Node.js

DI. H. Lackinger

2016/2017

Contents

---

node.js ist eine Runtime-Umgebung zur Ausführung von stand-alone JavaScript-Programmen. Dadurch kann JavaScript nicht nur clientseitig (im Browser), sondern auch serverseitig zur Anwendung kommen.

node.js ist plattform-unabhängig und ist auf vielen gängigen Betriebssystemen wie Linux, OS oder Windows lauffähig.

node.js ist für viele Bereiche wie Netzwerk, Datenbanken, etc. einsatzfähig

Installation von node.js

siehe http://www.nodejs.org

npm - node package manager

dient hauptsächlich zur Installation von node-Zusatzpaketen Weiters können damit auch Skripte zum Testen, Deployen, etc. definiert und ausgeführt werden.

Dokumentation siehe: - https://docs.npmjs.com/ - ./node_modules/npm/html/doc/index.html (lokal)

npm cheat sheet

- http://browsenpm.org/help

Installation

npm kann gemeinsam mit node.js installiert werden.

- siehe http://www.nodejs.org

Konfiguration (package.json)

Die Konfuguration erfolg über ein File mit dem Namen "package.json". Dieses File kann erzeugt werden durch:

~~~bash npm init~~~

Beispiel package.json:

```
{
  "name": "FirstNodeApp",
  "version": "1.0.0",
  "description": "First Node Applikation for demonstration",
  "main":"myClass.js",
  "scripts": {
    "start": "tsc && concurrently node myClass.js ",
    "test": "echo TEST",
    "tsc": "tsc",
    "tsc:w": "tsc -w"
  },
  "keywords": ["npm","demo"],
  "author": "H. Lackinger",
  "license": "ISC",
  "repository": {}
}
```

Installieren/Deinstallieren von packages

```
npm install *package_name*
npm install *package_name*  -g          // global
npm install *package_name*  --save      // lokal
npm install *package_name*  --save-dev  // lokal
```
~~~

```
npm uninstall ...
```

Starten von Scripts

```
npm run-script *script*
npm run *script*
npm run *script*:dev  (Starten mit dev-Eigenschaften)
```

Abkürzungen für vordefinierte Scripts

- npm start (npm start start)
- npm stop (npm start stop)
- npm test (npm start test)
- npm restart (npm start restart)

globales npm-Verzeichnis festlegen

Um in das standardmäßg eingestellte globale npm-Verzeichnis schreiben zu können, sind tw. root-Berechtigungen erforderlich. Das kann dadurch umgangnen werden, indem das globale npm-Verzeichnis in ein eigenes User-Verzeichnis gelegt wird.

```
npm config set prefix \userdir\...
```

Achtung: Auch der Pfad muss angepasst werden, damit die npm Programme und Files vom System gefunden werden

Ein erstes nodejs-Beispiel (app.js)

- Anlegen und Editieren einer Datei app.js

```
console.log('hello world');
```

- starten

```
node app.js
```

Weiteres Beispiel: Webserver

- server.js editieren

```
var http = require('http');
http.createServer(function (request, response) {
  response.writeHead(200, {'content-type': 'text/plain; charset=utf-8'});
  var body = 'hello node.js';
  response.end(body);
}).listen(8080, '127.0.0.1');
console.log('Webserver wird ausgeführt.');
```

- Webserver starten

```
node server.js
```

- Browser öffnen und als URL

```
http://127.0.0.1:8080
oder
http://localhost:8080
```

eingeben.

REPL

REPL ist das Tool für den interaktiven Modus von node.js.

Der Start erfolgt durch

```
node
```

Anschliessend kann JavaScript-Code eingegeben und ausgeführt werden.

Beispiel:

```
$ node
> console.log("Hallo Welt");
Hallo Welt
undefined
>
```

Weitere REPL-Befehle sind:

```
.break
.clear
.exit
.help
.load
.save
```

Entwicklungsumgebungen (IDEs)

Zur Entwicklung größerer Projekt reicht REPL natürlich nicht aus, deswegen gibt es dafür eine Reihe von Tools, Editoren, Testing-Frameworks, usw.

Für schulische Zwecke kann WebStorm (von JetBrains) kostenlos verwendet werden.

HTTP

Mit diesem Modul lassen sich HTTP-Applikationen implementieren. Es ist nach der Installation von node bereits integriert und braucht nicht explizit installiert werden.

Beispiel: HTTP-Server

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

http.createServer(
function(request,response) {
   response.writeHead(200,{'content-type': 'text/plain'});
   response.end('Hello World');
}
).listen(8080, 'localhost');
```

Beispiel: HTTP-Client mit node.js

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

http.request('http://www.google.de', function (response) {
  response.setEncoding('utf8');
  response.on('readable',function () {
    console.log(response.read());
  });
}).end();
```

Express

ist ein Zusatzmodul für node.js, das die Entwicklung von Webapplikationen erleichtert.

Aufbau

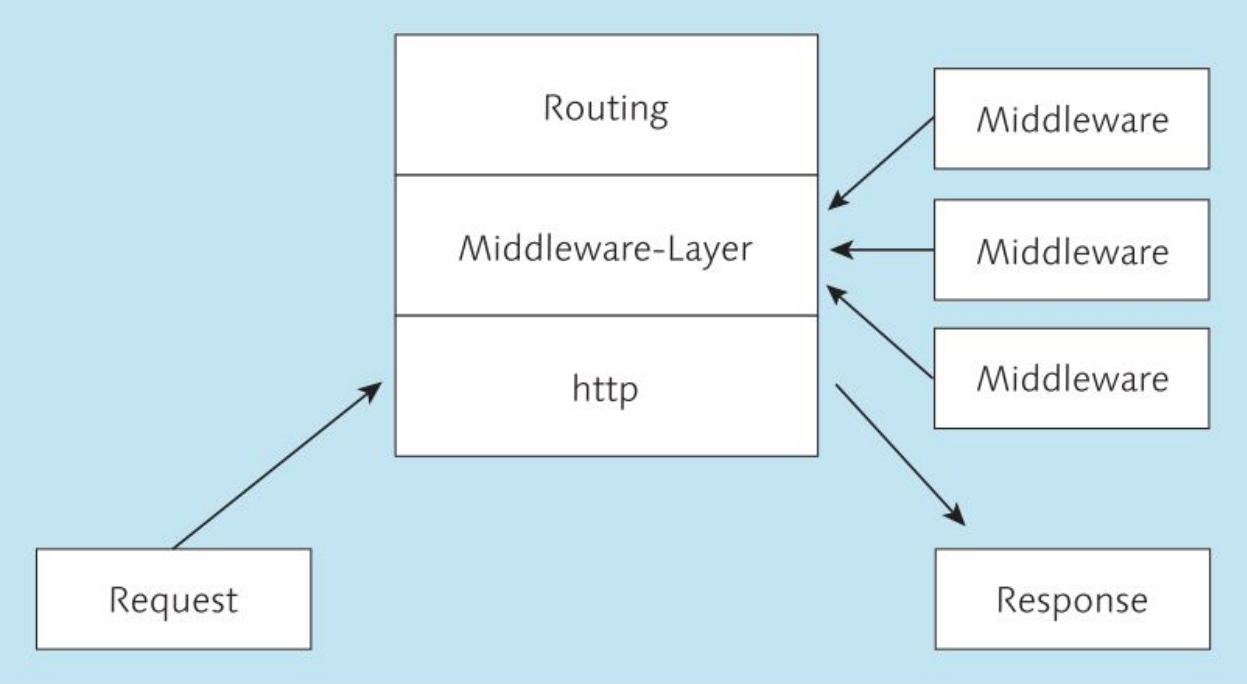

Figure 1: Express

Installation

```
npm install express
```

Beispiel: Webserver mit express

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

app.get('/', function (req, res) {
  res.send('My first express application');
```

```
});

app.listen(8080, function () {
  console.log('Server listening on port 8080')
});
```

Routing

Der Router ist eine der wichtigsten Komponenten in einer Express.js-Applikation. Eine Anfrage eines Benutzers geht ein und wird anhand des gewählten URL-Pfades an eine Funktion weitergeleitet, die dafür zuständig ist, eine Antwort zu generieren.

Beispiel:

```
module.exports= function(app){
  app.get('/',function (req,res){
    res.send('Myfirstexpressapplication');
  });
  app.get('/list',function (req,res){
    res.send('Gamma,Helm,Johnson,Vlissides');
  });
  app.post('/list',function(req,res){
    console.log('Createnew user');
    res.send('Createnewuser');
  });
  app.all('/info',function(req,res){
    console.log('ALL/info\n');
    res.send('ALL/info');
  });
};
```

Muster in Routen: regular expressions

```
module.exports = function (app) {
  app.get(/.*\/user.*$/, function (req, res) {
    res.end('User Route');
  });
};
```

Variablen in Routen

```
module.exports = function (app) {
  app.get('/user/:name', function (req, res) {
    res.end(`Hello ${req.params.name}`);
  });
};
```

Mehrere Callbacks pro Route

```
module.exports = function (app) {
  app.get('/',
    function (req, res, next) {
      res.set('X-Powered-By', 'Node.js');
      next();
  },
    function (req, res) {
```

```
      res.send('My first express application');
    });
};
```

Alternative 1: Route-Objekt Damit ist es relativ übersichtlich möglich zu einer Route mehrere Methoden zu definieren.

```
module.exports= function(app){
  app.route('/user')
   .get(function(req,res){
     //fetchuserdata
     res.send('Userdata');
    })
    .post(function(req,res){
      //createuser
      res.send('Usercreated');
    })
    .put(function(req,res){
      //updateuser
      res.send('Userupdated');
    })
    .delete(function(req,res){
      //deleteuser
      res.send('Userdeleted');
    });
};
```

Alternative 2: Router - Objekt

```
var express = require('express');
var router = express.Router();

var app = express();

router.get('/groups', function(req, res) {
  res.send('Groups');
});
router.get('/permissions',function (req, res){
  res.send('Permissions');
});
app.use('/user', router);
app.listen(8080);
```

Logging

Das Protokollieren eingehender Requests kann mit Hilfe der Middleware **morgan** sehr einfach durchgeführt werden.

Dokumentation: - https://github.com/expressjs/morgan

Installation:

```
npm install morgan
```

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

```
var app = express();

var options = {
immediate: true
};

app.use(morgan('combined', options));
require('./router')(app);
app.listen(8080, function () {
  console.log('Server listening on port 8080')
});
```

serve-static

Webapplikationen, die mit express.js implementiert werden, bestehen in der Regel nicht nur aus dynamischen Inhalten, sondern benötigen auch statische Dateien. So müssen HTML-, JavaScript-, CSS- und Bilddateien geladen werden. Am Einfachsten kann dies mit der serve-static-Middleware gemacht werden.

Dokumentation: - https://github.com/expressjs/serve-static

Installation:

```
npm install serve-static
```

Beispiel:

```
var express = require('express');
var morgan = require('morgan');
var serveStatic = require('serve-static');

var app = express();

var options = {
   immediate: true
};
app.use(morgan('combined', options));
app.use(serveStatic('public'));

require('./router')(app);

app.listen(8080, function () {
  console.log('Server listening on port 8080')
});
```

Verbindung nodeJS-Server mit Angular - Client

Server: serveStatic mit Verzeichnis des Angular2-Projektes definieren

Beispiel:

```
...
app.use(serveStatic('C:\\H T L\\JavaScript\\Angular2\\_Uebungen\\05_PersonsDetailsHttp_V2.0.0'));
...
```

Client: in der Angular2-App über localhost zugreifen

Beispiel:

```
...
      this.http.get('http://localhost:8080/persons.json')
            .subscribe((res: Response) => this.persons = res.json() );
...
```

Body Parser

Das Auslesen von Daten, die vom Client übermittelt werden, gestaltet sich gerade bei POST-Requests etwas umständlich. Aus diesem Grund gibt es die Body-Parser-Middleware.

Sie sorgt dafür, dass die Daten aus der Anfrage aufgeschlüsselt werden undüber das body-Objekt innerhalb des Request-Objekts als einzelne Eigenschaften verfügbar sind.

Dokumentation: - https://github.com/expressjs/body-parser

Installation:

```
npm install body-parser
```

Beispiel:

```
var express = require('express');
var morgan = require('morgan');
var serveStatic = require('serve-static');
var bodyParser= require('body-parser');

var app = express();

app.use(morgan('combined', {immediate:true}));
app.use(serveStatic('public'));
app.use(bodyParser.json());
require('./router')(app);
app.listen(8080, function () {
   console.log('Server listening on port 8080')
});
```

Datenbank-Anschluss

MongoDB

Useful Links: https://www.tutorialspoint.com/mongodb/mongodb_quick_guide.htm

Beispiel: Verbindung herstellen

```
var url= 'mongodb://localhost:27017/node';
MongoClient.connect(url, function(err,db){
     var collection=db.collection('Addresses');
     // ...
     db.close();
});
```

Beispiel: Datensätze einfügen

```
var address={street:'Limesstr',
             place:'Leonding',
             country: 'Austria'
```

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Figure 2: Vergleich

```
            };
collection.insert(address,{safe: true},function(err,res){
       if (err) throw err;
      console.log(res);
});
```

Beispiel: Datensätze lesen

```
collection.find({}, { _id: 0 }).toArray(function(err,docs){
       if (err) throw err;
       console.log(docs);
});
```

Beispiel: Datensätze updaten

```
var new Address={street: 'Tower Hill',
                 place:'London',
                 country: 'United Kingdom'
                };

collection.update({street:'Limesstr'},
                  {$set: newAddress},
                  {safe: true},
                  function (err) {
                            if (err) throw err;
                  });
```

Beispiel: Datensatz löschen

```
collection.remove({street:'Tower Hill'},
                  {safe: true},
                  function (err,res) {
                          if (err)throw err;
                  });
```

REST Server

Beispiel:

```
var express = require('express');
var morgan = require('morgan');
var serveStatic = require('serve-static');
var MongoClient= require('mongodb').MongoClient;
var bodyParser= require('body-parser');

var app = express();

app.use(serveStatic('public'));

var options = {
immediate: true
};
app.use(morgan('combined', options));

//app.use(bodyParser.urlencoded({extended: false}));
app.use(bodyParser.json());

// Suchen aller Personen
// URL: http://localhost:8080/rest
// Method: GET
app.get('/rest', function (req, res) {
    var url= 'mongodb://localhost:27017/test';  // test ist der Name der DB
    MongoClient.connect(url, function(err,db){
    var collection=db.collection('persons');
    collection.find({}, { _id: 0 }).toArray(function(err,docs){
      if (err) throw err;
      console.log(docs);
      res.send(docs);
      });
    db.close();
    });
});

// Suchen EINER Personen
// URL: http://localhost:8080/rest/17
// Method: GET
app.get('/rest/:id', function (req, res) {
    var url= 'mongodb://localhost:27017/test';  // test ist der Name der DB
    MongoClient.connect(url, function(err,db){
    var collection=db.collection('persons');
    var toFind={};
```

```
    toFind.id=parseInt(req.params.id);
    collection.find(toFind, { _id: 0 }).toArray(function(err,docs){
      if (err) throw err;
      console.log(docs);
      res.send(docs);
      });
    db.close();
    });
});

// Löschen einer Person
// URL: http://localhost:8080/rest/17
// Method: DELETE
app.delete('/rest/:id', function (req, res) {
    var url= 'mongodb://localhost:27017/test';  // test ist der Name der DB
    MongoClient.connect(url, function(err,db){
        var collection=db.collection('persons');
        toDelete={id:0};
        toDelete.id = parseInt(req.params.id);
        collection.remove(toDelete,
                          {safe: true},
                          function (err,res) {
                            if (err) throw err;
                          });
        console.log('person '+toDelete.id+' deleted')
        res.send('person '+toDelete.id+' deleted');
        db.close();
    });
});

// Einfügen einer Person
// URL: http://localhost:8080/rest/17
// Method: POST
// Header: Content-Type: application/json
// Body-Beispiel: {"id":17,"gender":"F","firstname":"AAA","lastname":"EEE","email":"phughesg@theatlanti
app.post('/rest/:id', function (req, res) {
    var url= 'mongodb://localhost:27017/test';  // test ist der Name der DB
    MongoClient.connect(url, function(err,db){
        var collection=db.collection('persons');
        console.log(req.body);
        toInsert=req.body;
        toInsert.id = parseInt(req.params.id);
        collection.insert(toInsert,
                          {safe: true},
                          function (err,res) {
                            if (err) throw err;
                          });
        console.log('person '+toInsert.id+' eingefuegt')
        res.send('person '+toInsert.id+' eingefuegt');
        db.close();
    });
});
```

```
// Updaten einer Person
// URL: http://localhost:8080/rest/17
// Method: PUT
// Header: Content-Type: application/json
// Body-Beispiel: {"id":17,"gender":"F","firstname":"AAA","lastname":"EEE","email":"phughesg@theatlanti
app.put('/rest/:id', function (req, res) {
    var url= 'mongodb://localhost:27017/test';  // test ist der Name der DB
    MongoClient.connect(url, function(err,db){
        var collection=db.collection('persons');
        console.log(req.body);
        toUpdate={};
        toUpdate.id = parseInt(req.params.id);
        var newPerson=req.body;
        collection.update(toUpdate,
                          {$set: newPerson},
                          {safe: true},
                          function (err,res) {
                            if (err) throw err;
                          });
        console.log('person '+toUpdate.id+' geändert');
        res.send('person '+toUpdate.id+' geändert');
        db.close();
    });
});

app.listen(8080, function () {
  console.log('Server listening on port 8080')
});
```

Einfügen von Testdaten in die MongoDB

File: persons2MongoDB.txt

```
db.persons.insert({"id":1,"gender":"M","firstname":"Fred","lastname":"Perez","email":"fperez0@google.co
db.persons.insert({"id":2,"gender":"F","firstname":"Phyllis","lastname":"Boyd","email":"pboyd1@youtube.
db.persons.insert({"id":3,"gender":"M","firstname":"Johnny","lastname":"Kelly","email":"jkelly2@answers
db.persons.insert({"id":5,"gender":"F","firstname":"Evelyn","lastname":"Coleman","email":"ecoleman4@fox
db.persons.insert({"id":6,"gender":"M","firstname":"Michael","lastname":"Williams","email":"mwilliams5@
db.persons.insert({"id":7,"gender":"M","firstname":"Jason","lastname":"Lewis","email":"jlewis6@referenc
db.persons.insert({"id":8,"gender":"F","firstname":"Judy","lastname":"Moore","email":"jmoore7@bravesite
db.persons.insert({"id":9,"gender":"M","firstname":"Gregory","lastname":"Green","email":"ggreen8@over-b
db.persons.insert({"id":10,"gender":"F","firstname":"Andrea","lastname":"Howard","email":"ahoward9@jiat
db.persons.insert({"id":11,"gender":"M","firstname":"Frank","lastname":"Williams","email":"fwilliamsa@y
db.persons.insert({"id":12,"gender":"M","firstname":"Gregory","lastname":"Garcia","email":"ggarciab@squ
db.persons.insert({"id":13,"gender":"M","firstname":"Philip","lastname":"Hill","email":"phillc@bbb.org"
db.persons.insert({"id":14,"gender":"M","firstname":"Keith","lastname":"Tucker","email":"ktuckerd@sohu.
db.persons.insert({"id":15,"gender":"F","firstname":"Marie","lastname":"Patterson","email":"mpattersone
db.persons.insert({"id":16,"gender":"M","firstname":"Gerald","lastname":"Robertson","email":"grobertson
db.persons.insert({"id":17,"gender":"F","firstname":"Phyllis","lastname":"Hughes","email":"phughesg@the
db.persons.insert({"id":18,"gender":"M","firstname":"Ronald","lastname":"Morris","email":"rmorrish@stum
db.persons.insert({"id":19,"gender":"F","firstname":"Lois","lastname":"Chavez","email":"lchavezi@openso
db.persons.insert({"id":20,"gender":"M","firstname":"James","lastname":"Campbell","email":"jcampbellj@s
```

File: loadData.bat (Windows only)

ACHTUNG: Bei Bedarf Pfad zu mongo-Client anpassen oder in $PATH aufnehmen

```
"C:\Program Files\MongoDB 2.6 Standard\bin\mongo" persons2MongoDB.txt
```

WebSockets

Template Engines

Arbeiten mit Dateien

Streams

Testen