

Aufgabe 1: Einführung in Git, Schlanker Chat-Client

1.1 Schlanker Chat-Client

In dieser Aufgabe soll der Client einer minimalistischen Chatanwendung weiterentwickelt werden. Aus Nutzersicht soll die Anwendung wie in Abbildung 1 aussehen. Das Interface wird in HTML beschrieben und wird durch die Datei `index.html` vorgegeben, die Sie in Ihrem Repository finden.

Der Server ist mit der Datei `server.json` als Node-RED Flow (siehe Tafelübung) vorgegeben. Laden Sie Node-RED mit `module load node-red` bevor Sie den Server mit `web-sys-red server.json` im Ordner `01_client/` im Git-Repository auf den Rechnern im CIP starten. Der Server muss nicht bei jeder Änderung neu gestartet werden, Änderungen werden automatisch neu ausgeliefert. Der Server liefert unter `localhost:1880/chat` den Inhalt der Datei `index.html` aus. Bitte überprüfen Sie dies, indem Sie `localhost:1880/chat` in Ihrem Browser öffnen.

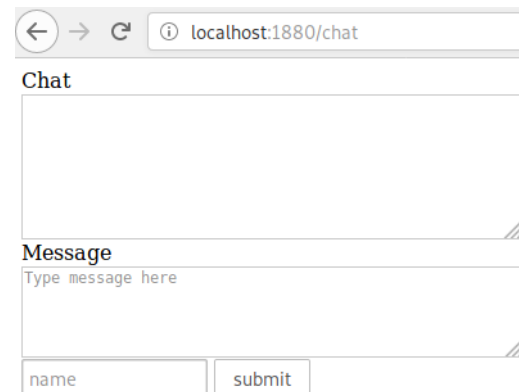


Abbildung 1: UI der Chatanwendung

Arbeiten Sie mit Git! Zum Ende der Abgabefrist muss Ihre Lösung auf dem GitLab Server im Branch `master` gepusht sein. Wir empfehlen *regelmäßige* Commits und Pushes um Versionsstände zu sichern.

1.1.1 Testen des Clients, Benutzung von Browser Development Tools

Der Client wird zwar ohne client-seitige Logik ausgeliefert, ist aber bereits funktionsfähig. Testen Sie den Client, indem Sie zwei Browsertabs oder -fenster öffnen und Nachrichten in beide Richtungen austauschen. Dabei müssen Sie die Seite mit F5 aktualisieren, um neue Nachrichten abzurufen!

- Obwohl der Client im Auslieferungszustand kein JavaScript Code enthält, werden neue Nachrichten bei dem sendenden Client automatisch angezeigt, weil der Browser die Seite neu lädt. Analysieren Sie die Informationen im *Network* Tab der Browser Developer Tools (siehe Tafelübung) um nachzuvollziehen, warum dies funktioniert! Achten Sie dabei insbesondere auf die HTTP Status Codes.

1.1.2 Asynchrone HTTP Requests

Mit dem Entwicklungskonzept Ajax (siehe Tafelübung) können Webanwendungen HTTP Requests und Responses *asynchron* versenden und empfangen. Üblicherweise wird dann der DOM-Tree der bereits dargestellten Website manipuliert um beispielsweise neue Inhalte anzuzeigen. Der Grundbaustein für Ajax ist das JavaScript-Objekt `XMLHttpRequest`¹, das von allen gängigen Browsern implementiert wird.

Implementieren Sie das folgende Verhalten, indem Sie die Datei `index.html` um JavaScript-Code erweitern, der dann an die Browser ausgeliefert wird: Der Client soll einmal pro Sekunde einen HTTP `GET`-Request mit der Location `/savedMessages` an den Server senden, um so neue Nachrichten abzurufen. Der Inhalt der Response soll in das HTML Element `chatHistory` eingefügt werden. Verwenden Sie zur Implementierung die in der Tafelübung vorgestellte `setInterval`²-Funktion (alternativ auch `setTimeout`). Nutzen sie dazu Promises, entweder mit der `async/await` oder `then` Syntax um die asynchronen Requests zu behandeln.

Zusätzliche Hinweise:

- Neben den Manpages gibt es weitere hilfreiche Dokumentation über Git: Cheat Sheet von GitHub³, Interaktiver Cheat Sheet⁴, Umfassende Git Dokumentation als E-Book⁵
- Eine gute Ressource Rund ums Web ist https://developer.mozilla.org/en-US/docs/Web#documentation_for_web_developers

¹<https://developer.mozilla.org/docs/Web/API/XMLHttpRequest>

²<https://developer.mozilla.org/docs/Web/API/WindowOrWorkerGlobalScope/setInterval>

³<https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>

⁴www.ndpsoftware.com/git-cheatsheet.html

⁵<https://git-scm.com/book/>

Letzter Commit bis zum 11. November.
Termine für die Vorstellungen in StudOn buchbar.

Präsentation der fertigen Lösung in der Woche der Abgabefrist in der Rechnerübung!