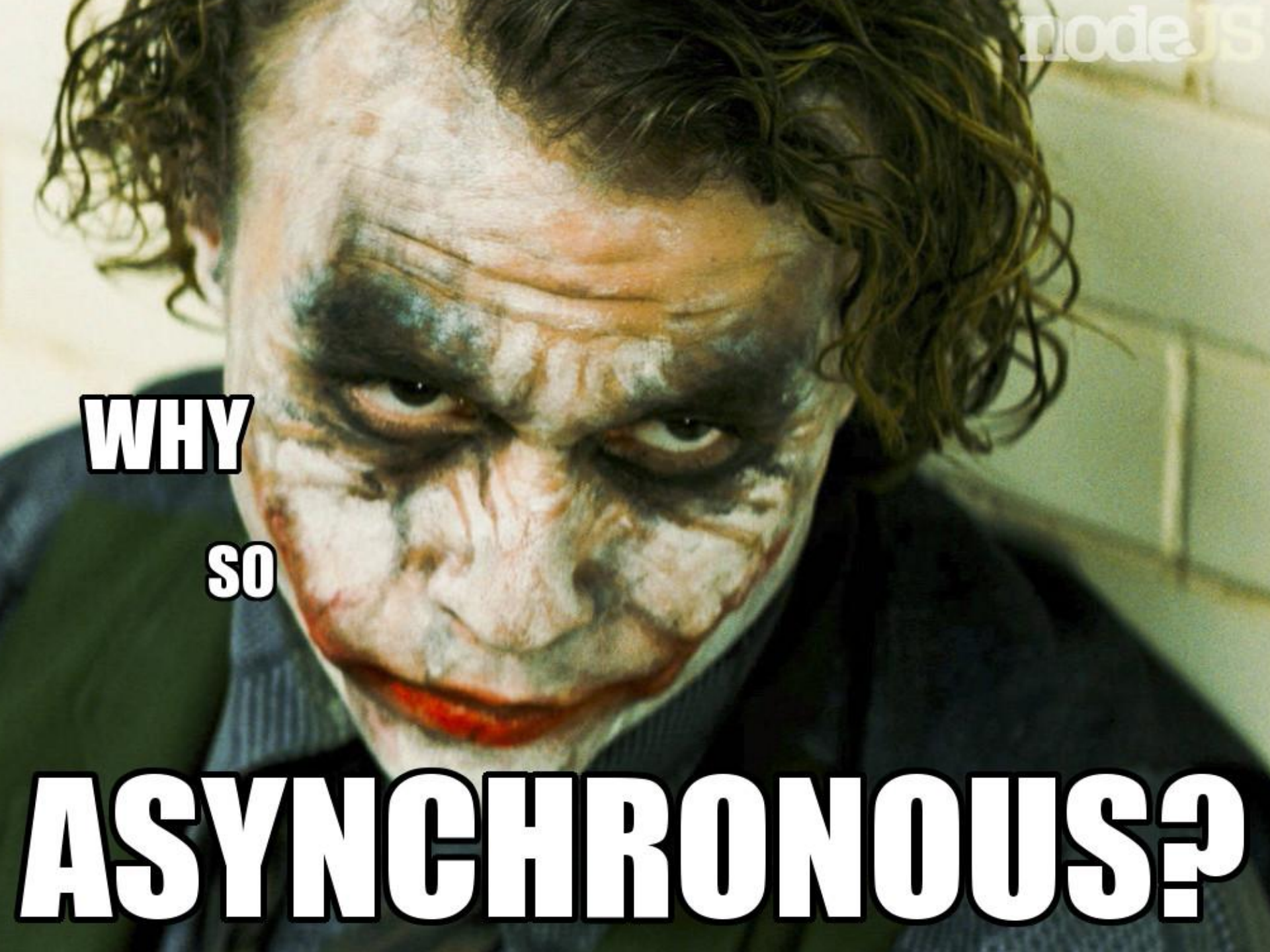

node
JS
BANANENGANG SE2

Inhalt

- Allgemeines über Node.js
- Pro / Contra
- Module
- Code
- Event-Loop
- Einsatzgebiete
- Performance / Statistiken
- Alternativen
- Links


WHY
SO
ASYNCHRONOUS?

Intro

„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 Google Chrome entwickelt wurde, und bietet daher eine ressourcensparende Architektur, die eine besonders grosse Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht.”

Quelle: Wikipedia

Was bedeutet das?

- JavaScript, welcher auf dem Server läuft
- V8
- hohen Konzentrat von Netzwerkverbindungen

- "serverseitiges JavaScript"

- Weit verbreitete Technologie
- Aktuelle Version: 9.2
- Plattformübergreifend
- Programmiersprachen: C++, C, JavaScript
- Kategorie: Framework, Interpreter


Microsoft


PayPal


DOWJONES


YAHOO!


GROUPON


ebay


github
SOCIAL CODING


LinkedIn

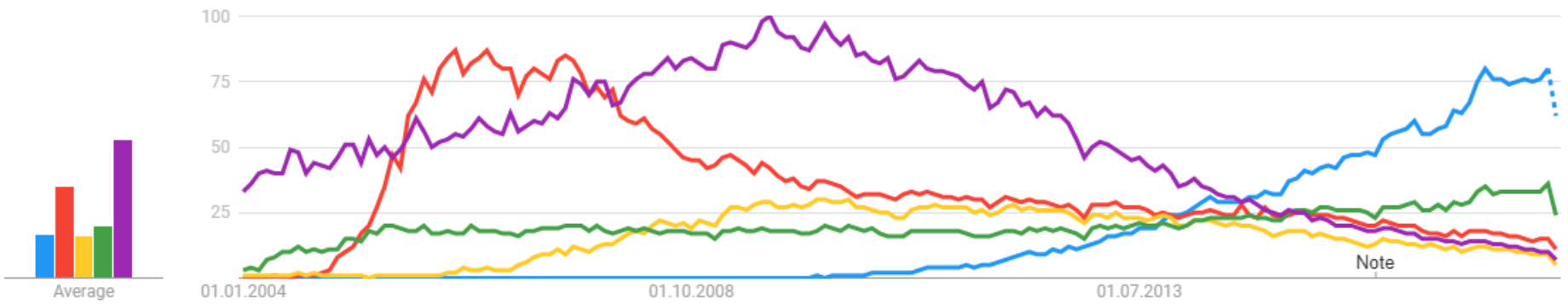

The New York Times

node js	ruby on rails	Grails	Java spring	Zend
Suchbegriff	Suchbegriff	Suchbegriff	Suchbegriff	Suchbegriff

Weltweit | 2004 - heute | Alle Kategorien | Websuche

Sie können sich jetzt Echtzeitdaten für Bilder-, Nachrichten-, Shopping- und YouTube-Suchtrends ansehen.

Interesse im zeitlichen Verlauf



Geschichte

Geschichte

- 2009 von Ryan Dahl erfunden
- Aufgabe: Dateiupload
- Verschiedenste Skript- und Programmiersprachen
- Fand die Lösung in JavaScript
- Node Package Manager und die Windows-Unterstützung

Geschichte

- Nach 5 Jahren gab es immer noch keine stabiles System
- Problem: Viele Unternehmen wollten Node.js für wichtige Systeme einsetzen
- Lösung: Dezember 2014 io.js als Fork
- Juni 2015 wurden die Einzelprojekte Node.js und io.js wieder vereint
- Node.js Foundation
- Heute: Häufige Releases und stabiler Langzeit-Support

V8

- Wichtige Rolle bei der Funktionsweise von Node.js
- Aktuelle und schnelle JavaScript-Engine
- Stets aktuelle Basis
- Just-In-Time-Kompilierung: Operationen schneller und effizienter
- Zweite Kompilierung: Ahead-Of-Time-Kompilierung
- Interne Statistiken
- Weitere Optimierungen

Vorteile

- Basiert auf JavaScript
- Gut dokumentiert
- Große Community
- Weitreichende Basisfunktionen
- nonblocking IO
- Native Unterstützung von JSON

Nachteile

- Einen entscheidenen Nachteil:

Das Stoppen der gesamten Laufzeit bei Auftritt eines Fehlers

Was kann man damit machen?

- HTTP Server und print('Hallo Welt') in nur 4 Zeilen
- TCP Server, ähnlich wie HTTP Server, in nur 4 Zeilen
- DNS Server
- Static File Server
- Web Chat Application
- Online Games

Aspekte für Node.js

- Open Source Server Framework
- Läuft auf vielen Plattformen
- Verarbeitung asynchronischer Ereignisse mit Hilfe von libuv
- Ressourcenschonend in bestimmten Szenarien
- Ermöglicht große Anzahl gleichzeitig bestehender Netzwerkverbindungen

Aspekte für Node.js

- Gleiche Programmiersprache für Browser und Server
- Sehr gutes Module System (npm)
- Bietet viele verschiedene Funktionen

Aspekte gegen Node.js

- CPU intensive Berechnungen serverseitig
- Enterprise level application frameworks

Module

- dasselbe wie JS libraries
- a set of functions you want to include to your application
- um Module zu benutzen: require()

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

- um eigene Module zu erstellen: exports()

```
exports.myDateTime = function () {
    return Date();
};
```

Beispielcode

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Best Practices

- Package JSON
- Style Guide
- automatischer Neustart bei Fehlern
- logging libraries
- immer asynchrone funktionen verwenden

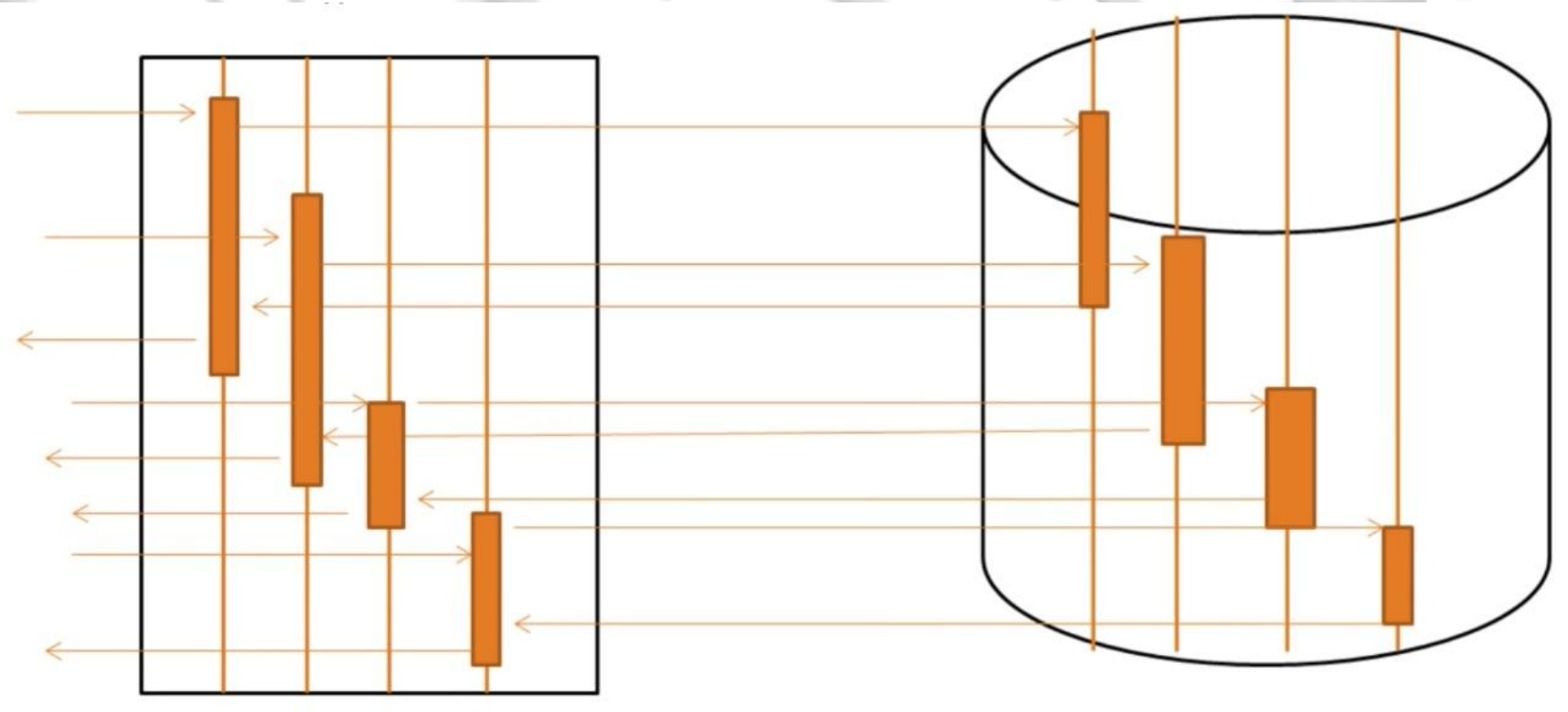
Best Practices

- alle module am anfang requiren
- callbacks validieren
- use strict

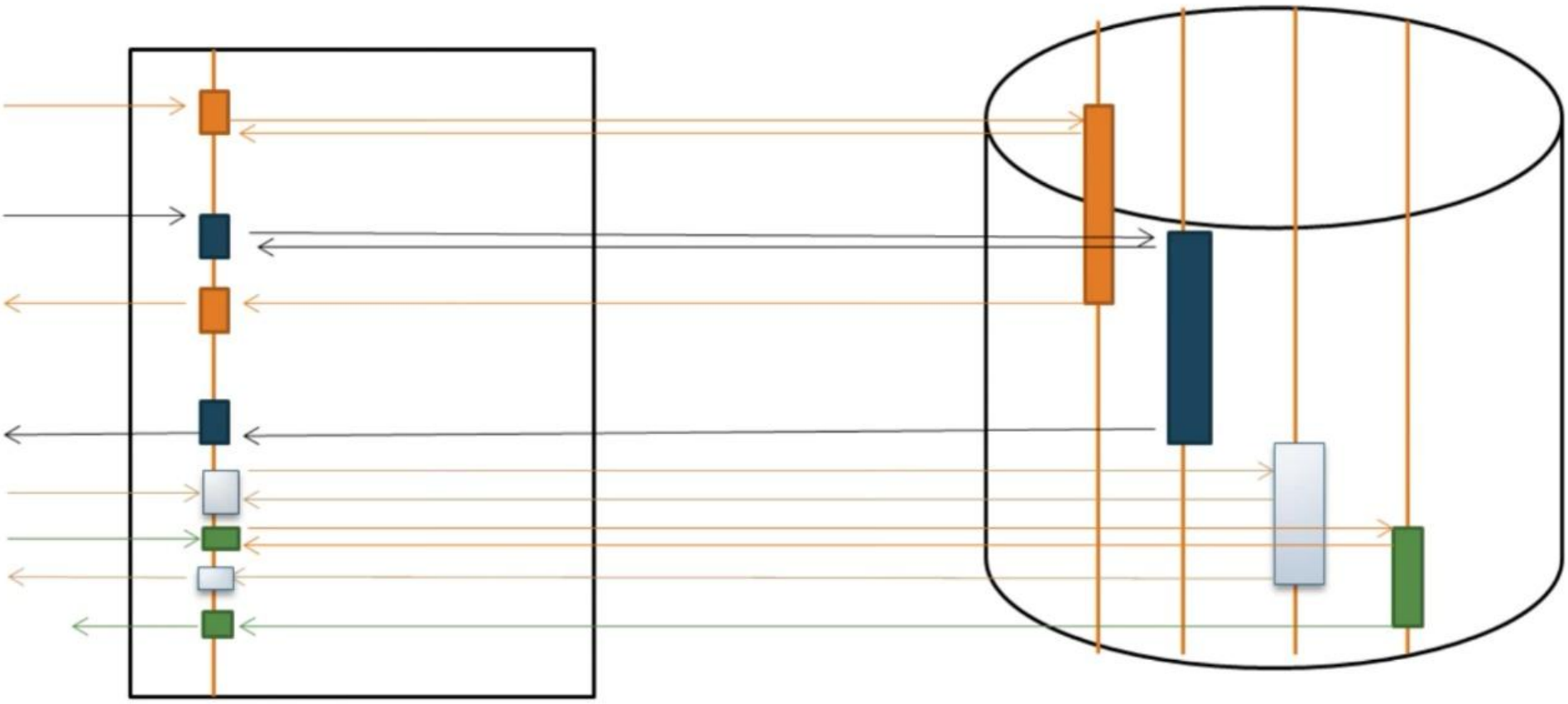
Design Patterns

- Singletons
- Observers
- Factory
- Middleware

Event-Loop



Event-Loop



Event-Loop

- Node.js nutzt nur einen Thread
- Benutzung des Threads nur auf Anfrage oder Antwort
- Keine Blockierung
- Benachrichtigung des Hauptthreads durch Worker Threads
- Vorteil bei I/O-lastigen Anwendungen

Einsatzgebiet

- JSON-basierte REST-Dienste
- Streaming
- Web-Echtzeit
- Single-Page-Anwendungen

Projekte

- Etherpad Lite
- PDFKit
- WebOS
- StackVM
- NodeBB
- Ghost
- Trello

Performance

n-body

source	secs	mem	gz	cpu	cpu load
Node.js	26.61	32,608	1297	26.61	0% 0% 1% 100%
Java	22.10	33,136	1429	22.20	31% 1% 1% 70%

regex-redux

source	secs	mem	gz	cpu	cpu load
Node.js	**4.23**	594,916	432	4.25	78% 1% 0% 24%
Java	10.34	627,224	929	29.88	74% 72% 65% 79%

binary-trees

source	secs	mem	gz	cpu	cpu load
Node.js	45.67	657,468	440	81.85	45% 39% 39% 58%
Java	8.34	893,008	835	28.20	87% 79% 96% 81%

n-body

source	secs	mem	gz	cpu	cpu load
Node.js	**26.61**	32,608	1297	26.61	0% 0% 1% 100%
Python 3	838.39	10,324	1196	838.20	95% 1% 5% 0%

regex-redux

source	secs	mem	gz	cpu	cpu load
Node.js	**4.23**	594,916	432	4.25	78% 1% 0% 24%
Python 3	15.22	447,324	512	27.44	25% 33% 32% 91%

binary-trees

source	secs	mem	gz	cpu	cpu load
Node.js	**45.67**	657,468	440	81.85	45% 39% 39% 58%
Python 3	93.55	280,624	589	337.74	92% 89% 87% 93%

n-body

source	secs	mem	gz	cpu	cpu load
Node.js	**26.61**	32,608	1297	26.61	0% 0% 1% 100%
Ruby	723.69	8,916	1137	723.48	25% 86% 1% 9%

binary-trees

source	secs	mem	gz	cpu	cpu load
Node.js	**45.67**	657,468	440	81.85	45% 39% 39% 58%
Ruby	54.24	510,868	1078	153.99	73% 98% 67% 72%

n-body

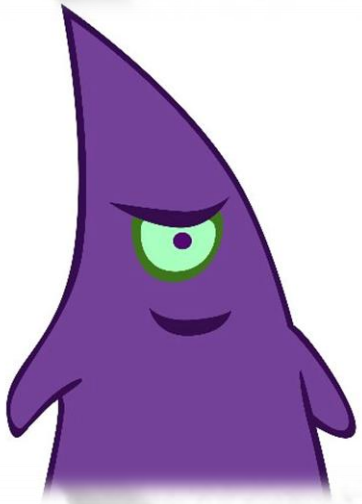
source	secs	mem	gz	cpu	cpu load
Node.js	26.61	32,608	1297	26.61	0% 0% 1% 100%
C++ g++	8.23	1,856	1763	8.23	100% 1% 1% 0%

Alternativen

- EventMachine für Ruby
- libuv für C
- Perl Object Environment für Perl
- ReactPHP (libev oder libevent) oder Amp (libev, libevent, oder libuv) für PHP
- Twisted für Python
- Vert.x für Java, JavaScript, Apache Groovy, Ruby, Python, Scala, Clojure, and Ceylon

Alternativen für Java

- RingoJS
- Nodyn
- PurpleJS



Links zum Lernen

- node.js https://nodejs.org/api
- node.js http://www.w3ii.com/de/nodejs/default.html
- web apis http://devdocs.io
- testing node.js server + npm online https://runkit.com/home
- testing node.js browserified bundles http://requirebin.com/

Quellen

- https://ringojs.org/
- https://webagility.com/posts/5-alternatives-to-nodejs-for-java
- https://netzleben.com/node-js-eine-einfuehrung-fuer-anfaenger-teil1
- https://de.slideshare.net/nodeio/nodejs-eine-kurze-einfhrung
- https://www.heise.de/developer/artikel/2x-Nein-4x-Ja-Szenarien-fuer-Node-js-2111050.html
- https://de.wikipedia.org/wiki/Node.js
- https://benchmarksgame.alioth.debian.org/u64q/javascript.html
- http://t3n.de/magazin/serverseitige-javascript-entwicklung-nodejs-einsatz-231152/
- http://www.fh-wedel.de/~si/seminare/ss11/Ausarbeitung/08.nodejs/konzepte.html
- https://www.w3schools.com/nodejs/nodejs_modules.asp
- https://github.com/alanjames1987/Node.js-Best-Practices
- https://github.com/FredKSchott/the-node-way
- https://blog.risingstack.com/node-js-best-practices/
- https://blog.risingstack.com/fundamental-node-js-design-patterns/

ENDE

Danke für eure Aufmerksamkeit ☺