WILLKOMMEN ZU

NODE JS

EINFÜHRUNG IN NODE.JS

- Über den Autor Ryan Dahl, die Javascript Engine V8 und die Idee für serverseitiges Javascript

„It will be very fun.“

–RYAN DAHL

Node.js was created by Ryan Dahl starting in 2009. Its development and maintenance is sponsored by Joyent.

–WIKIPEDIA

Dahl was inspired to create Node.js after seeing a file upload progress bar on Flickr. The browser did not know how much of the file had been uploaded and had to query the web server. Dahl wanted an easier way.

–WIKIPEDIA

Ryan Dahl, the creator of Node.js, originally had the goal in mind of creating web sites with push capabilities as seen in web applications like Gmail. After trying solutions in several other programming languages he chose JavaScript because of the lack of an existing I/O API. This allowed him to define a convention of non-blocking, event-driven I/O.

–WIKIPEDIA

„Node.js ist eine serverseitige Plattform zum Betrieb von Netzwerkanwendungen. Insbesondere lassen sich Webserver damit realisieren. Node.js basiert auf der JavaScript-Laufzeitumgebung „V8“, die ursprünglich für den Chrome-Browser entwickelt wurde und bietet daher eine ressourcensparende Architektur, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht.“

— WIKIPEDIA

Node.js was first published by Dahl in 2011 and could only run on Linux. NPM, a package manager for Node.js libraries, was introduced the same year. In June 2011, Microsoft partnered with Joyent to help create a native Windows version of Node.js. The first Node.js build to support Windows was released in July.

–WIKIPEDIA

On January 30, 2012 Dahl stepped aside, promoting coworker and NPM creator Isaac Schlueter to the gatekeeper position.

...

On January 15, 2014 Schlueter announced he was making NPM his main focus and Timothy J Fontaine would be Node.js new project lead.

–WIKIPEDIA

JAVASCRIPT AUF DEM SERVER ZU VERWENDEN, IST GUT, WEIL

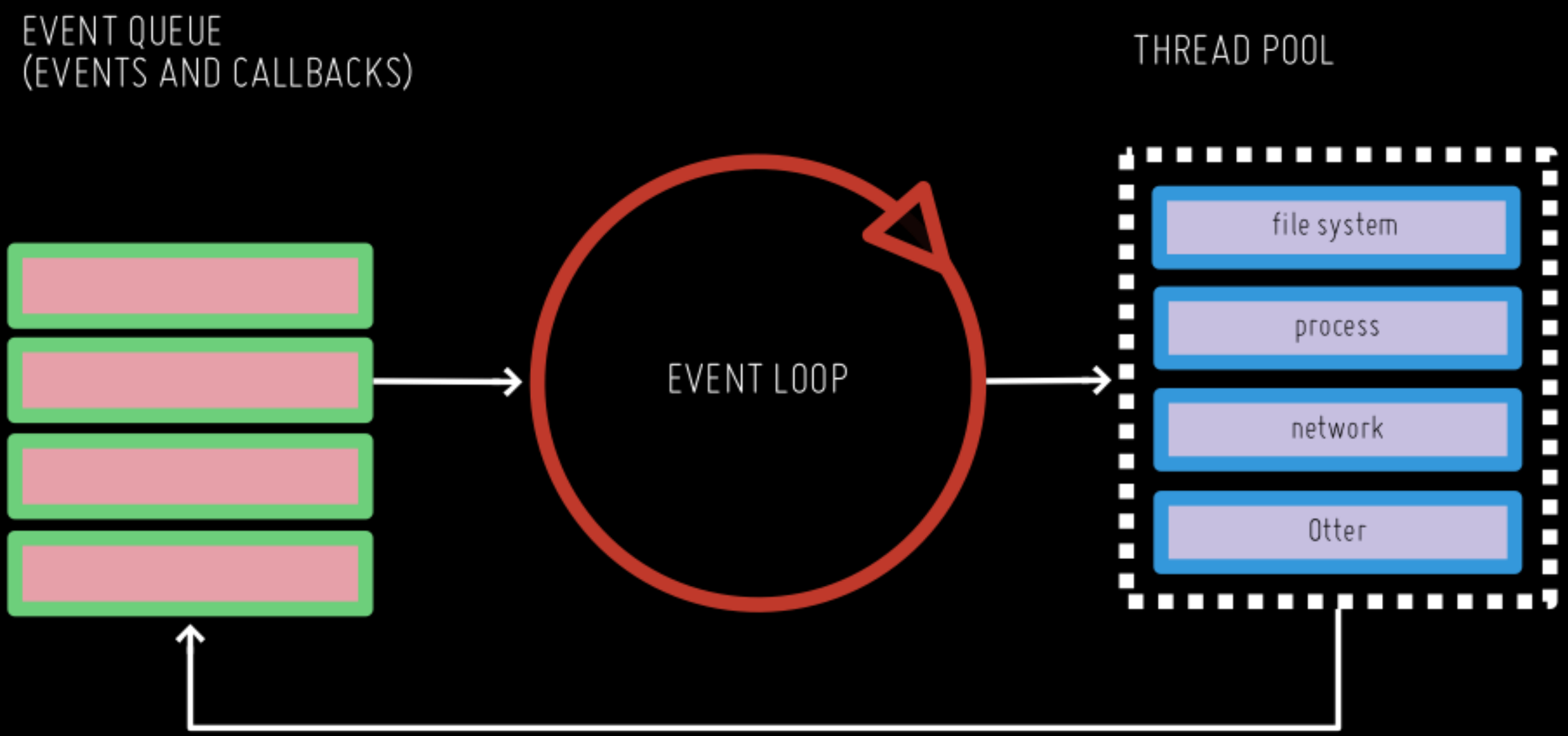
- Client und Server technisch auf derselben Basis stehen.
- Format- und Kompatibilitätsprobleme zwischen unterschiedlichen Sprachen, zum Beispiel Javascript und PHP wegfallen.
- JavaScript alle Aufgaben eines Web-, Socket- oder Datenservers übernehmen kann.

JAVASCRIPT AUF DEM SERVER ZU VERWENDEN, IST GUT, WEIL

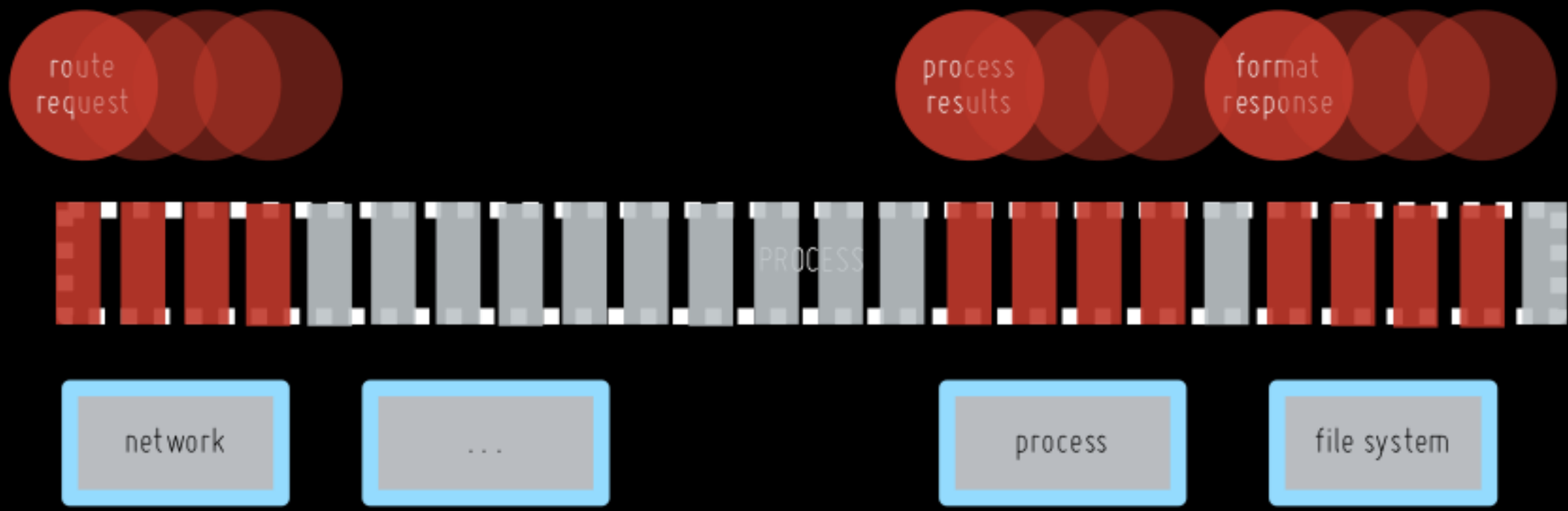
- Server und Client zu einem Ganzen verschmelzen.

NON BLOCKING I/O, EVENT LOOPS, SINGLE THREADS

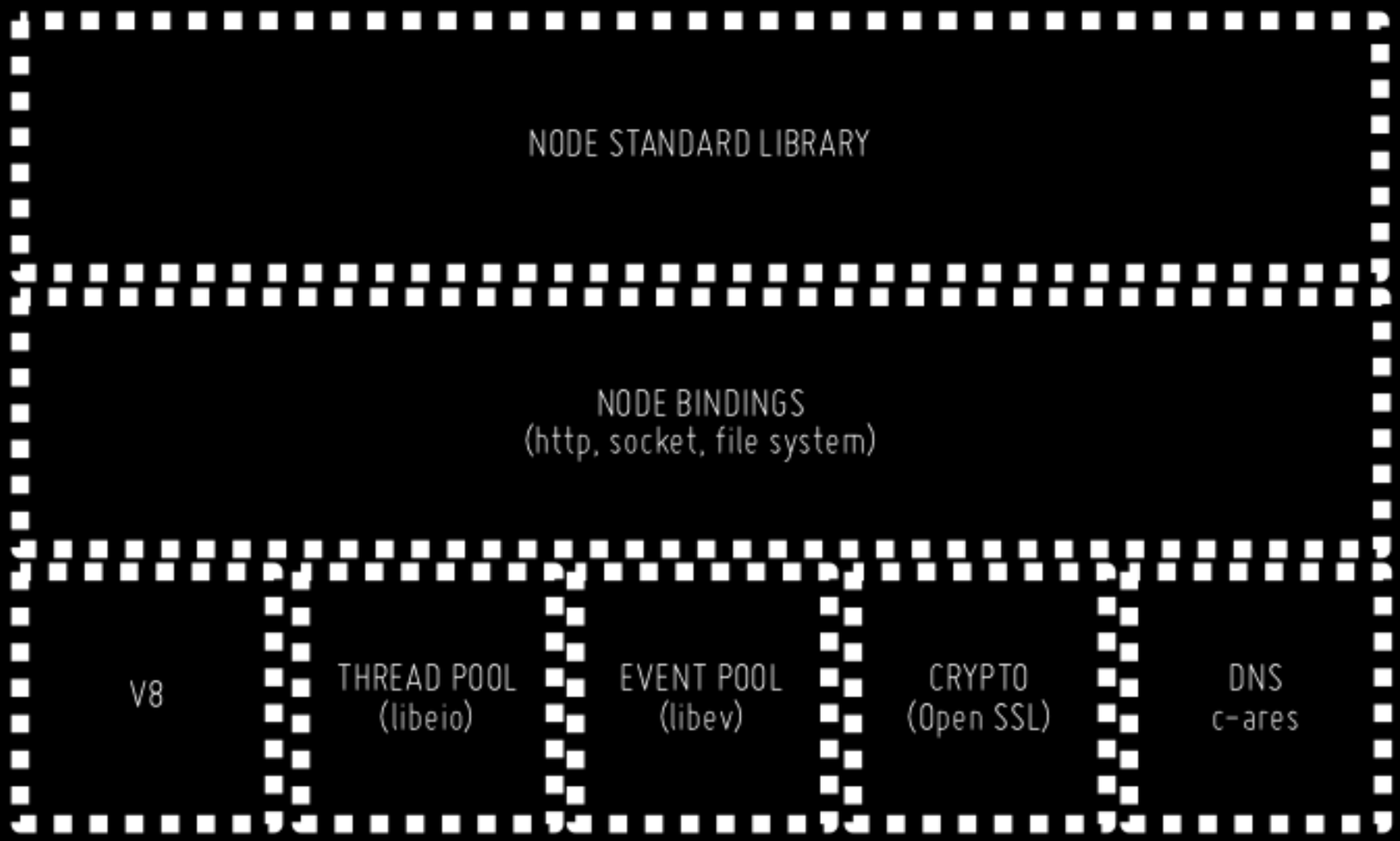
JAVASCRIPT THREAD VERHALTEN UND EVENTLOOP


EVENT QUEUE
(EVENTS AND CALLBACKS)
EVENT LOOP
THREAD POOL
file system
process
network
Otter

SCALING WITH AN EVENT LOOP



NODE TIERS AND BINDINGS



FROM THE SCRATCH

INSTALLATION UND KONFIGURATION

HOME | ABOUT | DOWNLOADS | DOCS | FOUNDATION | GET INVOLVED | SECURITY | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Download for Windows, Linux, Mac OS

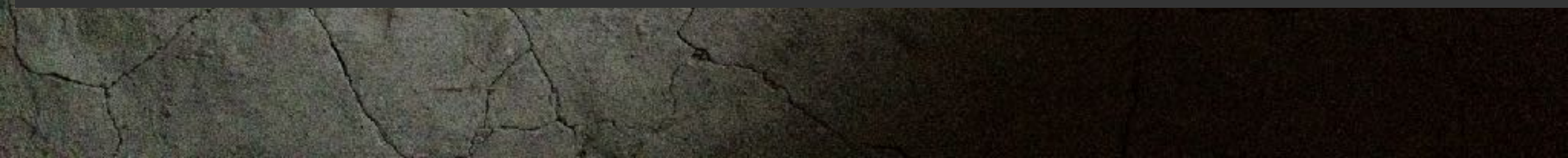
v8.9.1 LTS
Recommended For Most Users

Other Downloads | Changelog | API Docs

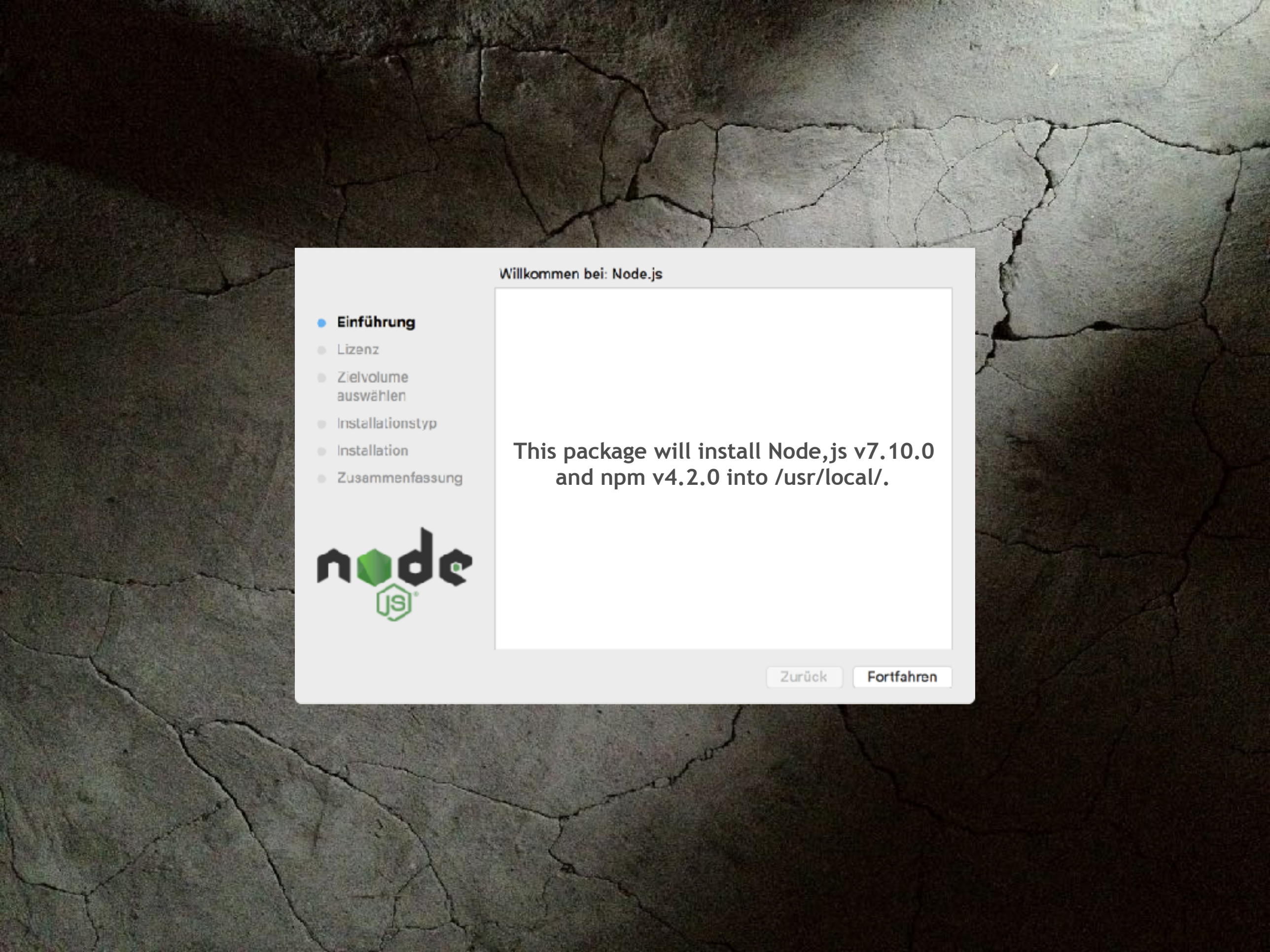
v9.1.0 Current
Latest Features

Other Downloads | Changelog | API Docs

Or have a look at the LTS schedule.



- Installer für Windows und Mac OS
- Erweiterbar über Node Package Manager
- geschrieben in C++, individuell kompilierbar


Willkommen bei: Node.js
Einführung
Lizenz
Zielvolume auswählen
Installationstyp
Installation
Zusammenfassung
This package will install Node,js v7.10.0 and npm v4.2.0 into /usr/local/.
Zurück
Fortfahren

Installation erfolgreich abgeschlossen

- Einführung
- Lizenz
- Zielvolume auswählen
- Installationstyp
- Installation
- **Zusammenfassung**

Node.js was installed at

/usr/local/bin/node

npm was installed at

/usr/local/bin/npm

Make sure that /usr/local/bin is in your $PATH.

Zurück | Schließen

NODE ABFRAGEN

```
$ node -v
v9.1.0
```

```
$ npm -v
V5.5.1
```

NODEJS VIA TERMINAL AKTUALISIEREN

```
[sudo npm cache clean -f]
[sudo npm install -g n]
```

```
sudo n stable

sudo n latest

sudo n 7.8.0
```

Nutmeg Plundering Muse

npm Enterprise features pricing documentation support

npm

find packages

sign up or log in

Build amazing things

npm is the package manager for JavaScript. Find, share, and reuse packages of code from hundreds of thousands of developers — and assemble them in powerful new ways.

Get started

Empower your team with private packages

- Securely manage private code with the same workflow as open source projects

Packages people 'npm install' a lot

browserify
browser-side require() the nod...
14.0.0 published a week ago by **fenxs**

grunt-cli
The grunt command line interf...
1.2.0 published 10 months ago by **vladikoff**

bower
The browser package manager
1.8.0 published 3 months ago by **sheerun**

gulp
The streaming build system
3.9.1 published 12 months ago by **phated**

grunt
The JavaScript Task Runner
1.0.1 published 10 months ago by **shama**

express
Fast, unopinionated, minimalis...
4.14.1 published 6 days ago by **dougwilson**

npm
a package manager for JavaScr...
4.1.2 published 3 weeks ago by **iarna**

cordova
Cordova command line interfa...
6.5.0 published 2 weeks ago by **stevegill**

forever
A simple CLI tool for ensuring t...
0.15.3 published 3 months ago by **indexzero**

Natively Pronounced Mandarin

npm Enterprise features pricing documentation support

npm

find packages

Greetings, zenbox

npm is the package manager for javascript.

411.246 total packages

285.347.463 downloads in the last day

1.671.943.125 downloads in the last week

7.118.799.986 downloads in the last month

packages people 'npm install' a lot

browserify
browser-side require() the node way

14.0.0 published a week ago by feross

express
Fast, unopinionated, minimalist web framework...

4.14.1 published 7 days ago by dougwilson

pm2
Production process manager for Node.JS appl...

2.3.0 published 2 weeks ago by tknew

grunt-cli
The grunt command line interface

1.2.0 published 10 months ago by vladikoff

npm
a package manager for JavaScript

4.1.2 published 3 weeks ago by iarna

karma
Spectacular Test Runner for JavaScript.

1.4.1 published 6 days ago by dignifiedquire

bower
The browser package manager

1.8.0 published 3 months ago by sheerun

cordova
Cordova command line interface tool

6.5.0 published 2 weeks ago by stevegill

coffee-script
Unfancy JavaScript

1.12.3 published 2 weeks ago by lydell

gulp
The streaming build system

3.9.1 published 12 months ago by phated

forever
A simple CLI tool for ensuring that a given nod...

0.15.3 published 3 months ago by indexzero

statsd
Network daemon for the collection and aggreg...

0.8.0 published 9 months ago by pkhazzard

HALLO WELT

3 VERSIONEN

HALLO WELT - KONSOLENVERSION

```
$ node
> function hw () { console.log("hello world"); }
> hw()
hello world


(To exit, press ^C again or type .exit)
```

HALLO WELT - PROGRAMMVERSION

```
// hello world.js
// – – – – – – – – –
function hw () { console.log("hello world"); }
hw();
// – – – – – – – – –

$ node hello-world
hello world
$
```

SERVER.JS

```
let http = require('http'),
    host = 'http://localhost',
    port = 3000,
    server = null;

server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello world</h1>!');
});

server.listen(port);

console.log('a simple web service on ' + host + ':' + port);
```

HALLO WELT - KONSOLENVERSION

```
$ node server
```

Um Ports unterhalb der 1024 zu verwenden, muss das Node.js-Programm mit Root-Rechten ausgeführt werden.

–GUT ZU WISSEN.

NODE MONITORING

nodemon reload, automatically.

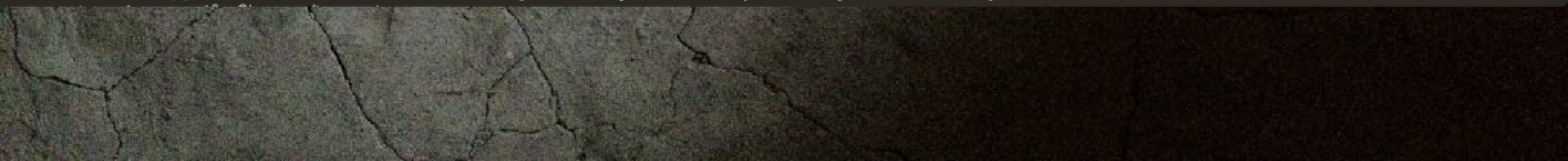
Nodemon is a utility that will monitor for any changes in your source and automatically restart your server. Perfect for development. Install it using **npm**.

Just use `nodemon` instead of `node` to run your code, and now your process will automatically restart when your code changes. To install, get **node.js**, then from your terminal run:

```
npm install -g nodemon
```

Features

- Automatic restarting of application.
- Detects default file extension to monitor.
- Default support for node & coffeescript, but easy to run any executable (such as python, make, etc).



NODEMON GLOBAL INSTALLIEREN

```
$ sudo npm install -g nodemon
Password:

/usr/local/bin/nodemon -> /usr/local/lib/node_modules/nodemon/bin/nodemon.js

> fsevents@1.0.17 install /usr/local/lib/node_modules/nodemon/node_modules/fsevents
> node-pre-gyp install --fallback-to-build

[fsevents] Success: "/usr/local/lib/node_modules/nodemon/node_modules/fsevents/lib/binding/Release/
node-v51-darwin-x64/fse.node" is installed via remote
/usr/local/lib
└─┬ nodemon@1.11.0
  ├─┬ chokidar@1.6.1
  │ ├─┬ anymatch@1.3.0
  │ │ ├── arrify@1.0.1
  │ │ └─┬ micromatch@2.3.11
  │ │   ├─┬ arr-diff@2.0.0
  │ │   │ └── arr-flatten@1.0.1
  │ │   ├── array-unique@0.2.1
  │ │   ├─┬ braces@1.8.5
  │ │   │ ├─┬ expand-range@1.8.2
  │ │   │ │ └─┬ fill-range@2.2.3
  │ │   │ │   ├── is-number@2.1.0
  │ │   │ │   ├── isobject@2.1.0
  │ │   │ │   ├── randomatic@1.1.6
  │ │   │ │   └── repeat-string@1.6.1
  │ │   │ ├── preserve@0.2.0
  │ │   │ └── repeat-element@1.1.2
. . .
```

AUTO RESTART MIT NODEMON

```
$ nodemon hello-world.js

$ nodemon --debug ./server.js 80
```

NODEMON KONFIGURIEREN

```
{
  "name": "nodemon",
  "homepage": "http://nodemon.io",
  ...
  ... other standard package.json values
  ...
  "nodemonConfig": {
    "ignore": ["test/*", "docs/*"],
    "delay": "2500"
  }
}
```

ABOUT NODEMON

```
https://github.com/remy/nodemon
```

AUSGABEN ÜBER DIE KONSOLE

KONSOLENBEFEHLE

```
console.time('my time measuring');

let value  = 'a primitive value',
    object = {
        prename  : 'Michael',
        lastname : 'Reichart'
    };

console.log(value);
console.dir(object);

console.timeEnd('my time measuring')
```

KONSOLENBEFEHLE

```
console.assert(value[, message][, ...args])
console.dir(obj[, options])
console.error([data][, ...args])
console.info([data][, ...args])
console.log([data][, ...args])
console.time(label)
console.timeEnd(label)
console.trace(message[, ...args])
console.warn([data][, ...args])
```

DEBUGGING

DEBUGGING MIT DEN CHROME DEVTOOLS

DER NODEJS V8 INSPECTOR

- Die Chrome Extension "nodejs V8 inspector" in einer aktuellen Chromeversion installieren.

- `$ node --inspect my-file.js`

- startet einen Port 9229, über den die Chrome Devtools Fehlermeldungen anzeigen und der Code fehlerbehandelt werden kann.

```
Debugger listening on port 9229.

Warning: This is an experimental feature and could change at
any time.

To start debugging, open the following URL in Chrome:

chrome-devtools://devtools/bundled/inspector.html?
experiments=true&v8only=true&ws=127.0.0.1:9229/2fb748b6-67ef-4
56c-bb72-0d20bfe4ecde
```

MANUAL DEBUGGING

```
$ node debug hello-world.js

< Debugger listening on 127.0.0.1:5858
connecting to 127.0.0.1:5858 ... ok
break in hello-world.js:2
  1 // - - - - - - - - - - -
> 2 console.time('processed time');
  3 // - - - - - - - - - - -
  4
debug>
```

SERVER.JS

```
let http = require('http'),
    host = 'http://localhost',
    port = 3000,
    server = null;

server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    debugger;
    response.end('<h1>Hello world</h1>!');
});

server.listen(port);

console.log('a simple web service on ' + host + ':' + port);
```

STEPPING

```
cont, c – Continue execution
next, n – Step next
step, s – Step in
out,  o – Step out
pause   – Pause running code
```

BREAKPOINTS

```
setBreakpoint(),
sb()            - Set breakpoint on current line

setBreakpoint(line),
sb(line)        - Set breakpoint on specific line

setBreakpoint('fn()'),
sb(...)         - Set breakpoint on a first statement
                  in functions body

setBreakpoint('script.js', 1),
sb(...)         - Set breakpoint on first line of script.js

clearBreakpoint('script.js', 1),
cb(...)         - Clear breakpoint in script.js on line 1
```

EREIGNISSE VERARBEITEN

EVENTHANDLER UND -LISTENER

- Viele Objekte senden in node Events aus: ein `net.server` sendet einen Event, wenn sich ein Client mit ihm verbindet. Ein `fs.readStream` Many objects in Node emit events: a net.Server emits an event each time a sendet ein Event, wenn die Datei geöffnet ist.

- Jedes Objekt, das Events sendet ist eine Instance des events.EventEmitter Konstruktors. Diese kann über `require("events");` eingebunden werden.

- Funktionen können an Objekte gebunden werden, wenn diese einen Event gesendet haben: Diese Funktionen werden Eventhandler genannt.

- In Eventhandlern verweist `this` auf das Event - aussendende Objekt.

EVENTLISTENER SETZEN

```
emitter.addListener(event, listener)
emitter.on(event, listener)
```

```
server.on('connection', function (stream) {
        console.log('someone connected!');
});
```

EINMAL - EVENT

```
emitter.once(event, listener)
```

```
server.once('connection', function (stream) {
        console.log('Ah, we have our first user!');
});
```

LÖSCHEN VON EVENTLISTENERN

```
emitter.removeListener(event, listener)
```

```
var callback = function(stream) {
 console.log('someone connected!');
};

server.on('connection', callback);
// ...
server.removeListener('connection', callback);
```

WEITERES …

```
emitter.removeAllListeners([event])
```

```
emitter.setMaxListeners(n) // 0 == unbegrenzt
```

EIN MODUL SCHREIBEN

MODUL SCOPES

- Eine Javascript Datei bildet einen eigenen Scope.
- Auf ein IIFE Pattern kann verzichtet werden.
- require() und exports verbinden Prozess und Module.

EIN NODE MODUL SCHREIBEN

```
let
    area = null,
    circumference = null;

area = function(radius) {
    'use strict';
    return Math.PI * radius * radius;
};

circumference = function(radius) {
    'use strict';
    return Math.PI * 2 * radius;
};

exports.area = area;
exports.circumference = circumference;
```

EIN MODUL VERWENDEN

```
let
    myModule = null,
    radius = 5;

myModule = require('./module.js');

console.log(myModule.circumference(radius[i]));
console.log(myModule.area(radius[i]));
```

MODULE VERWALTEN

PACKAGE.JSON

```
{
  "name" : "my-express-app",   // < 215 Zeichen, keine Großbuchstaben, url-safe
  "version":"0.0.1",
  "description" : "Maecenas sed diam eget risus varius blandit.",
  "license": "MIT",
  "repository": {
    "type" : "git",
    "url"  : "git+https://github.com/zenbox/workshop.git"
  },
  "contributors": [
    {
      "name": "Michael Reichart",
      "email": "michael@zenbox.de"
    }
  ],
  "dependencies" : {
    "express" : "4.15.2",
    "ejs" : "2.5.6",
    "jade" : "1.11.0",
    "body-parser" : "1.17.1",
    "cookie-parser": "1.4.3",
    "socket.io": "1.7.3"
  }
}
```

EIN MODUL EINBINDEN

```
let
  http = require('http'),
  fs = require(fs),
  express = require('express');
```

MODUL-VERSIONIERUNG

```
{
  "dependencies" : {
    "modulename" : " v1.23.3",
    "modulename" : "  1.23.3",

    "modulename" : " >1.23.1",
    "modulename" : ">=1.23.2",
    "modulename" : "<=1.23.3",
    "modulename" : " <1.23.4",

    "modulename" : " >1.23.1 <1.23.4",
    "modulename" : " 1.22.9 || >1.23.1 <1.23.4",

  }
}
```

MODUL-VERSIONIERUNG

```
{
  "dependencies" : {
    "modulename" : "1.23.1 - 1.23.5",    // inclusive set

    "modulename" : "1.x",
    "modulename" : "1.2.*",
    "modulename" : "~1.2",               // >=1.2.0 <1.3.0
    "modulename" : "~1.2.3",             // >=1.2.3 <1.3.0

    "modulename" : "^1.2.3",             // >=1.2.3 <2.0.0
    "modulename" : "^1.2.x",             // >=1.2.0 <2.0.0

  }
}
```

„ https://docs.npmjs.com/misc/semver “

DER BUFFER

DIE UTILITIES KLASSE

```
util.debuglog(section)
util.deprecate(function, string)
util.format(format[, ...args])
util.inherits(constructor, superConstructor)
util.inspect(object[, options])
```

```
%s - String.
%d - Number (both integer and float).
%j - JSON. Replaced with the string '[Circular]' if the argument contains circular references.
%% - single percent sign ('%'). This does not consume an argument.

util.format('%s:%s', 'foo', 'bar', 'baz'); // 'foo:bar baz'
```

```
dataString = util.format(
    '%s-%s-%s %s:%s:%s Lorem ipsum dolor sit.\n',
     now.getFullYear(),
    (now.getMonth() + 1),
     now.getDate(),
     now.getHours(),
     now.getMinutes(),
     now.getSeconds()
    );
```

DATEISYSTEM

EXPRESS

EXPRESS BASIS APPLIKATION

```
var express = require('express');
var app = express();

app.get('/', function(request, response){
  response.send('hello world');
});

app.listen(3000);
```

DIE ERSTEN APP-METHODEN

```
app.get, app.post, app.put, ...,
app.all,
app.param     - Routing HTTP requests.
app.route     - Configuring middleware.
app.render    - Rendering HTML views.
app.engine    - Registering a template engine.
```

ROUTES

```
app.get('/', callback);

Die Pfadangabe ('/') kann
- ein Stringausdruck,
- ein Pfadmuster,
- ein regulärer Ausdruck,
- oder ein Array mit einer Kombination daraus sein.
```

DAS REQUEST OBJEKT

- bildet den HTTP Request ab.
- Es enthält Eigenschaften wie den Querystring, übergebene Parameter, den Request-Body, die HTTP-Headers und so weiter.
- http://expressjs.com/de/4x/api.html#req

DAS REQUEST OBJEKT

```
app.get('/user/:id', function(request, response) {
  res.send('user ' + request.params.id);
});

console.log('The views directory is ' +
request.app.get('views'));

console.log(request.baseUrl);

console.log(request.cookies.name);

console.log(request.hostname);
console.log(request.ip);
console.log(request.method);
console.log(request.path);
console.log(request.query);
console.log(request.route);
```

REQUEST BODY

```
var app        = require('express')();
var bodyParser = require('body-parser');
var multer     = require('multer'); // v1.0.5
var upload     = multer(); // for parsing multipart/form-data

// for parsing application/json
app.use(bodyParser.json());

// for parsing application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: true }));

app.post('/profile', upload.array(), function (request,
response, next) {
  console.log(request.body);
  response.json(request.body);
});
```

DAS RESPONSE OBJEKT

- Bildet die HTTP Antwort, die Express bei einer Anfrage formuliert und abschickt.

- Es besitzt ausser einigen Eigenschaften vor allem die Methode, um eine Antwort zu bilden.

DAS RESPONSE OBJEKT

```
response.append('Link', ['<http://localhost/>', '<http://localhost:3000/
>']);
response.append('Set-Cookie', 'foo=bar; Path=/; HttpOnly');
response.append('Warning', '199 Miscellaneous warning');

response.attachment('path/to/logo.png');

response.cookie('name', 'tobi', { domain: '.example.com', path: '/admin',
secure: true });

response.download('/report-12345.pdf');

res.send(new Buffer('whoop'));
res.send({ some: 'json' });
res.send('<p>some html</p>');
res.status(404).send('Sorry, we cannot find that!');
res.status(500).send({ error: 'something blew up' });

response.end();
response.status(404).end();
```

```
response.json(null);
response.json({ user: 'tobi' });
response.status(500).json({ error: 'message' });

response.jsonp(null);
response.jsonp({ user: 'tobi' });
response.status(500).jsonp({ error: 'message' });
```

RESPONSE FORMATE

```
res.format({
  'text/plain': function(){
    res.send('hey');
  },

  'text/html': function(){
    res.send('<p>hey</p>');
  },

  'application/json': function(){
    res.send({ message: 'hey' });
  },

  'default': function() {
    // log the request and respond with 406
    res.status(406).send('Not Acceptable');
  }
});
```

ECHTZEITKOMMUNIKATION ZWISCHEN CLIENT UND SERVER

WEBSOCKETS

„Mit AJAX Requests kann ich über dem Browser keine Modelleisenbahn steuern.“

– JAN HICKSON

HTTP IST STATUSLOS

- -> Request, Response, Ende
- Echtzeit ist nicht möglich, da Client und Server bei jedem Request erneut verhandeln, bevor Daten übertragen werden.

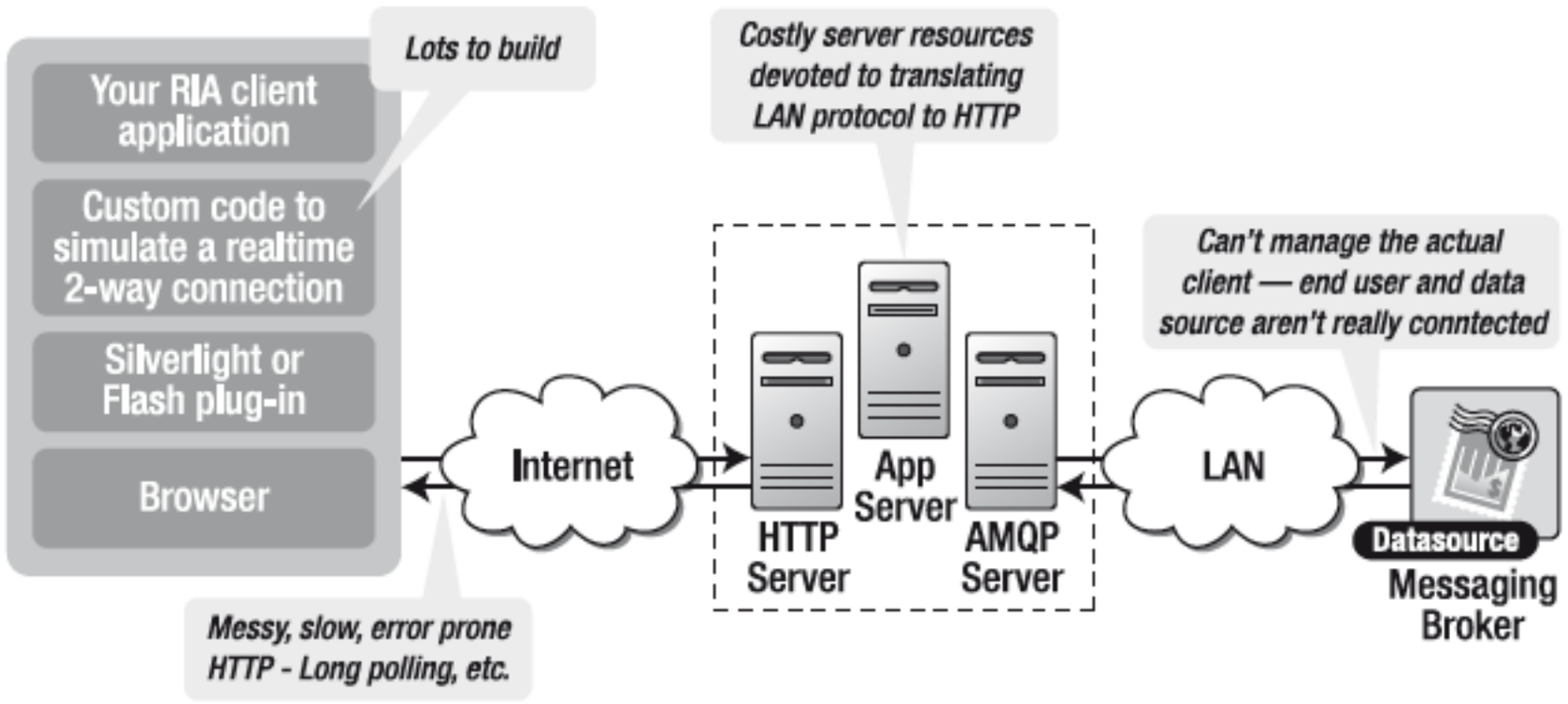
ECHTZEITVERBINDUNGEN?

- Seiten aktualisieren
- Polling
- Longpolling

- Polling und Longpolling setzen dabei in regelmäßigen Intervallen Requests an den Server ab, um neue Informationen zu erhalten.

- Beim Longpolling hält der Server die Verbindung eine Zeit lang offen.

- Keine echte Synchronisation mit severseitigen Informationsupdates



WEBSOCKETS

- ein Protokoll, das eine persistente Verbindung zwischen Browser und Webserver offenhält.

- ws:// - Websocket Protokoll
  wss:// - Websocket Secure Protokoll

BIDIREKTIONAL & FULLDUPLEX

- Über diese Verbindung kann in beiden Richtungen (Client <--> Server) kommuniziert werden.
- Aufgrund des binären Protokolls (ws:/wss:) hat es sehr wenig Overhead.

HTTP INITIALISIERUNG

- Der initiale Verbindungsaufbau einer Websocket-Verbindung läuft über HTTP (oder HTTPS),
- Ein Upgrade-Header teilt dem Server mit, dass auf das Websocket-Protokoll „upgegradet“ werden soll.

```
// From client to server:
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13

// From server to client:
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

SERVER PROXY

- Serverseitig wird ein Socketproxy installiert, zum Beispiel in node.js.

- nodejs beherrscht HTTP und Websockets und kann einen Upgrade-Headers verarbeiten.

WEITERE SOCKET SERVER

- **Jetty**
  HTTP-Server und Servlet-Container, der WebSockets seit Version 7.0.1 unterstützt.

- **phpwebsocket**
  einer der ersten WebSocket-Server in PHP

- **jWebSocket**
  High-Speed Kommunikationsserver inkl. Web, Java und Mobile Clients, Open Source

- **xLightweb**
  HTTP-Bibliothek inkl. HttpClient und HttpServer, welche WebSocket und ServerSentEvent unterstützt

SOCKET TOOLS

- **Tomcat**
  ab Version 7.0.27

- **PyWebsocket**
  Python Websocket Server (Apache Modul oder Standalone)

- **Node.js**
  Serverseitiges Javascript, mit dem ein Webserver geschrieben werden kann

- **Socket.io**
  Ein Framework, mit dem realtime apps für jeden Browser und jedes mobile Gerät möglich sind.

HTML5 SOCKETS BROWSER

- Chrome 4+
- Firefox 4+
- Safari 5+
- IE 10+
- Opera 11.5+

WAS BRINGEN WEBSOCKETS?

VORTEILE DER WEBSOCKETS GEGENÜBER HTTP REQUESTS

- Geschwindigkeitsteigerung
- Datenmengenreduktion

HEAD EINES HTTP-REQUEST

```
GET /PollingStock//PollingStock HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.5)
Gecko/20091102
Firefox/3.5.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.example.com/PollingStock/
Cookie: showInheritedConstant=false;
showInheritedProtectedConstant=false;
showInheritedProperty=false; showInheritedProtectedProperty=false;
showInheritedMethod=false; showInheritedProtectedMethod=false;
showInheritedEvent=false; showInheritedStyle=false;
showInheritedEffect=false
```

- 1,000 Clients pollen jede Sekunde
  6,968,000 bits per second
  6.6 Mbps

- 10,000 Clients pollen jede Sekunde
  69,680,000 bits per second
  66 Mbps

- 100,000 Clients pollen jede Sekunde
  696,800,000 bits per second
  665 Mbps

HEAD EINES SOCKET-REQUEST

```
\0x00 Hello, WebSocket \0xff
```

0,002%

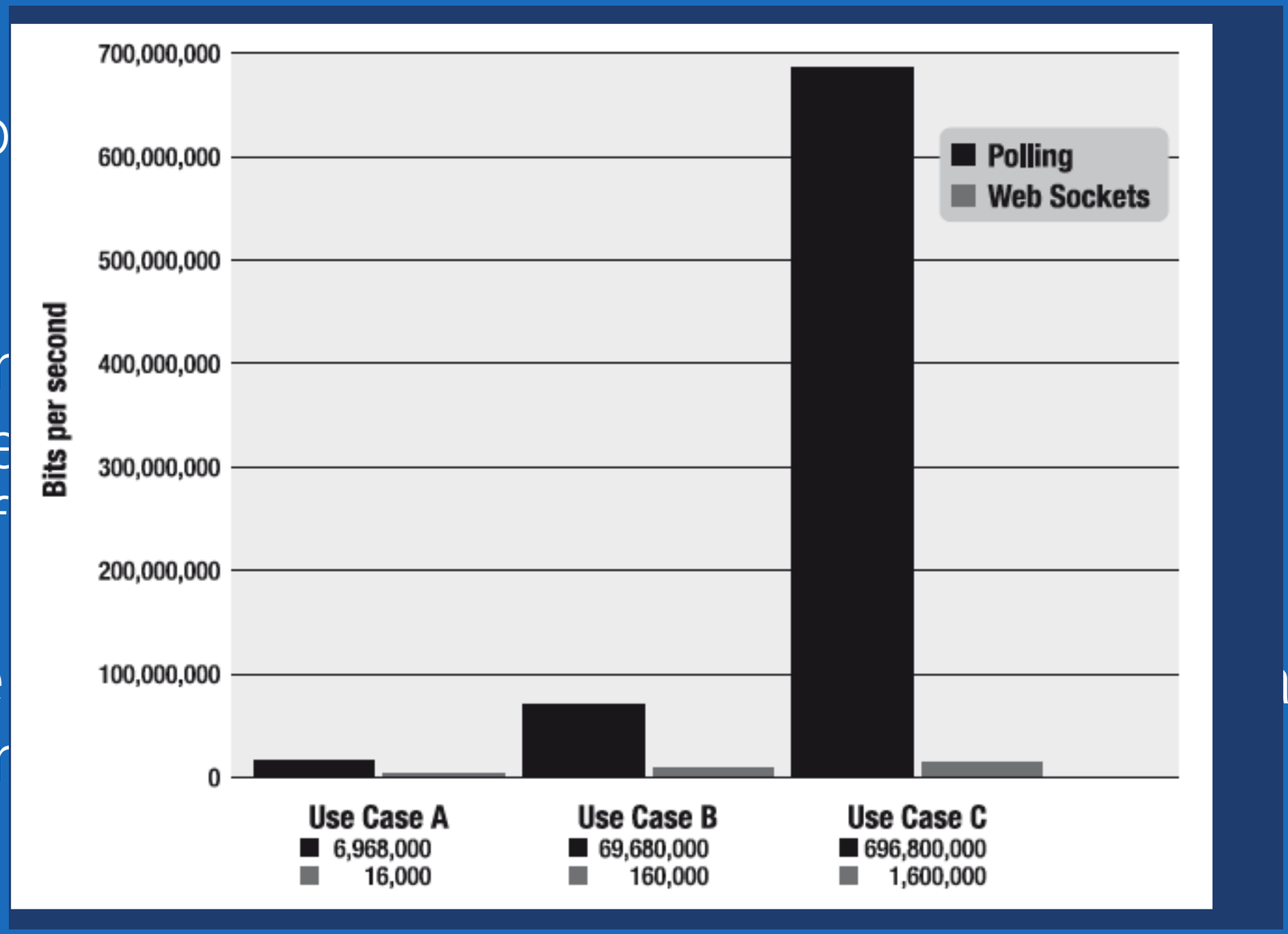
- 1,000 Clients erhalten 1 Nachricht pro Sekunde
  16,000 bits per second
  0,015 Mbps vs. 6.6 Mbps

- 10,000 Clients erhalten 1 Nachricht pro Sekunde
  160,000 bits per second
  0,153 Mbps vs. 66 Mbps ->

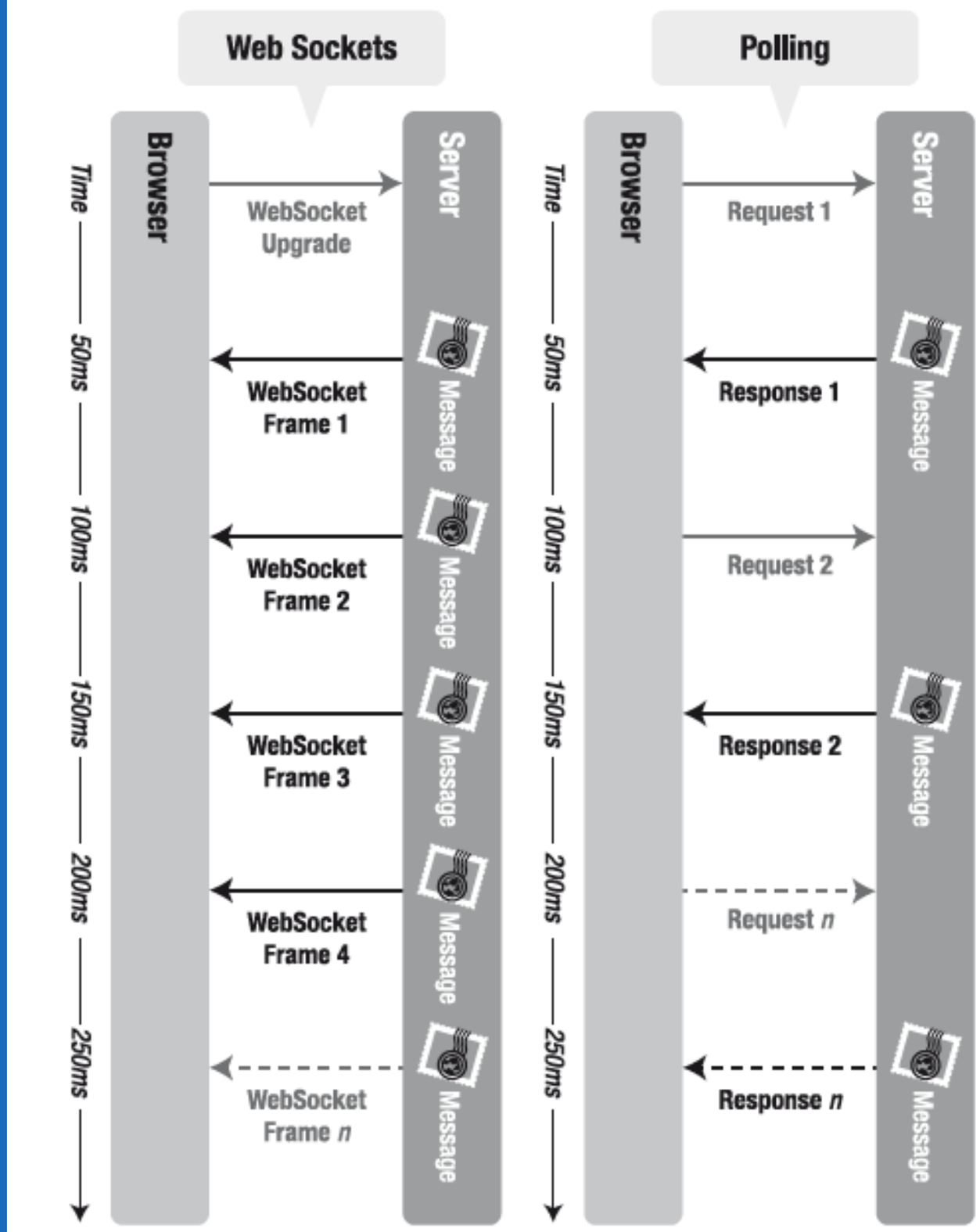
- 100,000 Clients erhalten 1 Nachricht pro Sekunde
  1,600,000 bits per second
  1,526 Mbps vs. 665 Mbps

Datenmengenvergleich des HTTP-Overheads

- Bro
- Chr
  Fire
  Saf
- Die
  Chr



Quelle: Pro HTML 5 - Programming Powerful



DAS WEBSOCKET INTERFACE

```
// The Websocket Interface - start und ready state section

[Constructor(DOMString url, optional (DOMString or DOMString[])
protocols)] interface WebSocket : EventTarget {
  readonly attribute DOMString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;

  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;
...
```

```
// The Websocket Interface – networking

[TreatNonCallableAsNull] attribute Function? onopen;
[TreatNonCallableAsNull] attribute Function? onerror;
[TreatNonCallableAsNull] attribute Function? onclose;

readonly attribute DOMString extensions;
readonly attribute DOMString protocol;

void close([Clamp] optional unsigned short code,
   optional DOMString reason);
...
```

```
//The Websocket Interface – messaging and end section

[TreatNonCallableAsNull]
attribute Function onmessage;
attribute DOMString binaryType;

void send(DOMString data);
void send(ArrayBufferView data);
void send(Blob data);
};
```

Quelle: http://dev.w3.org/html5/websockets/; Hickson

DIE JAVASCRIPT API FÜR DEN CLIENT

DAS READYSTATE ATTRIBUT ENTHÄLT DEN STATUS DER VERBINDUNG

CONNECTING (numeric value 0)
Die Verbindung wurde noch nicht hergestellt.

OPEN (numeric value 1)
Die Verbindung steht, Kommunikation ist möglich.

CLOSING (numeric value 2)
Die Verbindung führt den Closing Handshake aus.

CLOSED (numeric value 3)
Die Verbindung wurde geschlossen oder konnte nicht hergestellt werden.

DIE WEBSOCKET METHODEN

```
mySocket = new WebSocket();

mySocket.onopen = function(evt) {
   console.log(„Connection open ...");
};

mySocket.onmessage = function(evt) {
   console.log( "Received Message: " + evt.data);
};

mySocket.onclose = function(evt) {
   console.log("Connection closed.");
};

mySocket.onerror = function(evt) {
   console.log("An error happened.");
};
```

SOCKET ÖFFNEN UND DATEN SENDEN

```
var mySocket = new WebSocket('ws://game.example.com:12010/
updates');

mySocket.onopen = function () {

   setInterval(function() {

    if (mySocket.bufferedAmount === 0) {
       mySocket.send( );
    }

  }, 50);

};
```

ONMESSAGE

```
mySocket.onmessage = function (event) {

    if (event.data === 'on') {
       turnLampOn();
    } else if (event.data === 'off') {
       turnLampOff();
    }

};
```

JAVASCRIPT AUF DEM SERVER

NODEJS SOCKETS

NODEJS

- node.js ist ein Javascript Framework für den Servereinsatz.
- Es setzt auf der Opensource Javascript Engine V8 auf und wurde 2009 von Ryan Dahl entwickelt.
- Ein Socketproxy in nodejs ist einfach zu schreiben.

- Es läuft unter Windows, MacOs oder Linux. Zur Erweiterung gibt es einen Package Manager (npm).

- Nach der Installation kann node.js in der Konsole/im Terminal ausgeführt werden.

EIN HTTP SERVER IN NODE.JS

```
// server.js

var http = require(„http“);

http.createServer(function (request, response) {

   response.writeHead(200, {'Content-Type': 'text/plain'});
   response.end('Hello World\n');

}).listen(80, '127.0.0.1');

console.log(,http server runs at http://127.0.0.1:1337/');


$ node server
http-server is runs at http://127.0.0.1:80/
```

EIN SOCKETPROXY

```
// Websocket-Server (-> npm install ws!)

var WebSocketServer = require('ws').Server
var wss = new WebSocketServer({
    host: „192.168.2.1“,
    port: 8000
});

wss.on('connection', function(ws) {
    console.log('client verbunden...');

    ws.on('message', function(message) {
        console.log('von Client empfangen: ' + message);
        ws.send('von Server empfangen: ' + message);
     });
});
```

DER BROWSERPART ZUR SOCKETKOMMUNIKATION

```
function connect() {
        // Websocket
        var socket = new WebSocket(„ws://192.168.2.1:8000“);

        socket.onopen = function() {
            console.log(„Socket Status: “
            + socket.readyState + „ (open)“);
        }

        socket.onmessage = function(msg) {
            console.log(„Empfangen: “ + msg.data);
        }

        socket.onerror = function (err) {
            console.log(„Ein Fehler ist aufgetreten.“);
        }

        socket.send(„Hallo Welt“);
}
```

RESTFUL API

WHAT IS REST?

- REST ist ein Akronym für Representational State Transfer.
- Es basiert auf Web-Standards und dem HTTP-Protokoll
- Die REST Architektur beschreibt sechs Vorschriften (nach Roy Fielding)

1. Uniform Interface
2. Stateless
3. Cacheable
4. Client-Server
5. Layered System
6. Code on Demand (optional)

- ReSTful Anwendungen verwenden HTTP Requests, um die vier CRUD Operationen auszuführen.
  C: create, R: read, U: update, und D: delete).

- Für create/update wird POST samt Daten verwendet, GET für das Lesen von Daten, DELETE zum Löschen.

- RESTful wird aus Methoden zusammengesetzt:
  base URL, URL, media types, etc.

http://localhost:3000/tasks/3

```
_ _ _ _
app.use('/Tasks', Tasks);

_ _ _ _
Task = {
 getTaskById: function(id, callback) {
  return db.query("SELECT * FROM task WHERE Id=?", [id],
 callback);
 }
}
```

„ https://jinalshahblog.wordpress.com/2016/10/06/rest-api-using-node-js-and-mysql/ “