Node.js Einführung

Manuel Hart

FOSS@HFT Stuttgart

Adapted and extended by Daniel Benninger
for the KETE Module

Version 1 / September 2021

Inhalt

1. Grundlagen
2. Serverseitiges JavaScript
3. Express.js
4. Websockets
5. Kleines Projekt
   (Real-Time Chat Application)

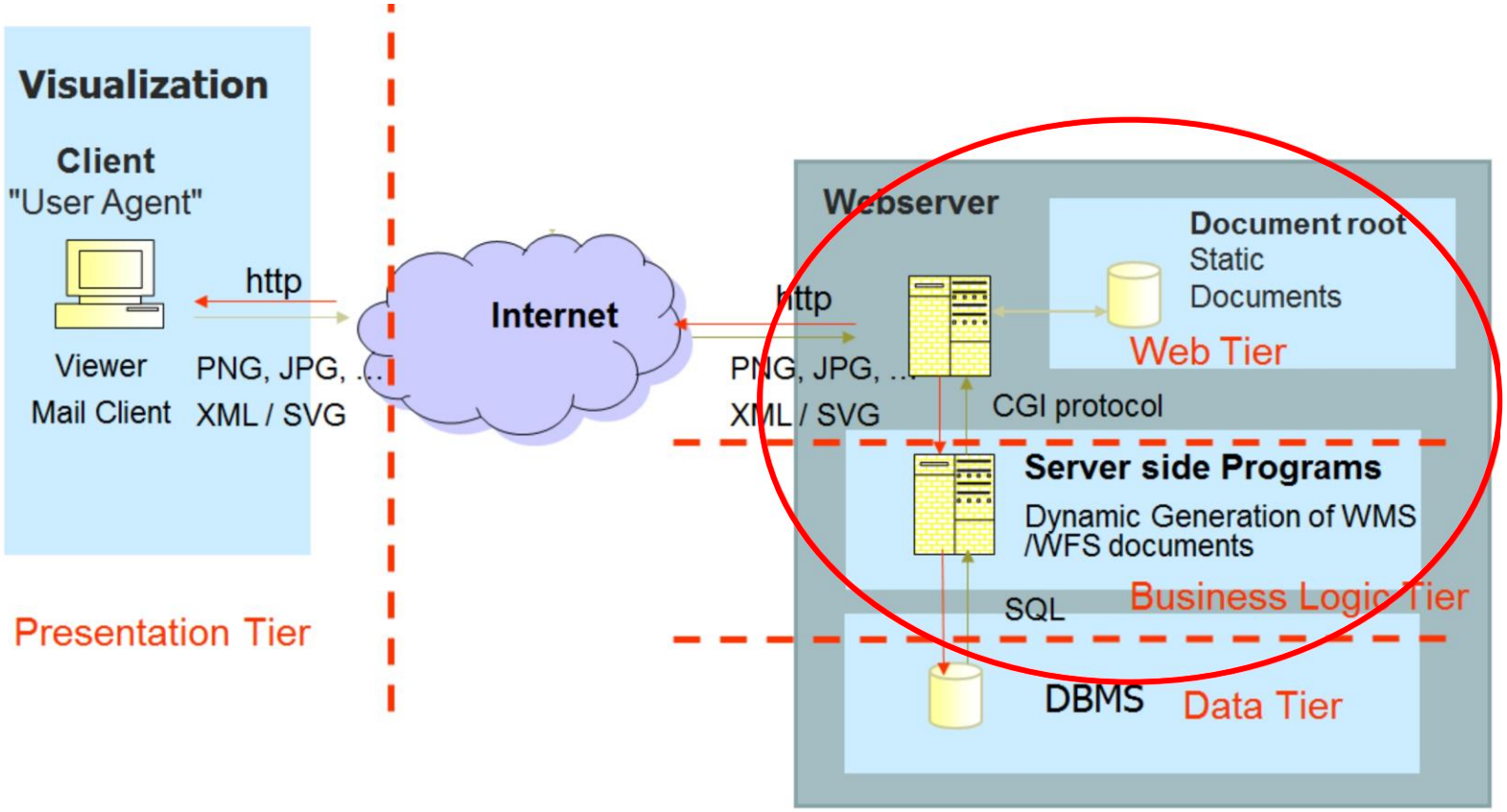
1. Grundlagen

„Node.js is a JavaScript runtime [environment] built on Chrome's V8 JavaScript engine.“ [nodejs.org]

→ Also eine Software die JavaScript Code verstehen und ausführen kann.

1. Grundlagen

Orientierung



Source: F.J.Behr

1. Grundlagen

Lizenz

Node.js ist **Open Source** und unter der **MIT-Lizenz**

(*Massachusetts Institute of Technology*) lizensiert.

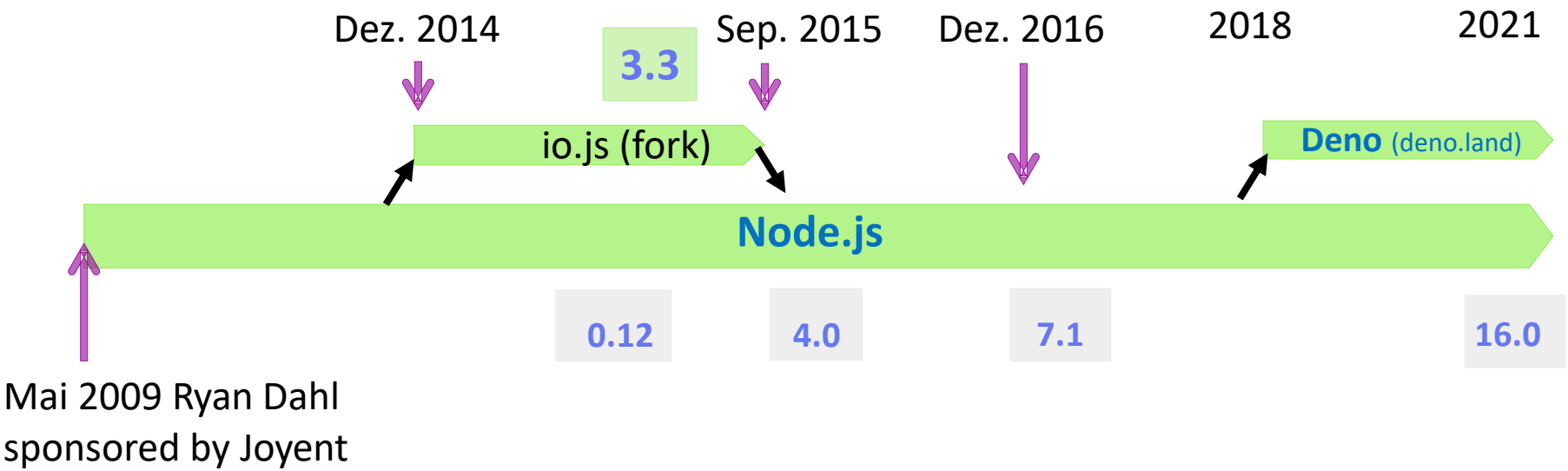
(https://raw.githubusercontent.com/nodejs/node/master/LICENSE)

https://github.com/nodejs/node

https://nodejs.org

1. Grundlagen

Historisches



1. Grundlagen

JavaScript

JavaScript ist eine **objektorientierte Skriptsprache** die von allen modernen Web-Browsern verstanden und ausgeführt werden kann.

1. Grundlagen

JavaScript

HTML DOM Element

```
<div id="demo"></div>

<script>
   var demo = document.getElementById("demo");
</script>
```

javaScript

Get the DOM Element

1. Grundlagen
Vergleich mit php

PHP
- Typisches Client-Server Modell
- Routing wird vom Webserver gesteuert
- Blockierend
- Sehr weit verbreitet

Node.js
- Sockets, HTTP,…
- Routing wird von der Node Anwendung gesteuert
- Nicht Blockierend

1. Grundlagen

Installation

1. Download von ***https://nodejs.org/en/download/***

- Windows Installer
- Macintosh Installer
- Linux Binaries (x86/x64 & ARM)

2. Installieren

3. (PATH Variable setzen)

1. Grundlagen

Build-in Module

node.js ist ideal für **Netzwerkanwendungen**.

Integrierte **Module** für:

- File System
- HTTP(S)
- OS
- Path
- URL
- …

1. Grundlagen

npm

Weitere externe Module können über **npm** (node package manager) geladen werden. (Artistic License 2.0)

```
npm install packagename --save
```

~ 250.000 open source packages

--g installiert Global

1. Grundlagen

npm – package.json

Enthält Informationen über das node.js Programm und die erforderlichen Abhängigkeiten.

```
1  {
2    "name": "basic_server",
3    "version": "1.0.0",
4    "description": "a basic node.js server for the FOSS@HFT group",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Manuel Hart",
10   "license": "null",
11   "dependencies": {
12     "express": "^4.14.0",
13     "mathjs": "^3.7.0",
14     "moment": "^2.16.0",
15     "path": "^0.12.7",
16     "socket.io": "^1.5.1"
17   }
18 }
```

1. Grundlagen
Erweiterungen

- Express.js / Socket.io
- moment.js
- bower
- jsdoc
- nodemon
- Yeoman
- proj4js

2. Serverseitiges JavaScript
Übersicht

- Einfachste Node.js Anwendung
- Was wollen wir eigentlich?
- Node.js Server Anwendung
- Debugging

2. Serverseitiges JavaScript

Starten des Servers

Node.js wird über die Kommandozeile ausgeführt.

1. Schritt Prüfen ob **Node.js** korrekt installiert wurden.

```
node -v
```

2. Schritt Prüfen ob **npm** korrekt installiert wurden.

```
npm -v
```

2. Serverseitiges JavaScript

Die erste Applikation

Erste Node.js Applikation erstellen.

```
//Dies ist unsere erste Applikation (Programm)
console.log('running');
```

Speichern als app.js.

2. Serverseitiges JavaScript

Die erste Applikation

Erste Node.js Applikation starten.

```
node app.js
```

start

Kommandozeile:

```
G:\nodeJS_atHFT>node app.js
running

G:\nodeJS_atHFT>_
```

ausgeführt

stop

2. Serverseitiges JavaScript

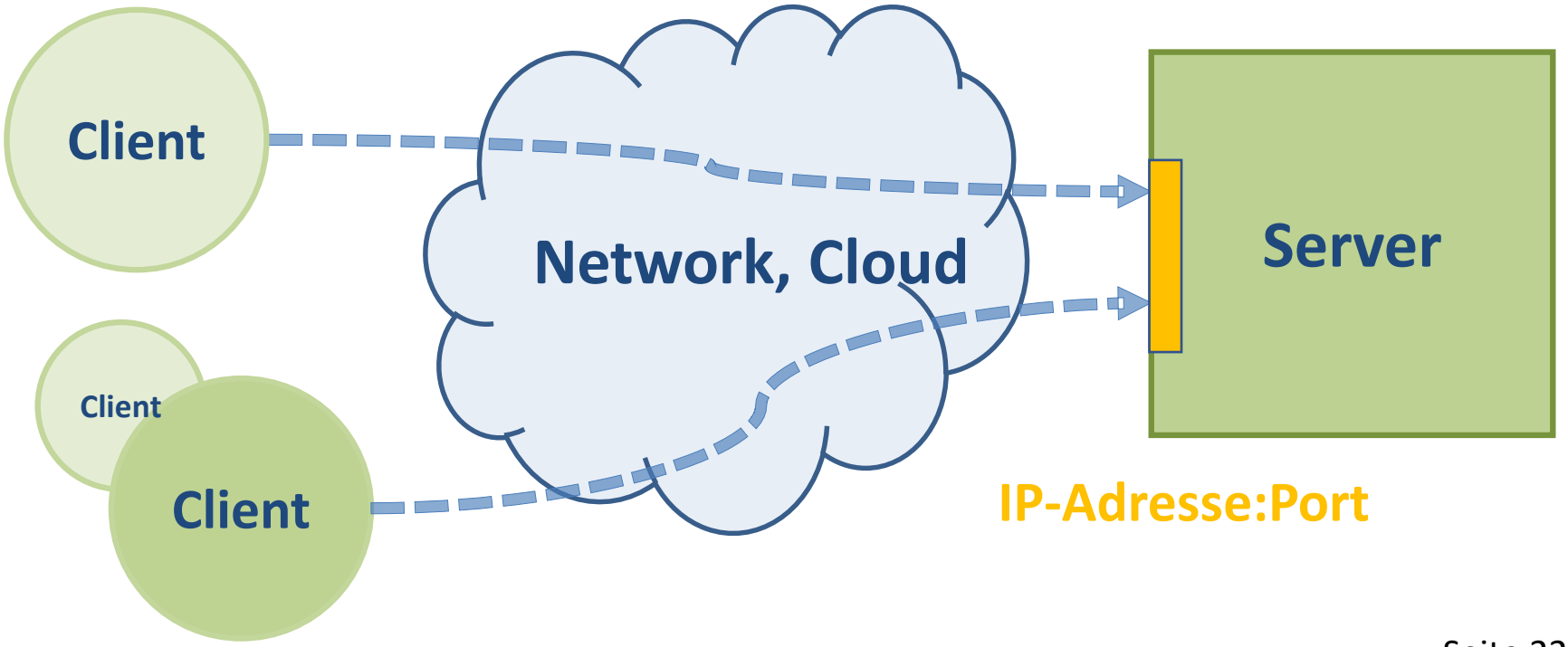
Was wollen wir eigentlich?

- Daten bereitstellen
- Website bereitstellen
- Dienste(Services) bereitstellen
- Web-Applikation bereitstellen

Anwendungsfall	Software (Beispiel)
Daten bereitstellen	Apache (\htdocs) / php (Datenbank)
Website bereitstellen	Apache (\htdocs)
Dienste(Services) bereitstellen	Tomcat → Java Servlet / php

2. Serverseitiges JavaScript

Server Erreichbarkeit



2. Serverseitiges JavaScript

Ein Server der „weiterläuft“.

Server soll auf Anfragen reagieren können.

JavaScript – Ereignis gesteuert.

```
server.listen(3001, function(){
     console.log("Server listening on: http://localhost:„ + port);
});
```

2. Serverseitiges JavaScript

Erste Server-Applikation

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

//port
const port=3001;

//request handle function
function handleRequest(request, response){
  response.end('request URL: http://localhost:'+ port  + request.url);
}

//create the server
var server = http.createServer(handleRequest);

//start the server
server.listen(port, function(){
  console.log("Server listening on: http://localhost:" + port);
});
```

Server-Port

Request Funktion

Create Server

Start Server

2. Serverseitiges JavaScript

Zugriff auf Server

Web-Browser:

```
http://localhost:3001/Hallo
```

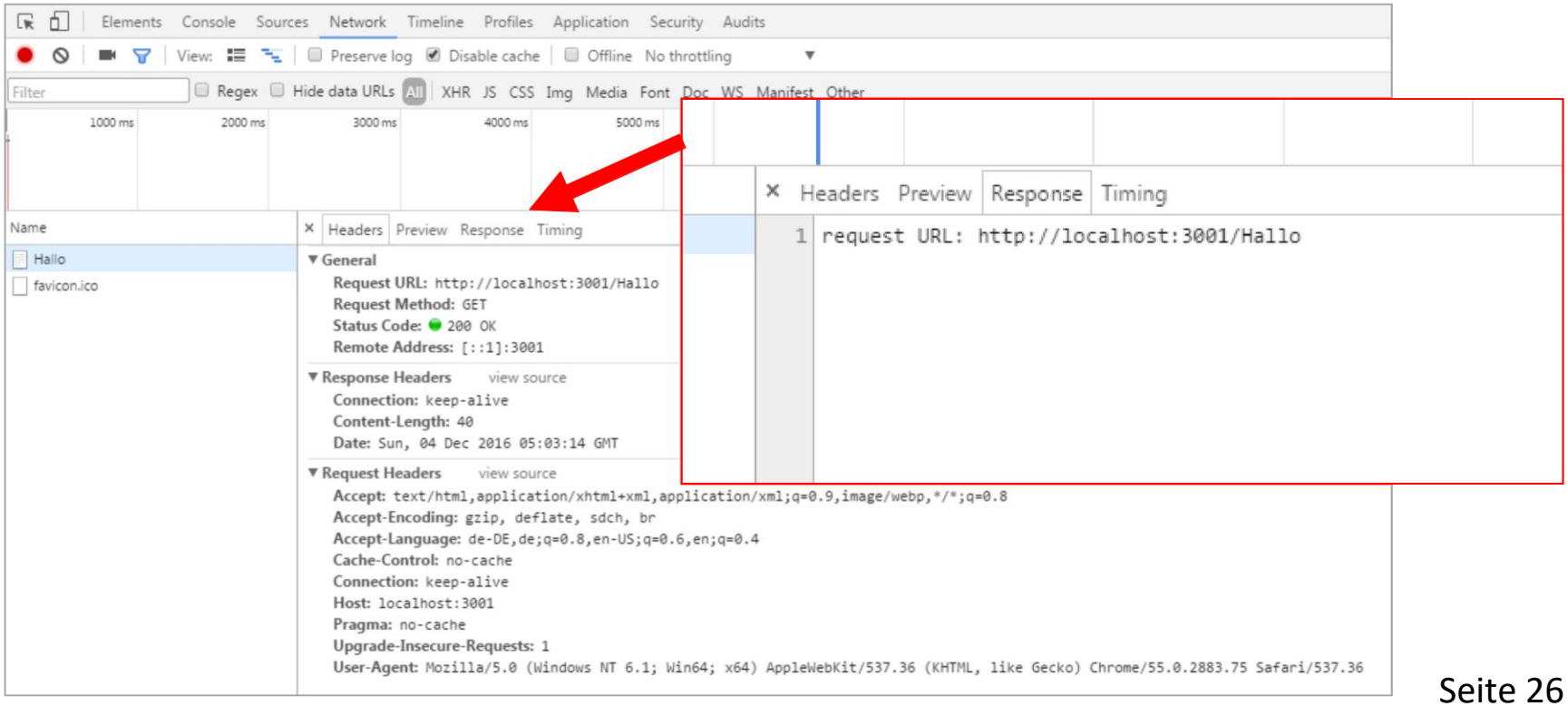
Ausgabe (Browser):

```
Request URL: http://localhost:3001/Hallo
```

2. Serverseitiges JavaScript

Request auf Clientseite

Web-Browser Debugging(F12):



2. Serverseitiges JavaScript

Debugging

Fehler im Code finden und beheben.

```
1   var express = require('express');
2   var app = express();
3
4   var t = gg.help();
5
6   app.get('/', function (req, res) {
7
8     res.send('Hello World!')
9   })
10
11  app.listen(3000, function () {
12    console.log('Example app listening on port 3000!')
13  })
```

```
C:\Users\Manu\Desktop\nodeJS_atHFT>node express_basic.js
C:\Users\Manu\Desktop\nodeJS_atHFT\express_basic.js:4
var t = gg.help();
          ^

ReferenceError: gg is not defined
    at Object.<anonymous> (C:\Users\Manu\Desktop\nodeJS_atHFT\express_basic.js:4:12)
    at Module._compile (module.js:570:32)
    at Object.Module._extensions..js (module.js:579:10)
    at Module.load (module.js:487:32)
    at tryModuleLoad (module.js:446:12)
    at Function.Module._load (module.js:438:3)
    at Module.runMain (module.js:604:10)
    at run (bootstrap_node.js:394:7)
    at startup (bootstrap_node.js:149:9)
    at bootstrap_node.js:509:3

C:\Users\Manu\Desktop\nodeJS_atHFT>
```

2. Serverseitiges JavaScript

Debugging

```
node debug app.js
```

```
C:\Users\Manu\Desktop\nodeJS_atHFT>node debug express_basic.js
< Debugger listening on [::]:5858
connecting to 127.0.0.1:5858 ... ok
break in C:\Users\Manu\Desktop\nodeJS_atHFT\express_basic.js:1
> 1 var express = require('express')
  2 var app = express()
  3
next
break in C:\Users\Manu\Desktop\nodeJS_atHFT\express_basic.js:2
  1 var express = require('express')
> 2 var app = express()
  3
  4 app.get('/', function (req, res) {
```

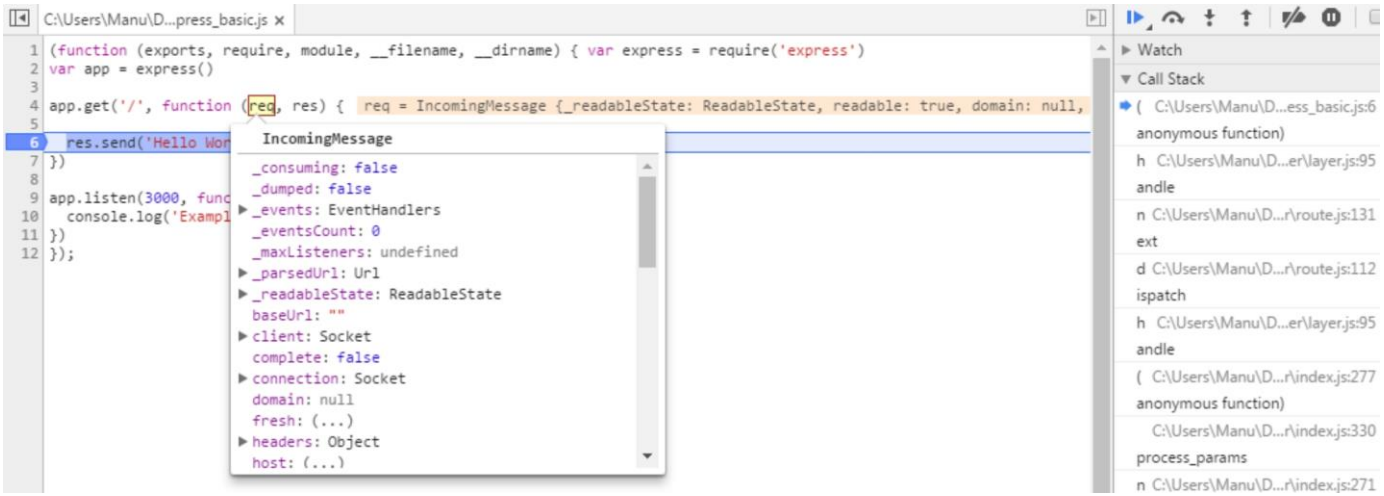
```
next
cont → debugger;
help
```

2. Serverseitiges JavaScript

Debugging

```
node --inspect app.js
```

```
C:\Users\Manu\Desktop\nodeJS_atHFT>node --inspect  express_basic.js
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/remote/serve_file/@60cd6e859b9f557d2312f5bf532f6aec5f284980/inspector.html?experiments=tr
ue&v8only=true&ws=localhost:9229/a2c27c60-f508-4a3e-aff3-2bf7c3f03359
Example app listening on port 3000!
Debugger attached.
```



2. Serverseitiges JavaScript

Erweiterungen

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

//port
const port=3001;

//request handle function
function handleRequest(request, response){
  response.end('request URL: http://localhost:'+ port  + request.url);
}

//create the server
var server = http.createServer(handleRequest);

//start the server
server.listen(port, function(){
  console.log("Server listening on: http://localhost:" + port);
});
```

```
var server = require('server');
server.start(3001, function(request, response){…});
```

3. Express.js

Das Framework - Übersicht

express

„Schnelles, offenes, unkompliziertes Web-Framework für Node.js“
http://expressjs.com/de/ (MIT Lizenz)

Anwendungsszenarien (exempl.):

1. Einfacher Server
2. HTTP Methods
3. Router
4. Middlewares
5. Statische Daten und Ordner

3. Express.js

Einfacher Server

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

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

app.listen(3000, function () {
  console.log('Example app listening on port 3000!')
})
```

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

//port
const port=3000;

//request handle function
function handleRequest(request, response){
  console.log('Hello World!');
}

//create the server
var server = http.createServer(handleRequest);

//start the server
server.listen(port, function(){
  console.log("Server listening on: http://localhost:" +
});
```

3. Express.js
HTTP Methods

HTTP GET und POST + Routing

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

app.get('/getRoute', function (req, res) {
  res.send('GET-Request');
});

app.post('/postRoute', function (req, res) {
  res.send('POST-Request');
});

app.listen(3000, function () {
  console.log('app listening on port 3000!')
});
```

HTTP-GET

HTTP-POST

3. Express.js

Router

```
1   var express = require('express');
2   var app = express();
3
4   var router1 = express.Router();
5   var router2 = express.Router();
6
7   app.use('/Route1', router1);
8   app.use('/Route2', router2);
9
10  router1.get('/', function (req, res) {
11    res.send('Router1');
12  });
13
14  router2.get('/', function (req, res) {
15    res.send('Router2');
16  });
17
18  router2.get('/Anfrage', function (req, res) {
19    res.send('Anfrageergebnis von Router2');
20  });
21
22  app.listen(3000, function () {
23    console.log('app listening on port 3000!')
24  });
```

Router Objekte

URL Einstellung

Route1/

Route2/

Route2/Anfrage

3. Express.js
Middlewares

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

let user = 'Manuel';
let password = '1990'

function middleHandler(req, res, next) {
    console.log("execute middle ware");
    next();
}

app.use(function (req, res, next) {
    //Authorization test

    let query_user = req.query.user;
    let query_pw = req.query.pw;

    if(query_user === user && query_pw === password){
      next();
    }else{
      res.send('Zugriff verweigert');
    }
});

app.get('/', middleHandler, function (req, res) {
    res.send("Zugriff gestattet");
});

app.listen(3002);
console.log('start server on 3002');
```

Middleware Für „/“

Middleware Für Alle

GET Request

3. Express.js

Statische Daten

Um komplette Verzeichnisse bereitzustellen.

```
1
2
3   var express = require('express');
4   var path = require('path');
5   var app = express();
6
7   //make directory 'public' public.
8   app.use('/public', express.static(path.join(__dirname, 'public')));
9
10  app.listen(3000, function () {
11    console.log('Example app listening on port 3000!')
12  })
```

4. Websockets

Bi-direktionale Web-Verbindung

Der Server kann dem Client Nachrichten senden sobald eine Socket-Verbindung besteht.

Framework: **Socket.io**

4. Websockets

socket.io

Web-Socket Framework (MIT Lizenz) Bestehend aus eine Server Komponente und einer Client Komponente.

4. Websockets

socket.io – Beispiel

Auf Server Seite

```
1   var express = require('express');
2   var app = express();
3   var server = require('http').Server(app);
4   var io = require('socket.io')(server);
5   var colors = require('colors/safe');
6
7
8
9   app.use('/',express.static(__dirname + '/public/socket'));
10  app.use('/src',express.static(__dirname + '/public/src'));
11
12  io.of('/socket').on('connection', function(socket){
13    console.log('connected');
14    socket.on('chat message', function(msg){
15      console.log('received: ' + msg);
16      io.of('/socket').emit('chat message', msg);
17    });
18  });
19
20  server.listen(3001);
21  console.log('Server is running on port 3001');
```

4. Websockets

socket.io – Beispiel

Auf Client Seite

```html
<!doctype html>
<html>
  <body>

    <form action="">
      <input id="m" autocomplete="off" /><button>Send</button>
    </form>

    <ul id="liste"></ul>

    <script src="src/socket.io.js"></script>
    <script src="src/jquery-3.1.1.js"></script>

    <script>

      var socket = io('http://localhost:3001/socket');

      $('form').submit(function(){
        socket.emit('chat message', $('#m').val());
        $('#m').val('');
        return false;
      });

      socket.on('chat message', function(msg){
        $('#liste').append('<li>'+msg+'</li>');
      });

    </script>
  </body>
</html>
```

5. Kleines Projekt

Real-Time Chat Application Using Socket.io in Node.js

Souvik Paul Follow
Oct 13, 2020 · 5 min read

https://medium.com/swlh/real-time-chat-application-using-socket-io-in-node-js-37806e98918c

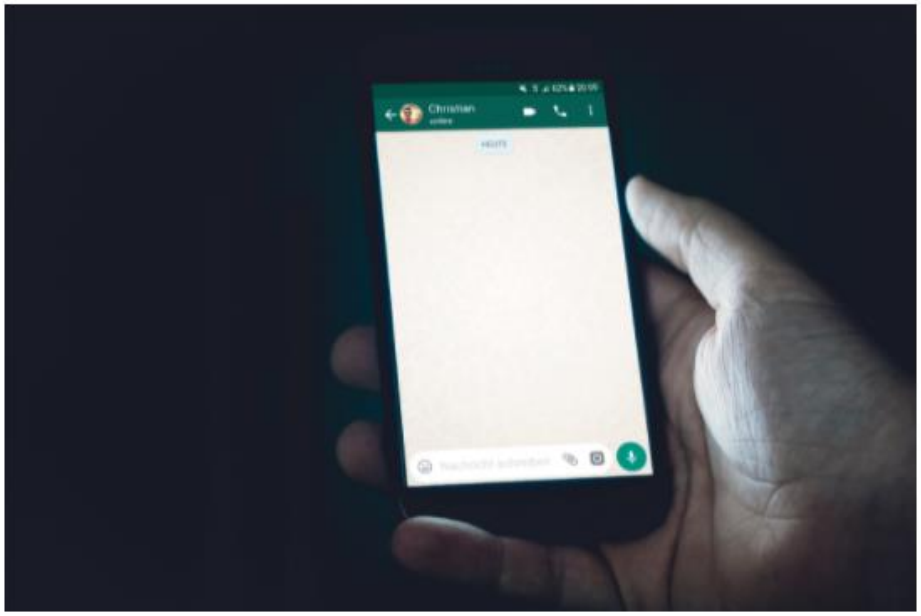


Photo by Christian Wiediger on Unsplash

'The expert at anything was once a beginner' -Helen Hayes.

5. Kleines Projekt

Real-Time Chat Application

https://youtu.be/1iQGoenm0ug

This entire project is also available in the github repository of Paul Souvik (https://github.com/souvik-pl/chatRoom)