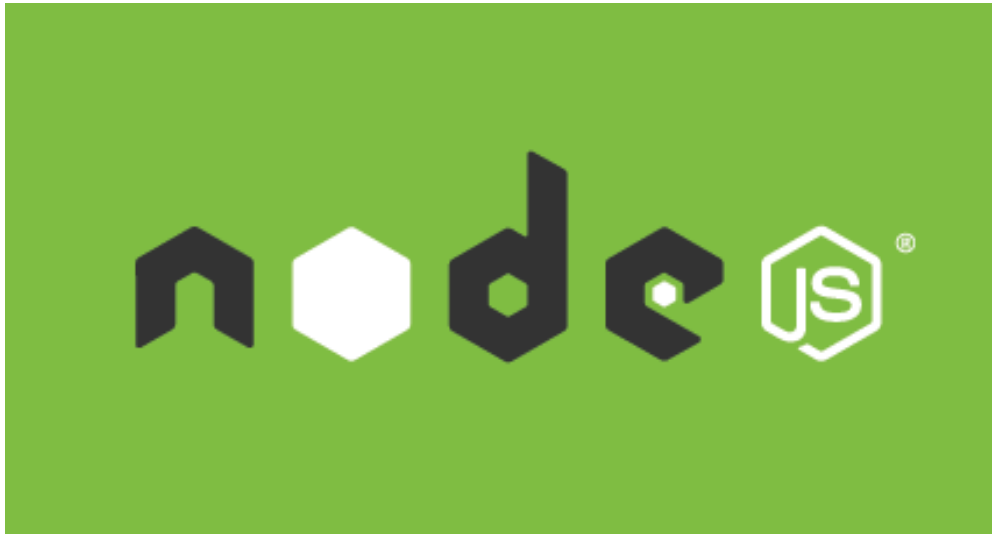


Node.js

Schnupperkurs für Anfänger

Anastasiia Aftakhova
Friedrich-Schiller-Universität Jena

04. Mai 2017

Was ist Node.js?

"Node.js is many things, but mostly it's a way of running JavaScript outside the web browser." – [1] *Node: Up and Running*

- JavaScript Interpreter

Was ist gut an Node.js?

- 💪 Implementiert mit C
- ⏱️ V8 JavaScript Runtime
- 🛠️ Modularität
- 💬 Community

Was kann Node.js?

- WebSockets
- File Upload Clients
- Real-Time Data Apps

und viel mehr...

- Installieren → nodejs.org
- Node.js REPL starten:

Auswählen Eingabeaufforderung - node

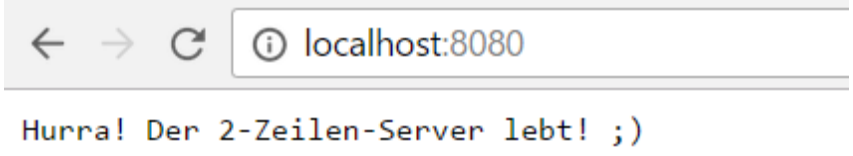
```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\aftak>node -v
v6.10.2

C:\Users\aftak>node
> 'Hi, Node!'
'Hi, Node!'
>
```

- Experimentieren!

```
var http = require('http');
    http.createServer(function(request, response) {
        response.writeHead(200);
        response.write("Hurra! Der kleine Server lebt!");
        response.end();
    }).listen(8080, function(){
        console.log('Listening on port 8080...');
    });
```

Server mit Node.js starten:

Im Browser ausprobieren:



- CommonJS Module Spezifikation
  - DRY - Don't repeat yourself
  - lokaler Kontext
  - Variable exports
  - Modul einbinden mit require('modulname')
- Kernmodule (Core APIs)
  - EventEmmitter – Events Handling
  - HTTP – Connection & Requests Handling
  - Streams – Data Handling, Buffering
- npm Package Manager → npm universe

- EventEmitter erzeugen

```
var util = require('util'),
    EventEmitter = require('events').EventEmitter;

function CustomEmitter() {
    EventEmitter.call(this);
};
util.inherits(CustomEmitter, EventEmitter);

var customEmitter = new CustomEmitter();
```

- Event erzeugen

```
customEmitter.emit('customEvent', 'valid');
customEmitter.emit('customEvent', 'valid');
customEmitter.emit('customEvent', 'erroneous');
```

- Auf Event hören

```
var addnewlistener = function() {
  // catch event every time
  customEmitter.on('customEvent', function(par) {
    console.log('customEvent occured: ' + par);
  });
};

// catch event only once
customEmitter.once('customEvent', function(par) {
  console.log('first time customEvent occured: ' + par);
  addnewlistener();
});
```

- Fehler werfen

```
customEmitter.on('customEvent', function(par) {
  if (parameter == 'valid') {
    console.log('customEvent occured: ' + par);
  } else if (parameter == 'erroneous') {
    // error emit point
    customEmitter.emit('error',
        new Error('Errouneous parameter!')
    );
  }
});

// error listener
customEmitter.on('error', (err) => {
    console.log(err);
});
```

- Request-Response Protokoll zwischen dem Client und Server
- Client sendet einen Request, evtl. Daten an den Server
- Server sendet eine Response, evtl. Daten zurück

- Request-Response Protokoll zwischen dem Client und Server
- Client sendet einen Request, evtl. Daten an den Server
- Server sendet eine Response, evtl. Daten zurück

Request

- URL
- Typ: GET, POST, PUT...
- Daten
- Headers: Accept, Connection, Content-Type...

Response

- Status: 200, 404, 500...
- Daten
- Headers

- Request-Response Protokoll zwischen dem Client und Server
- Client sendet einen Request, evtl. Daten an den Server
- Server sendet eine Response, evtl. Daten zurück

Request

- URL
- Typ: GET, POST, PUT...
- Daten
- Headers: Accept, Connection, Content-Type...

Response

- Status: 200, 404, 500...
- Daten
- Headers

GET-Request an 2-Zeilen-Server:

- Server anlegen

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

var server = http.createServer();
server.listen(8080, function() {/*...*/});
```

- Antwort senden

```
function showWelcomePage(request, response) {
  var html = '<!DOCTYPE html><html><head><title>'
      + 'NodeServer - 200 OK</title></head><body><h3>'
      + 'Wilkommen zu Node Server</h3></body></html>';
    response.writeHead(200,
      {'Content-Type': 'text/html'});
    response.end(html);
}

server.on('request', function (request, response) {
  showWelcomePage(request, response);
});
```

- Anfragen bearbeiten

```
function showRequestHeaders(request, response) {/*...*/};
function showNotFoundPage(request, response) {/*...*/};

server.on('request', function (request, response) {
  console.log('request.url: ' + request.url);

  if (request.method === 'GET') {
    if (request.url === '/') {
      showWelcomePage(request, response);
    } else if (request.url === '/requestheaders') {
      showRequestHeaders(request, response);
    } else {
      showNotFoundPage(request, response);
    }
  } else {
    showNotFoundPage(request, response);
  }
});
```

- Fehlerbehandlung

```
function showErrorPage(err, response) {/*...*/}

server.on('request', function (request, response) {
  console.log('request.url: ' + request.url);

  if (request.method === 'GET') {
    if (request.url === '/') {/*...*/}
    else if (request.url === '/requestheaders') {/*...*/}
    else if (request.url === '/errorpage') {
      var errmsg = 'Ein Server-Fehler ist aufgetreten.';
      showErrorPage(errmsg, response);
      server.emit('error', new Error(errmsg));
    } else {/*...*/}
  } else {/*...*/}
});

server.on('error', (err) => {
  console.log(err);
});
```

Stream Typen:

- Readable
- Writable
- Duplex
  (= Readable + Writable)
- Transform

Stream Beispiele:

- HTTP requests,
- HTTP responses
- File System Stream
  ('fs' Modul)

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

var server = http.createServer();
server.listen(8080, function() {
    console.log('Listening for port 8080...');
});

server.getdata = function(response) {
    var readStream = fs.createReadStream('bsptext.txt',
        {encoding: 'utf8'});
    readStream.pipe(response);
};

server.on('request', function (request, response) {
    if (request.url === '/gettxt') {
        response.writeHead(200,
            {'Content-Type': 'text/plain;charset=utf-8'});
        server.getdata(response);
    } else {
        response.writeHead(200, {});
        response.end('Welcome to Nodejs Server');
    }
});
```

- Routing Node.js:

```
http.on('/', function(req, res) {
    if (req.URL === 'POST') {
        res.writeHeader(200, {});
        res.end('Hello World!');
    }
});
```

Express:

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

- Intergriertes/Unterstütztes Middleware
  body-parser, compression, cookie-parser...
- HTML Template Engines: ejs, pug, jade...
- App-Generator

Vielen Dank für die Aufmerksamkeit!

Die Folien und den Quellcode für die Beispiele:

`https://github.com/aaaftakhova/NodejsVortragPraktischesTeil`

- Tom Hughes-Croucher and Mike Wilson. *Node: Up and Running*. Scalable Server-Side Code with JavaScript. O'Reilly Media, Apr. 2012. ISBN: 978-1-4493-9876-7.
- Inc Joyent. *Node.js v5.12.0 Manual & Documentation*. 2017. URL: https://nodejs.org/docs/latest-v5.x/api/ (visited on 05/04/2017).
- Thomas Steur. *CommonJS – Plattformübergreifende JavaScript Spezifikationen*. 2012. URL: https://blog.mayflower.de/860-CommonJS-Plattformuebergreifende-JavaScript-Spezifikationen.html (visited on 05/04/2017).
- StrongLoop and IBM. *Express*. Fast, unopinionated, minimalist web framework for Node.js. 2016. URL: http://expressjs.com/ (visited on 05/04/2017).