console.log(„hello“);
                 });
```

lambda

# Funktionen

First-Class citizens, lambdas & closures

```
var name = "ralph";
var foo = function() {
    var xy = 1;
}
var greeter = function() {
    console.log("hello, "+name);
    console.log("xy: "+xy);
}
```

# Funktionen

## First-Class citizens, lambdas & closures

```
var name = "ralph";
var foo = function() {
    var xy = 1;
}
var greeter = function() {
    console.log("hello, "+name);
    console.log("xy: "+xy);
}
```

lexical scope

# Funktionen

## First-Class citizens, lambdas & closures

```
var name = "ralph";
var foo = function() {
    var xy = 1;
}
var greeter = function() {
    console.log("hello, "+name);
    console.log("xy: "+xy);
}

var wrapper = function() {
    var name = "ralph";
    var greeter = function() {
        console.log("hello, "+name);
    }
    greeter();
    return greeter;
}
```

lexical scope

# Funktionen

## First-Class citizens, lambdas & closures

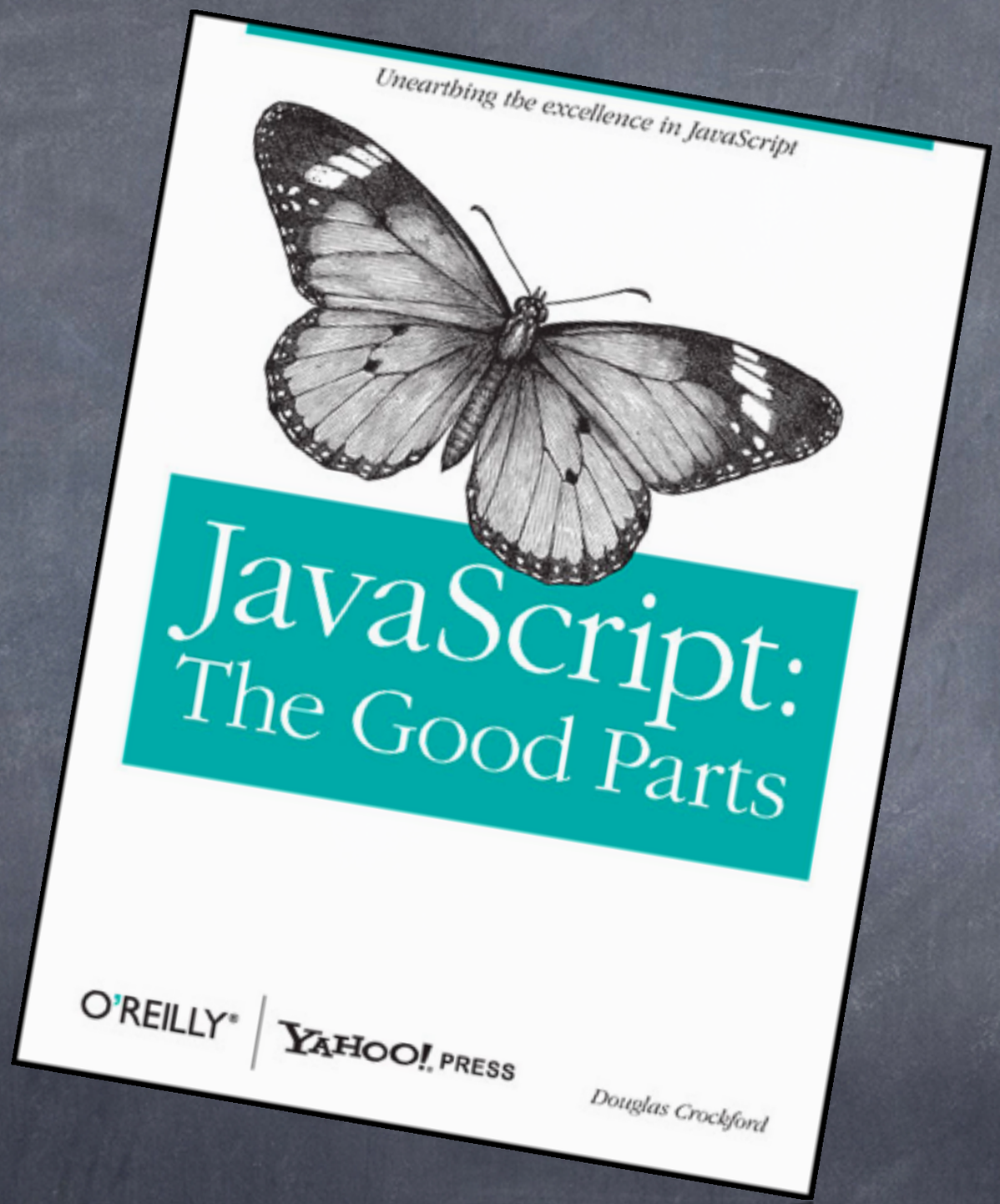
```
var name = "ralph";
var foo = function() {
    var xy = 1;
}
var greeter = function() {
    console.log("hello, "+name);
    console.log("xy: "+xy);
}

var wrapper = function() {
    var name = "ralph";
    var greeter = function() {
        console.log("hello, "+name);
    }
    greeter();
    return greeter;
}
```

lexical scope

closure

Mittwoch, 19. September 12


Unearthing the excellence in JavaScript
JavaScript:
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford

# History, Part 2

## node.js

Ryan Dahl, 2006, Mathematiker

# History, Part 2

## node.js

Ryan Dahl, 2006, Mathematiker

Suedamerika, Web-Programmer

# History, Part 2

## node.js

Ryan Dahl, 2006, Mathematiker

Suedamerika, Web-Programmer

# History, Part 2

# node.js

Ryan Dahl, 2006, Mathematiker

Suedamerika, Web-Programmer

JSConf Berlin, 2009

# History, Part 2

## node.js

Ryan Dahl, 2006, Mathematiker

Suedamerika, Web-Programmer

JSConf Berlin, 2009

Joyent Funding

# History, Part 2

# node.js

Ryan Dahl, 2006, Mathematiker

Suedamerika, Web-Programmer

JSConf Berlin, 2009

Joyent Funding

Employee

# History, Part 2

## node.js

Warum jetzt? Die Idee zu SSJS ist
doch schon 10 Jahre alt ... ?

Web 2.0 - AJAX, HTML5, jQuery

Mobile - HTML WebApps, jQueryMobile, PhoneGap

Cloud, Distribution, Networking

Performancezuwachs


# Die node-Idee

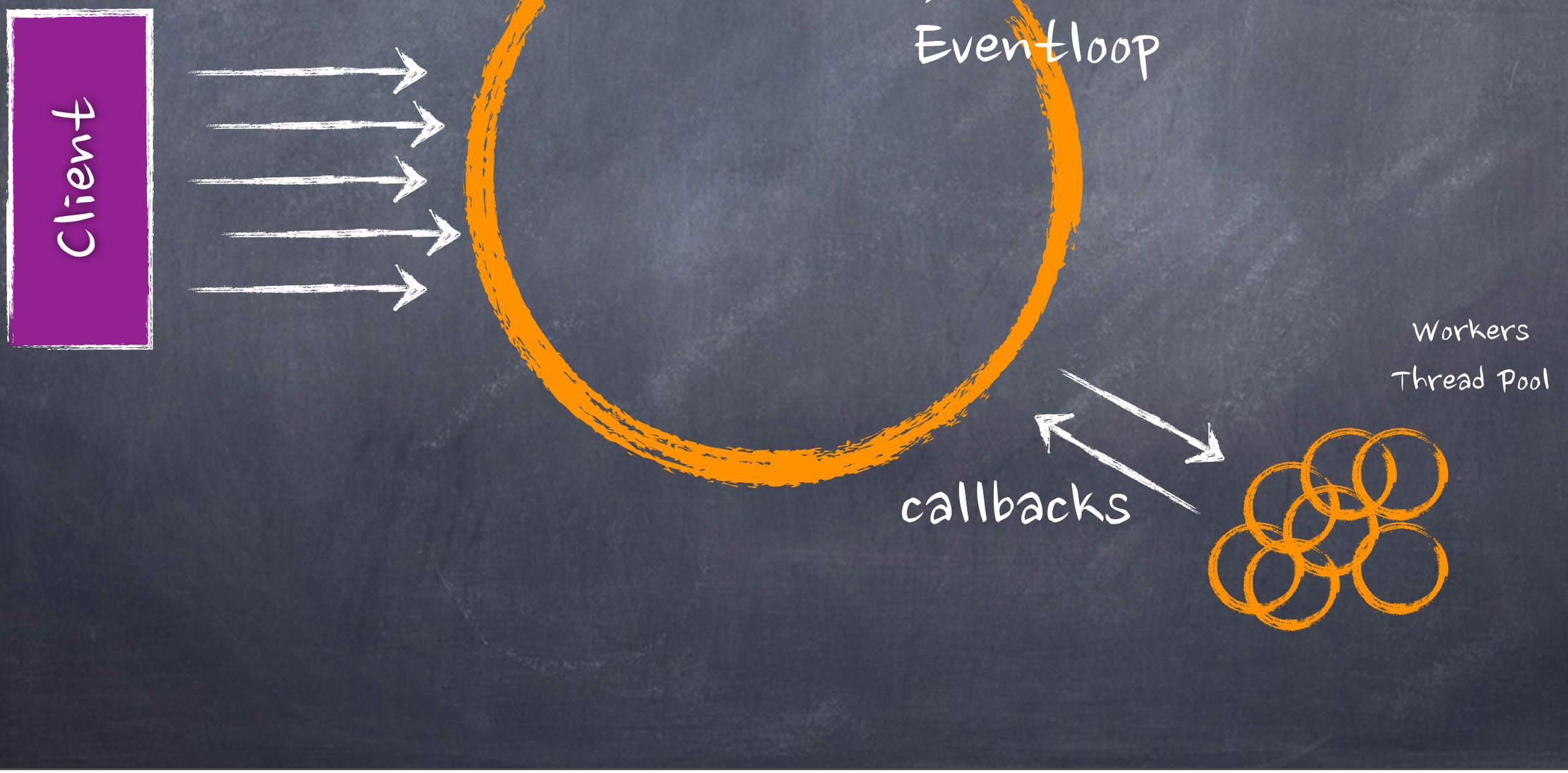
Single Threaded

# Die node-Idee

Single Threaded
Eventloop

# Die node-Idee

# Die node-Idee



# node.js

- Installation
- Laufzeitumgebung
- Docs
- Modules
- Debugging

# node.js

- Installation
- Laufzeitumgebung
- Docs
- Modules
- Debugging
- Chat Anwendung

# Installation

1. www.nodejs.org
2. „install“
3. http-Server schreiben

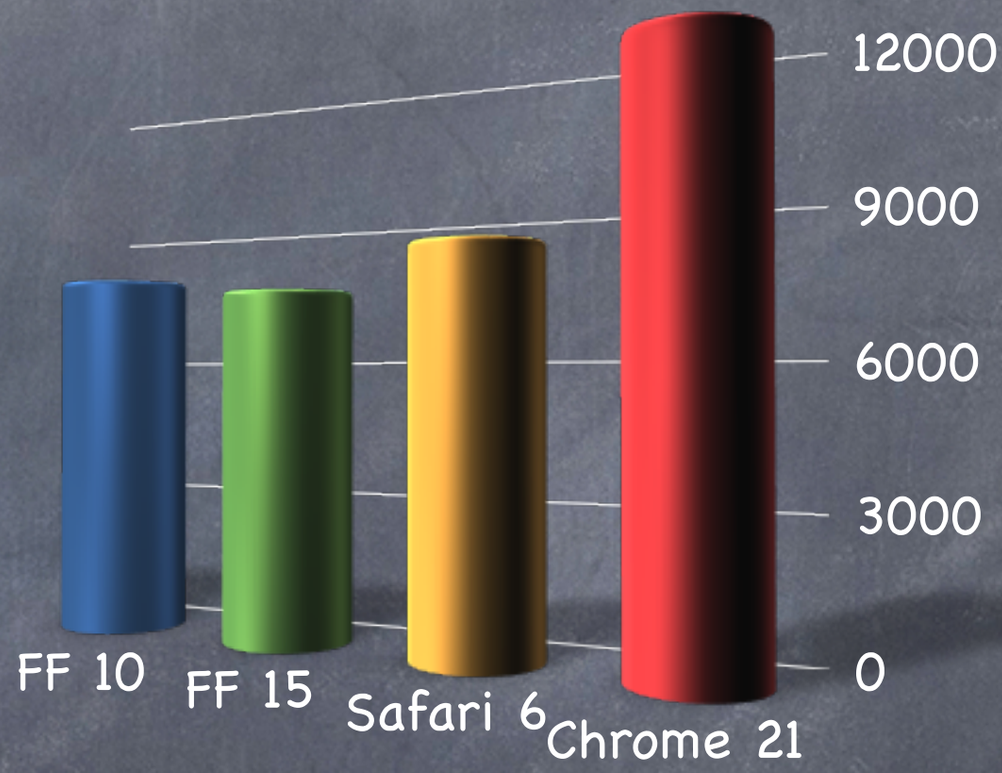
# Installation

1. www.nodejs.org
2. „install“
3. http-Server schreiben

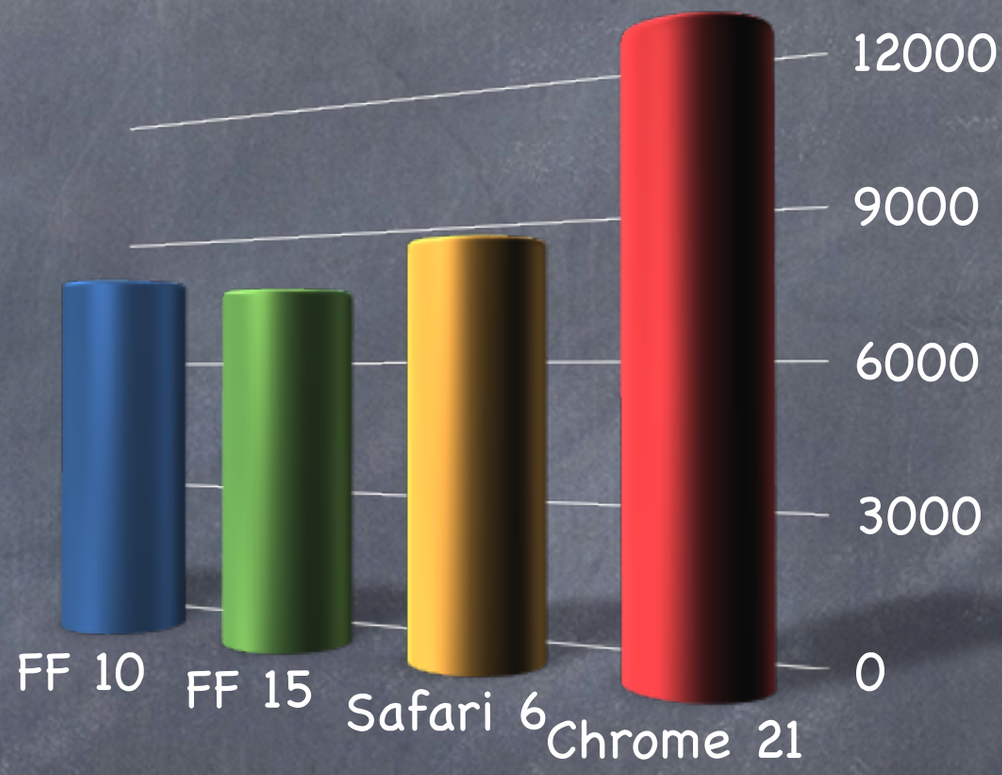
167 Sekunden bis zum „incoming request“

# V8 Engine

# V8 Engine

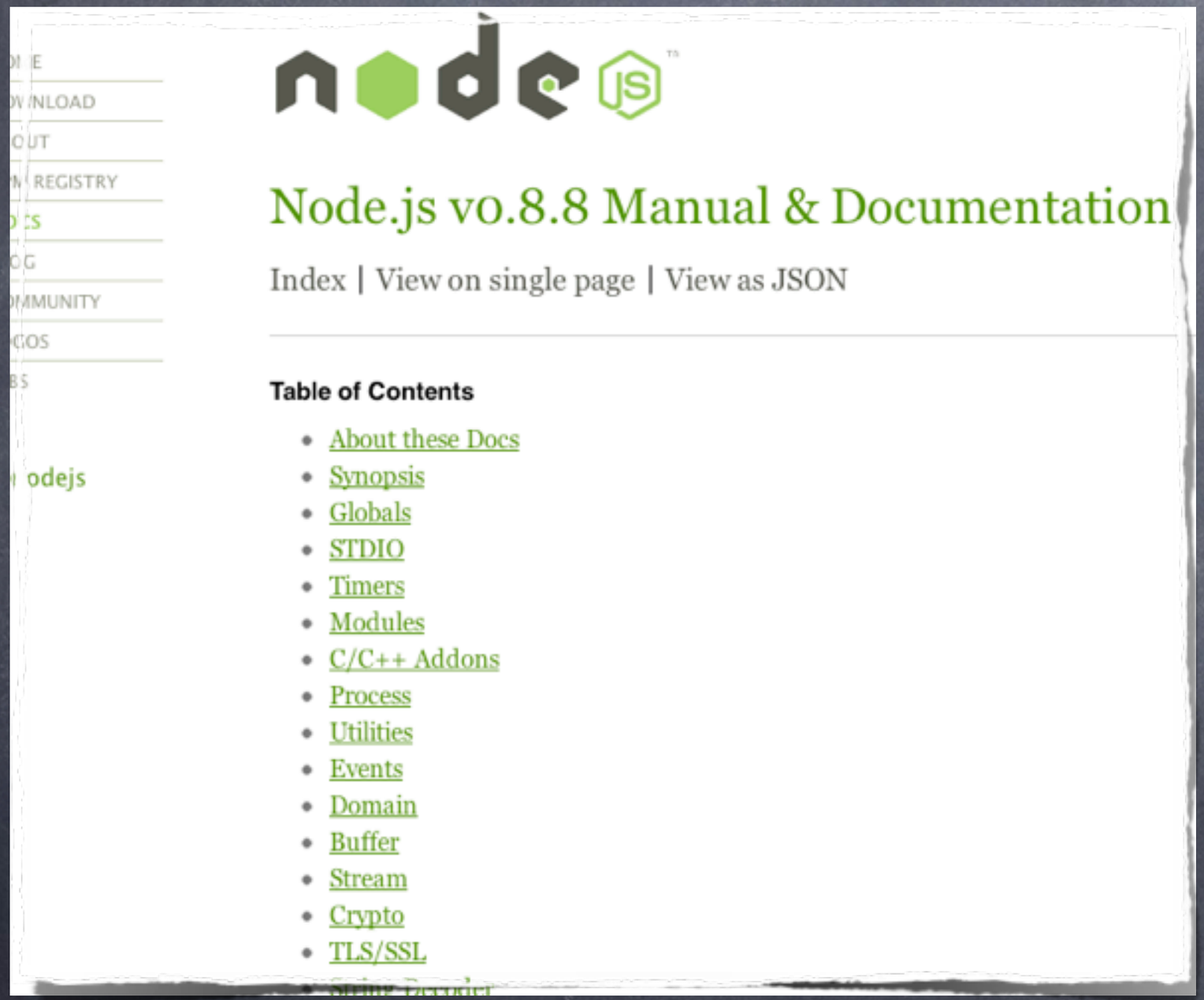



# V8 Engine

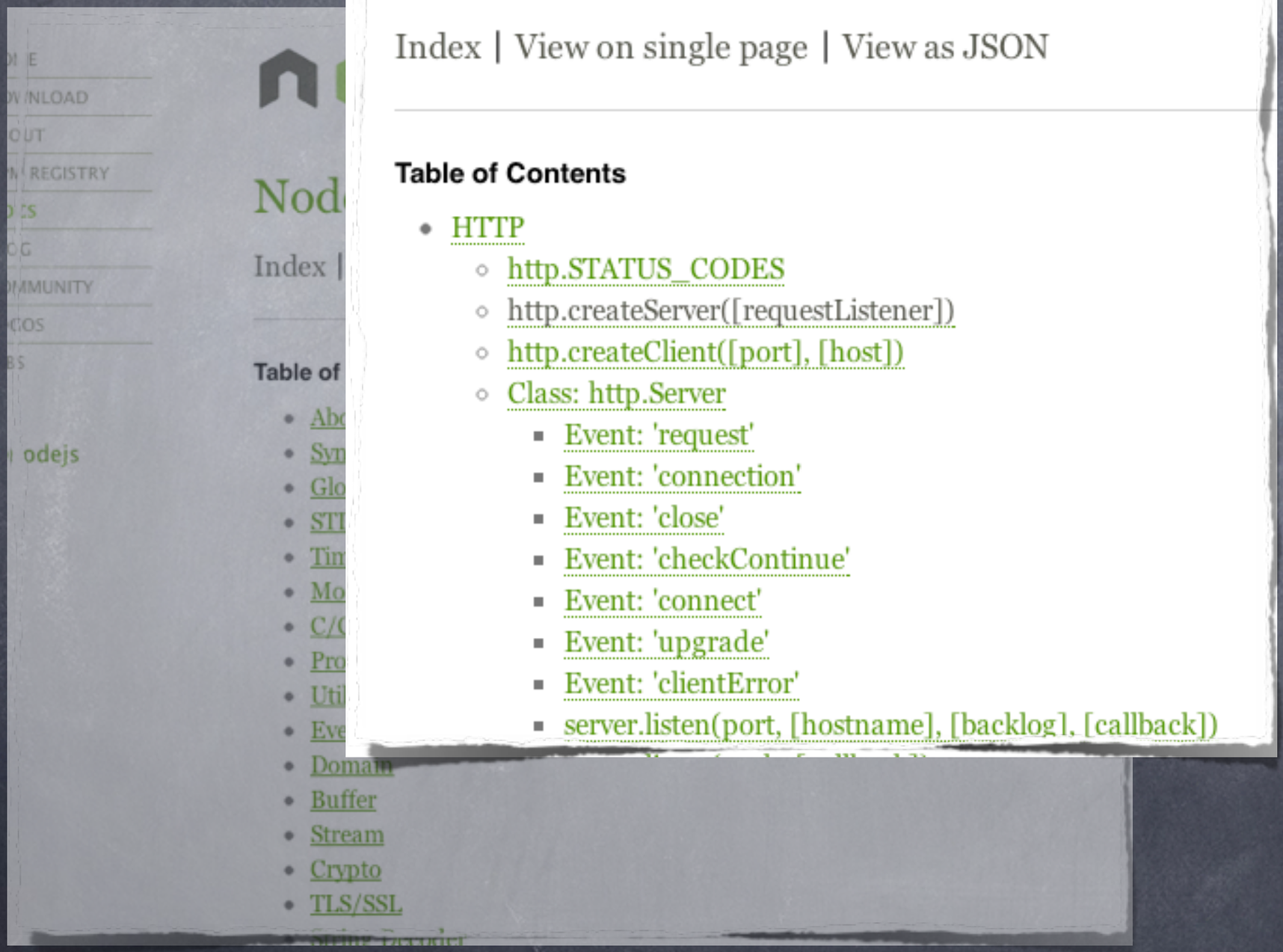




- hidden classes
- Caches, Statistiken
- zwei Compiler, kein Interpreter

# Docs


E
NLOAD
UT
REGISTRY
S
G
MMUNITY
OS
S
odejs
Node.js v0.8.8 Manual & Documentation
Index | View on single page | View as JSON
Table of Contents
About these Docs
Synopsis
Globals
STDIO
Timers
Modules
C/C++ Addons
Process
Utilities
Events
Domain
Buffer
Stream
Crypto
TLS/SSL

# Docs


Node.js v0.8.8 Manual & Documen
Index | View on single page | View as JSON
Table of Contents
HTTP
http.STATUS_CODES
http.createServer([requestListener])
http.createClient([port], [host])
Class: http.Server
Event: 'request'
Event: 'connection'
Event: 'close'
Event: 'checkContinue'
Event: 'connect'
Event: 'upgrade'
Event: 'clientError'
server.listen(port, [hostname], [backlog], [callback])
Table of
Domain
Buffer
Stream
Crypto
TLS/SSL

# Docs

Node.js v0.8.8 Manual & Documen

Index | View on single page | View as JSON

Table of Con

- HTTP
  - http
  - http
  - http
  - Clas

Node.js v0.8.8

Index | View on sing

Table of Contents

- About these Docs
- Synopsis
- Globals
- STDIO
- Timers
- Modules
- C/C++ Addons
- Process
- Utilities
- Events
- Domain
- Buffer
- Stream
- Crypto
- TLS/SSL

## http.STATUS_CODES

- Object

A collection of all the standard HTTP response status codes, and the short description of each. For exa
`http.STATUS_CODES[404] === 'Not Found'`.

## http.createServer([requestListener])

Returns a new web server object.

The `requestListener` is a function which is automatically added to the `'request'` event.

## http.createClient([port], [host])

This function is **deprecated**; please use http.request() instead. Constructs a new HTTP client. `port` a
refer to the server to be connected to.

## Class: http.Server

This is an EventEmitter with the following events:

## Event: 'request'

```
function (request, response) { }
```

# server.js

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

server = http.createServer(function(req, res) {
    console.log("incoming request");
});

server.listen(8888, function() {
    console.log("server started");
})
```

# NPM

Node Packaged Modules

| 3rd Party | Core | Eigene Module |
|---|---|---|
| ~15000 | ~30 | |
| express | http | |
| socket.io | crypto | |
| connect | vm | |
| ... | ... | |

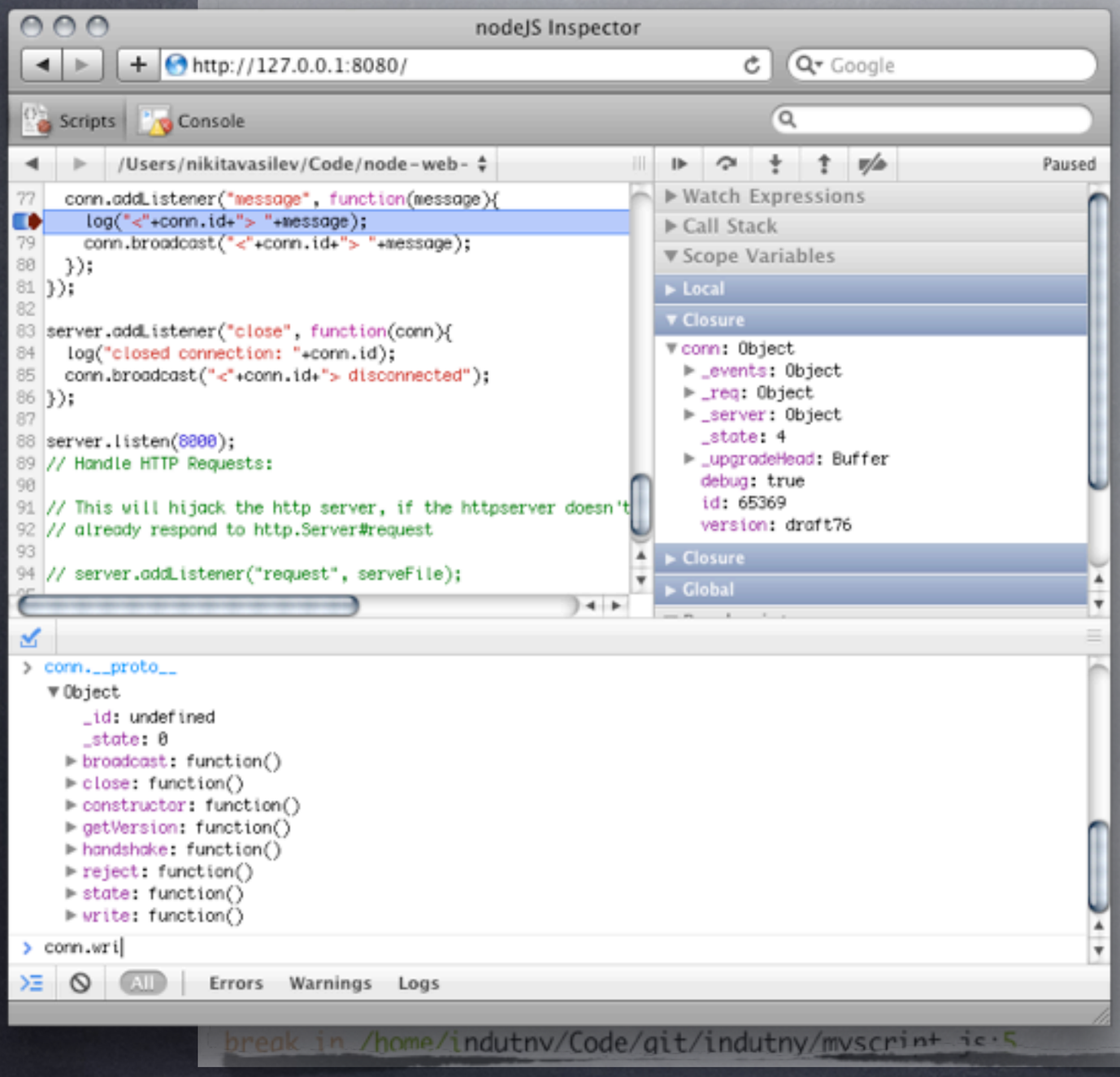
```
exports.mylog =
function(msg) {
   console.log(msg);
}
```

# Debugging

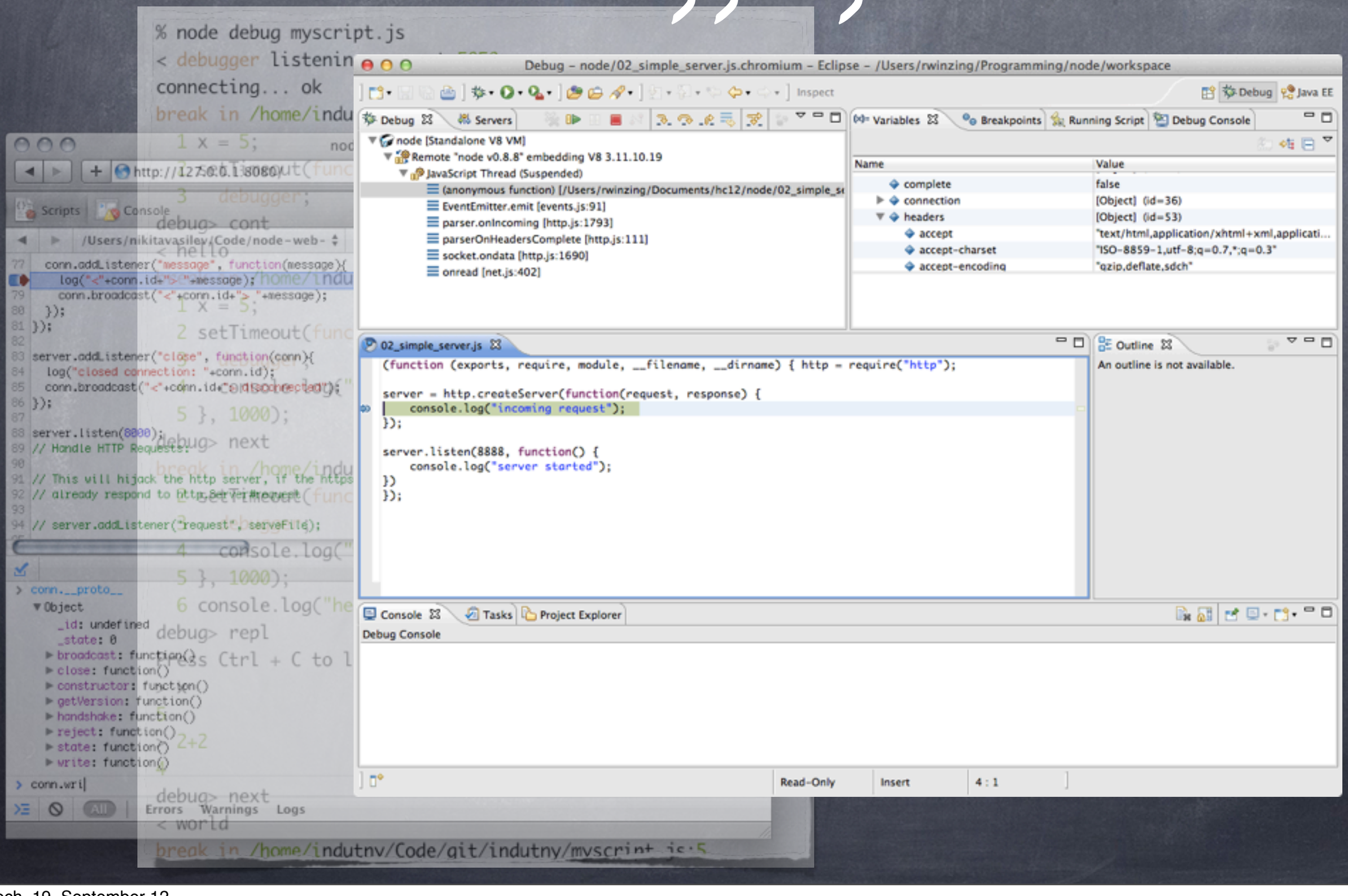
```
% node debug myscript.js
< debugger listening on port 5858
connecting... ok
break in /home/indutny/Code/git/indutny/myscript.js:1
  1 x = 5;
  2 setTimeout(function () {
  3   debugger;
debug> cont
< hello
break in /home/indutny/Code/git/indutny/myscript.js:3
  1 x = 5;
  2 setTimeout(function () {
  3   debugger;
  4   console.log("world");
  5 }, 1000);
debug> next
break in /home/indutny/Code/git/indutny/myscript.js:4
  2 setTimeout(function () {
  3   debugger;
  4   console.log("world");
  5 }, 1000);
  6 console.log("hello");
debug> repl
Press Ctrl + C to leave debug repl
> x
5
> 2+2
4
debug> next
< world
break in /home/indutny/Code/git/indutny/myscript.js:5
```

# Debugging

```
% node debug myscript.js
< debugger listening on port 5858
connecting... ok
break in /home/indutny/Code/git/indutny/myscript.js:1
```



# Debugging


% node debug myscript.js
< debugger listenin
connecting... ok
break in /home/indu
Debug – node/02_simple_server.js.chromium – Eclipse – /Users/rwinzing/Programming/node/workspace
Debug
Servers
node [Standalone V8 VM]
Remote "node v0.8.8" embedding V8 3.11.10.19
JavaScript Thread (Suspended)
(anonymous function) [/Users/rwinzing/Documents/hc12/node/02_simple_s
EventEmitter.emit [events.js:91]
parser.onIncoming [http.js:1793]
parserOnHeadersComplete [http.js:111]
socket.ondata [http.js:1690]
onread [net.js:402]
Variables
Breakpoints
Running Script
Debug Console
Name
Value
complete
false
connection
[Object] (id=36)
headers
[Object] (id=53)
accept
"text/html,application/xhtml+xml,applicati...
accept-charset
"ISO-8859-1,utf-8;q=0.7,*;q=0.3"
accept-encoding
"gzip,deflate,sdch"
02_simple_server.js
(function (exports, require, module, __filename, __dirname) { http = require("http");
server = http.createServer(function(request, response) {
console.log("incoming request");
});
server.listen(8888, function() {
console.log("server started");
})
});
Outline
An outline is not available.
Console
Tasks
Project Explorer
Debug Console
Read-Only
Insert
4 : 1
http://127.0.0.1:8080/
Scripts
Console
/Users/nikitavasilev/Code/node-web-
conn.addListener("message", function(message){
log("<"+conn.id+">"+message);
conn.broadcast("<"+conn.id+"> "+message);
});
});
server.addListener("close", function(conn){
log("closed connection: "+conn.id);
conn.broadcast("<"+conn.id+"> disconnected");
});
server.listen(8080);
// Handle HTTP Requests:
// This will hijack the http server, if the https
// already respond to http.ServerRequest
// server.addListener("request", serveFile);
conn.__proto__
Object
_id: undefined
_state: 0
broadcast: function()
close: function()
constructor: function()
getVersion: function()
handshake: function()
reject: function()
state: function()
write: function()
conn.wri
Errors
Warnings
Logs
debug> cont
< hello
debug> next
debug> repl
Press Ctrl + C to l
2+2
debug> next
< world
break in /home/indutnv/Code/git/indutnv/myscript.js:5

# chat.js

node app

postMessage(nick, msg)

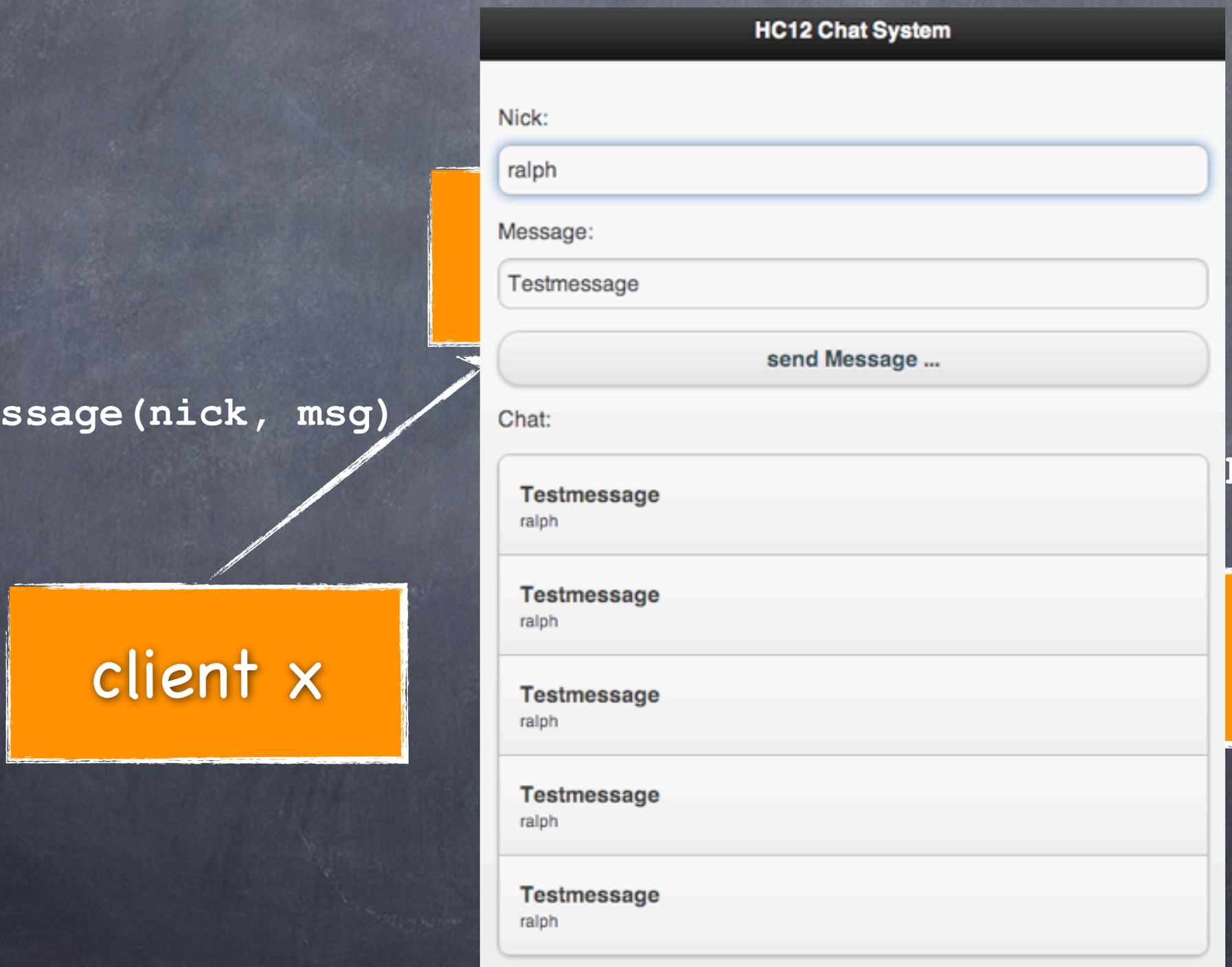
client x

(push)
newMessage(nick, msg)

client z

client y

# chat.js


HC12 Chat System
Nick:
ralph
Message:
Testmessage
send Message ...
Chat:
Testmessage
ralph
Testmessage
ralph
Testmessage
ralph
Testmessage
ralph
Testmessage
ralph
postMessage(nick, msg)
client x
k, msg)

# chat.js

<code/>

# chat.js

192.168.129.218:8080/chat.hmtl

# Danke!

ralph.winzinger@senacor.com

@rwinz on twitter

# Vielen Dank!

Ralph Winzinger

Senacor Technologies AG