pmh philipp-matthäus-hahn Technische Schule Nürtingen	**Arbeitsblatt**	Name:	pmhs Nürtingen Kanalstraße 29 72622 Nürtingen bar@pmhs.de
	Client-Server Chat mit Node.js	Datum:	

Aufgabe 1: Server einrichten

a.) Folge der Anleitung, um einen Server einzurichten und zu starten.

Am Schul-PC schon installiert

1. Node.js installieren (siehe https://nodejs.org/en/)
2. die Node.js Command Prompt öffnen (Suche hilft :-) und mit dem "cd " Befehl zum Pfad des quizFight Ordners navigieren(https://ss64.com/nt/cd.html).
3. Neues Projekt mit Node.js initialisieren und die wichtigsten Informationen angeben
   `npm init`
4. Plugins „express“ und „socket.io“ im Ordner „quizFight“ installieren mit dem Befehl
   `npm install --save express socket.io`
5. Öffne den Projektordners mit einer Programmierumgebung (z.B. VS Code).
6. Lege eine neue Datei mit dem Namen "server.js" im Hauptverzeichnis an.
7. Füge die folgenden Codezeilen hinzu und schreibe Kommentare, um dir klar zu machen was geschieht.

```
'use strict';

const http = require('http');
const express = require('express');
const socketio = require('socket.io');

let app = express();
let server = http.createServer(app);
let io = socketio(server);

io.on('connection', connected)


app.use(express.static(__dirname+'/quizFight-client'));
server.listen(8080, showReady);


function showReady(){
  console.log('Server connected!');
}

function connected(sock){
  sock.emit('msg', 'Verbindung zum Server aufgebaut');
  sock.on('msg', (txt) => io.emit('msg', txt));
}
```

Einbinden von Bibliotheken

App-Objekt mit Hilfe von express erzeugen

Ein neues HTTP-Server-Objekt für die App erzeugen

Ein Socket-Objekt für ein- und ausgehende Nachrichten erzeugen

Wenn ein Connection-Event über den Socket kommt, rufe diese Funktion auf

Der Server stellt die App im angegebenen Ordner über http Port 8080 zur Verfügung

Diese Funktion wird vom Server aufgerufen, sobald die App auf Port 8080 abgerufen werden kann.

Wenn sich ein Client verbindet wird ihm ein Socket (sock) zugeteilt. Mit `sock.emit` wird eine Nachricht über den Socket geschickt.

Wenn eine Nachricht von einem Client ankommt (msg-Event wird ausgelöst), dann wird diese über `io.emit` an alle verbundenen Sockets geschickt.

b.) Recherchiere die aufgaben der Bibliotheken http, socket.io und express im Internet. Beschreibe außerdem die Aufgabe von Node.js.

Aufgabe 2: Start und Test des Servers

a.) Folge dieser Anleitung, um den Server zu starten und eine Verbindung zu ihm aufzubauen.

1. Node.js Command Prompt öffnen und zum Pfad des Projektes wechseln.
2. Starte den Server mit folgendem Befehl: `npm start server.js`
   ✔ Sollten Fehler auftreten versuche sie zu beheben bis die Meldung "Server Connected!" erscheint.
3. Firewall-Ausnahmen müssen unbedingt zugelassen werden.
4. Öffne einen Browser und gib die IP-Adresse deines Rechners (cmd mit ipconfig /all) gefolgt von der Port-Nummer ein und öffne die App. z.B. 10.22.25.1:8080
5. Deine quizFight-client App müsste übertragen werden.
6. Du kannst den Test beenden, indem du im Node.js Command Prompt STRG+C drückst (evtl. Mehrmals notwendig)

Aufgabe 3: Chat-Client

Man kann einen einfachen Chat implementieren, da der Server alle Nachrichten die ankommen, an alle Clients weiterleitet.

a.) Übernehme den folgenden Code in die Datei client.js und kommentiere jede Zeile sinnvoll, damit du später noch verstehst was dieser Teil des Codes macht.

```
'use strict';
let sock = io();

sock.on('msg', onMessage);

function onMessage(text){
  let list = document.getElementById('chat');
  let el = document.createElement('li');
  el.innerHTML = text;
  list.appendChild(el);
}

let form = document.getElementById('chat-form');
form.addEventListener('submit', sendMessage);

function sendMessage(ev){
  var input = document.getElementById('chat-input');
  var value = input.value;
  input.value = '';

  sock.emit('msg', value);

  ev.preventDefault();
}
```

Auch hier wird ein io-Objekt von socket.io erzeugt, damit auf ein- und ausgehende Nachrichten reagiert werden kann

Msg-Event-Handler, der auf eingehende Nachrichten reagiert und die Funktion onMessage aufruft.

Was passiert hier? (Tipp: DOM wird verändert)

Event-Listener für den Senden-Button, der die Funktion sendMessage aufruft.

Was passiert hier?

Nachricht (Typ: msg, Wert: value) wird über Socket an den Server gesendet

Standard-Aktion beim abschicken eines Formulars (andere Seite laden) wird vermieden

b.) Füge in die index.html den folgenden Verweis auf die socket.io Bibliothek hinzu:

```
<script src="/socket.io/socket.io.js"></script>
```

c.) Teste den Code. Vergiss nicht den Server neu zu starten und den Browser zu aktualisieren.

d.) Lasse einen Mitschüler auf deinen Server verbinden und chatte mit ihm.

e.) Erweitere den Chat, so dass auch Spielernamen übertragen und angezeigt werden können.