

HTTP Module

Das Node.js HTTP Module ermöglicht das Bauen eines Webserver. Die Daten zwischen Client und Server werden dabei über das HTTP Protokoll übertragen.

My First Webserver

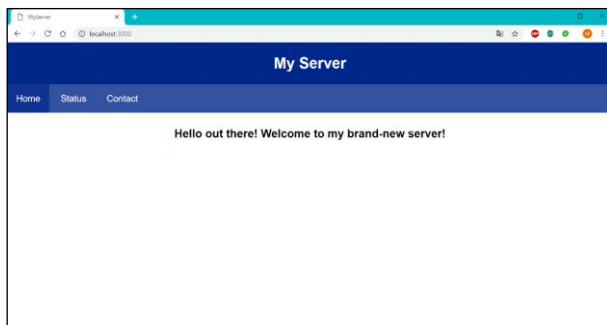
Ziel dieser Übung ist es, einen Node.js Server zu hosten, der über *localhost* erreichbar ist. Als Zugabe soll der Node.js Server über die IP-Adresse von anderen Computern aus erreichbar sein.

Anforderungen:

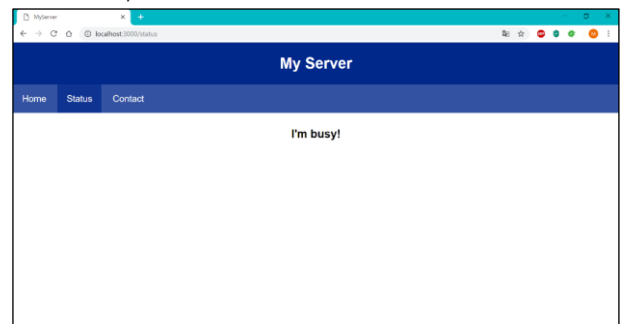
- Ein Node.js Server, erreichbar über Localhost @Port 3000
- 3 Seiten, die jeweils einen kurzen Text anzeigen:
index.html | contact.html | status.html
- Navigation zu allen Seiten
- Schönes Front-End (etwas CSS-Code im externen Stylesheet)

Screenshots einer möglichen Lösung:

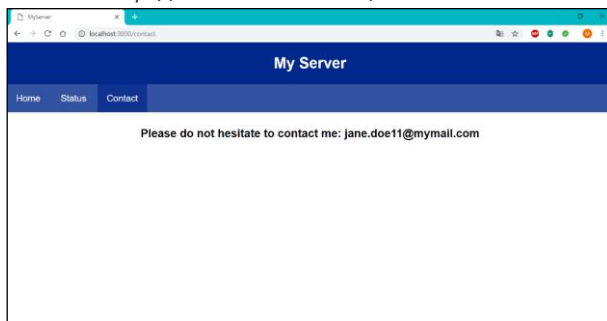
http://localhost:3000



http://localhost:3000/status



http://localhost:3000/contact



Tutorial 1: Server anlegen

Legen Sie ein neues WebStorm-Projekt *MyFirstServer* an und erstellen Sie darin die JavaScript-Datei *server.js*.

Zuerst müssen Sie das HTTP Module einbinden:

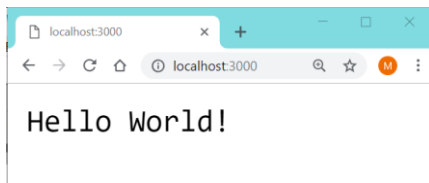
```
let http = require('http');
```



Nun können Sie einen Server definieren:

```
http.createServer(function (req, res) {  
  res.write('Hello World!');  
  res.end();  
}).listen(3000);
```

Testen Sie den Server!

- das Programm in WebStorm starten Run -> Run 'server.js'
- Browser öffnen und localhost aufrufen http://localhost:3000



i Jedes Mal, wenn die JavaScript Datei *server.js* verändert wird (neue Codeteile oder Code wurde gelöscht), muss das Programm in WebStorm terminiert  und erneut gestartet werden .

Tutorial 2: Simple Routing

Jede Webseite besteht aus mehreren Seiten, jede App setzt sich aus mehreren Views zusammen. Man benötigt die Möglichkeit, auf eine Anfrage an eine bestimmte Seite weiterzuleiten (Routing).

Das **req Objekt** beinhaltet alle Infos zum aktuellen Request. Mit **req.url** lässt sich die URL als String auslesen. Durch simple Verzweigungen können Sie dadurch verschiedene Antworten (mit dem **res.write Objekt**) geben:

```
http.createServer(function (req, res) {  
  
  if (req.url === '/') {  
    res.write('Hello out there! My Name is Jane Doe!');  
    res.end();  
  }  
  
  if (req.url === '/status') {  
    res.write('I\'m busy.');    res.end();  
  }  
  
  if (req.url === '/contact') {  
    res.write('Please do not hesitate to contact me: jane.doe1@mymail.com');  
    res.end();  
  }  
  
}).listen(3000);
```

Testen Sie den Server!

- das Programm in WebStorm starten
- im Browser die Startseite aufrufen
- im Browser die Unterseite aufrufen
- im Browser die Unterseite aufrufen

Run -> Run 'server.js' (falls es noch läuft vorher stoppen!)
http://localhost:3000
http://localhost:3000/status
http://localhost:3000/contact

Tutorial 3: Routing to HTML File

In der Praxis wird man nie einen einfachen String als Antwort retournieren. Um stattdessen ein HTML zu senden, benötigen wir wieder das ReadFile Module:

Zuerst müssen Sie das ReadFile Module einbinden:

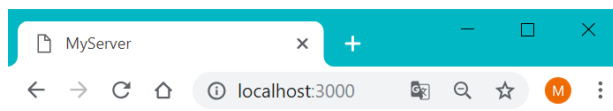
```
let fs = require('fs');
```

Nun können Sie die erste Verzweigung so adaptieren, dass anstelle eines Strings die ausgelesenen Daten *data* aus der übergebenen Datei *index.html* gesendet werden.

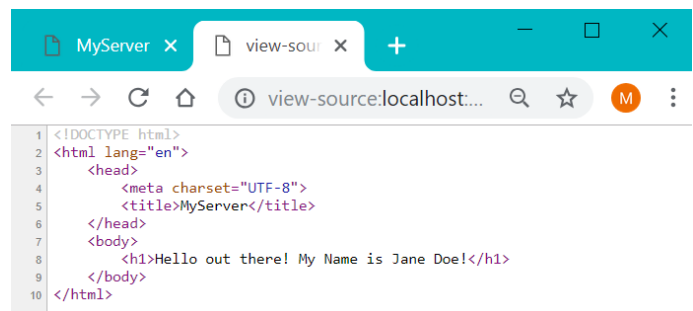
```
if (req.url === '/') {  
  fs.readFile('html/index.html', function (err, data) {  
    res.write(data);  
    res.end();  
  });  
}
```

Erstellen Sie nun im Unterordner *html* die drei HTML Files *index.html*, *contact.html* und *status.html* mit dem HTML Grundgerüst und den anzuzeigenden Texten als *<h1>* Überschrift.

Verändern Sie alle Requests, sodass auf das entsprechende HTML File weitergeleitet wird.



Hello out there! My Name is Jane Doe!



Testen Sie wieder ihre Seiten!

Zur Kontrolle können Sie sich im Browser die wirklich gesendeten Daten anzeigen lassen:

Rechtsklick -> Seitenquelltext anzeigen

Hier soll nun ein valides HTML-Dokument erscheinen.



Wenn Sie nur die Veränderungen einer HTML-Datei im Browser sehen möchten (z.B. nach neuen Codeteilen in einer .html Datei), muss das Programm in WebStorm nicht terminiert werden.

Einfach das Browserfenster aktualisieren (F5), dadurch wird die aktuelle neue HTML-Datei angefordert.

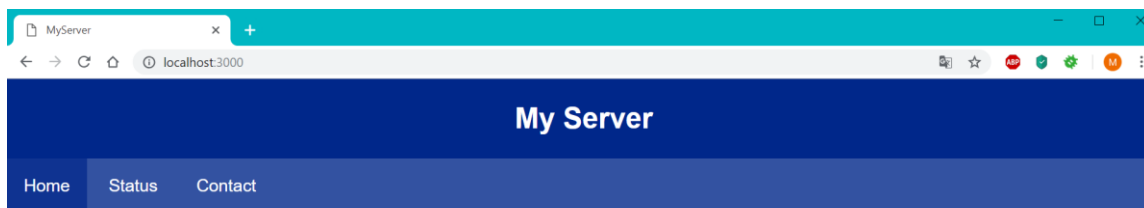
Was jetzt noch fehlt ist eine Navigation für den User und etwas Kosmetik für das Auge.

So könnte ein valides HTML-File mit Header, Navigation und Main-Content aussehen:

```
<body>
  <header>
    <h1>My Server</h1>
  </header>
  <nav>
    <a class="active" href="http://localhost:3000">Home</a>
    <a href="http://localhost:3000/status">Status</a>
    <a href="http://localhost:3000/contact">Contact</a>
  </nav>
  <main>
    <h2>Hello out there! Welcome to my brand-new server!</h2>
  </main>
</body>
```

Im <head> Bereich brauchen Sie nur mehr einen <style> Bereich für das interne Stylesheet definieren und etwas CSS einfügen und schon nimmt das Front-End Gestalt an. Beispiel:

```
<style>
  body {
    padding: 0;
    margin: 0;
    font-family: sans-serif;
  }
  header {
    background-color: rgba(0,39,138,1);
  }
  header h1 {
    margin: 0;
    padding: 2%;
    color: #fff;
    text-align: center;
    font-size: 2rem;
  }
  nav {
    background-color: rgba(0,39,138,0.8);
  }
  nav a {
    display: inline-block;
    padding: 1.5%;
    color: #fff;
    text-decoration: none;
    font-size: 1.2rem;
  }
  nav a:hover, nav a.active {
    background-color: rgba(0,39,138,0.7);
  }
  main {
    padding: 0 10%;
  }
  main h2 {
    margin: 3% 0;
    text-align: center;
    font-size: 1.5rem;
  }
</style>
```



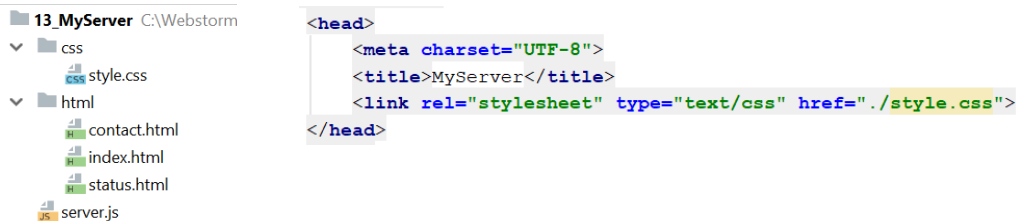
Hello out there! Welcome to my brand-new server!

Tutorial 4: Routing to CSS File

Als professioneller Webentwickler verwendet man eigentlich immer ein externes Stylesheet (Redundanzen vermeiden, Codereduzierung, bessere Wartbarkeit, ...).

Versuchen Sie, ein externes Stylesheet einzubetten:

- erstellen Sie im Unterordner `css` eine neue Datei `style.css`
- kopieren Sie den Code vom internen Stylesheet in das externe Stylesheet
- löschen Sie das interne Stylesheet
- binden Sie das externe Stylesheet in allen HTML-Files ein



Testen Sie das Ergebnis in Ihrem Browser – leider wird es nicht funktionieren, der Browser lädt ewig!



Der Grund: Sie haben die CSS-Datei (noch) nicht freigegeben, der Browser darf nicht darauf zugreifen.

Wie kann man den Zugriff erlauben? Genauso, wie man auch an HTML-Files weiterleitet: über eine bestimmte Request-URL.

Sie müssen also die JavaScript Datei `server.js` um eine weitere Verzweigung erweitern, die die CSS-Datei liefert:

```
if (req.url === '/style.css') {
  fs.readFile('css/style.css', function (err, data) {
    res.writeHead(200, { 'Content-Type': 'text/css' });
    res.write(data);
    res.end();
  });
}
```

Testen Sie erneut das Ergebnis in Ihrem Browser (in WebStorm Programm terminieren und erneut starten nicht vergessen).

Die Seite muss nun vollständig mit externem Stylesheet geladen werden!



res.writeHead(...) teilt dem Browser mit, welche Daten er erwarten kann und wie er sie interpretieren soll.

Wird eine HTML-Datei gesendet: `res.writeHead(200, { 'Content-Type': 'text/html' });`

Wird eine CSS-Datei gesendet: `res.writeHead(200, { 'Content-Type': 'text/css' });`

Tutorial 5: Connect from another Computer

Auf einen lokalen Server (der am eigenen Computer läuft) zuzugreifen, ist nicht das Gelbe vom Ei. Spannend wird es, wenn man mit anderen Computern (auch Smartphones!) auf den Server zugreifen möchte. Dazu muss aber lediglich die IP-Adresse des Servers konfiguriert werden.

Zuerst ist es sinnvoll, den Server in einer Variablen zu speichern (um ihn danach konfigurieren zu können):

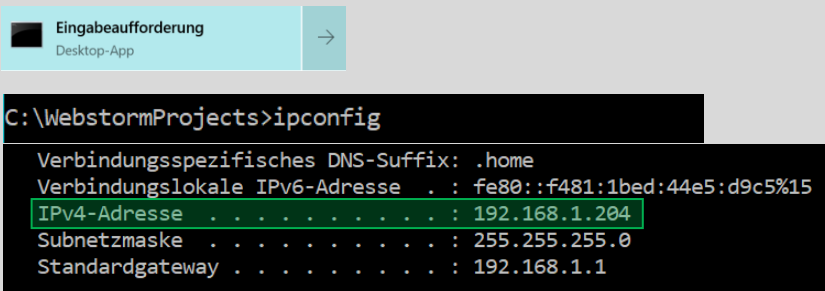
```
let server = http.createServer( requestListener: function (req, res) {  
  
    /* some code here */  
  
});
```

Nun können Sie den Server konfigurieren:

```
server.listen( port: 3000, hostname: '192.168.1.204', listeningListener: function() {  
    console.log('Access via IP 192.168.1.204');  
    console.log('Listening to port: ' + 3000);  
});
```



Der Hostname (IP-Adresse) muss die aktuelle IP-Adresse Ihres Computers sein.
In der Command Line (CMD) kann man mit dem Befehl *ipconfig* die aktuelle IP auslesen:



```
Eingabeaufforderung  
Desktop-App  
→  
  
C:\WebstormProjects>ipconfig  
  
Verbindungsspezifisches DNS-Suffix: .home  
Verbindungslokale IPv6-Adresse . : fe80::f481:1bed:44e5:d9c5%15  
IPv4-Adresse . . . . . : 192.168.1.204  
Subnetzmaske . . . . . : 255.255.255.0  
Standardgateway . . . . . : 192.168.1.1
```

Der Server ist nun von allen Geräten, die im selben lokalen Netzwerk sind, erreichbar unter der URL:
http://your_computer_ip:3000 im obigen Fall: *http://192.168.1.204:3000*

Note: falls Sie in den HTML-Dateien beim Home-Link noch einen absoluten Link haben, müssen Sie diesen durch einen relativen Link ersetzen. `Home`