LT 56 - NODE.JS

ES GIBT AUCH SCHÖNE MOMENTE

Ralf Enderle

BE AWARE!

Wir betrachten in diesem Lighting Talk ausschließlich die schönen Effekte einiger Feature von Node.js bzw. JavaScript. Über die dunklen Seiten haben wir schon oft genug geredet ;-)

IMPLIZITER CAST - TYPE COERCION

Ich bekomme Numerische Daten über meine Schnittstelle leider nicht immer korrekt - manche Sender können nur Strings

```
3   === 3 // true
"3" === 3 // false
// ------ aber ------
3    == 3 // true
"3"  == 3 // true
```

STRINGS IN STRINGS... EASY

Es gibt ja ' und "

```
// java way:
LOG.debug("Valid data for attribute \"lastname\"");
// JS way:
LOG.debug('Valid data for attribute "lastname"');
```

FUNCTION BINDING

Gib deiner Methode eine neue Heimat - vergiss **that = this** und setzte den passenden Kontext!

```
var myObj = {
    specialFunction: function () {
    },
    getAsyncData: function (cb) {
        cb();
    },
    render: function () {
        this.getAsyncData(function () {
            this.specialFunction();
        });
    }
};
myObj.render();
```

TypeError: this.specialFunction is not a function

```
var myObj = {
    specialFunction: function () {
    },
    getAsyncData: function (cb) {
        cb();
    },
    render: function () {
        this.getAsyncData(function () {
            this.specialFunction();
        }.bind(this));
    }
};
myObj.render();
```

NODE.JS

...

FAIL FAST

Uncaught exceptions führen in Node zu einem Stopp des Servers

- Folgefehler somit oft kein Thema
- Tools zur Überwachung + restart
- typische Zeit zum restart < 2 sec
- forever, nodemon, node-supervisor, ...

```
$ forever -w app.js
```

SIMPLE SCALING

Ein Node Server läuft eigentlich 'single thread' - lässt sich aber sehr einfach forken

```
var cluster = require('cluster');
var http = require('http');
var numCPUs = 5;
if (cluster.isMaster) {
  for (var i = 0; i < numCPUs; i++) {
    cluster.fork();
  }
} else {
  http.createServer(function(req, res) {
    res.writeHead(200);
    res.end("hello world\n");
  }).listen(8000);
}
```

A BUNCH OF BUILD TOOLS

Trotz der recht jungen Historie von Node ist die Auswahl an Build Tools und Plugins schon recht groß.

- grunt
- gulp
- brunch
- broccoli
- ...

NODE MODULES

Abhängigkeiten mal ganz einfach gemacht

- Module bringen ihre eigenen (Versionen) von dependencies mit
- Menge an Modulen ist gigantisch: > 120.000
- Plattformspezifika werden ausgenutzt - tw. compile bei 'npm install'
- Eigene Module lassen sich sehr einfach veröffentlichen

```
npm adduser
// create new awesome module
npm publish
```

NODE MODULES

Zyklische Abhängigkeiten sind kein Problem

- Module werden quasi als Singletons registriert
- Node lädt ein Module so weit es geht
- Bei Zyklen wird mit dem nächsten Modul weitergemacht
- das ganze beginnt von vorne...
- irgendwann sind alle Module da ;-)

PASSENDES BACKEND - KEIN AUFWAND!

Mit einem passenden JSON basierten Backend laufen viele Dinge ohne Konvertierung ab

- Objekte z.B. aus der MongoDB sind quasi schon JS Objekte
- Das umständliche und teure Mapping fällt weg
- Objekte aus der DB können direkt erweitert werden

BLAZING FAST - SINGLE THREADED

Durch Non blocking IO und Single thread kann Node sehr viele Connections bedienen

- Proof of concepts mit 1 Million concurrent connections
- Kein Thread overhead
- Kein Context switch notwendig
- Praktisch alles läuft asynchron
- Vorsicht jedoch bei großen Berechnungen

NODE PIPELINES

Pattern in Node: Call -> Step1, Step2, Step3, ... , -> fertig

- Pipelines in Node sind allgegenwärtig
- man kann (fast) an beliebigen Stellen eine middleware dazwischenklemmen
- sehr viele Frameworks folgen der Idee und lassen sich kombinieren

JAVASCRIPT FULL STACK

Browser + Server

- nur eine Sprache
- perfekt geeignet für beide Welten
- ich kann mich als Entwickler darauf konzentrieren
- codesharing (.z.B. für Validierungen) einfach möglich

DANKESCHÖN

für die Aufmerksamkeit