

WBE-Praktikum 10

Client-Server-Interaktion

Aufgabe 1: Spielzustand auf dem Server (Miniprojekt)

Das im letzten Praktikum implementierte Spiel «Vier gewinnt» soll nun so umgebaut werden, dass der aktuelle Zustand des Spiels auf einem Server gespeichert werden kann.

Damit es nicht zu kompliziert wird, verwenden wir als Server unseren REST Service aus Praktikum 7. Wir speichern einfach den gesamten Spielzustand (Variable *state*) auf dem Server.

Damit wir keine Probleme wegen *Cross Origin Requests* bekommen, laden wir die HTML- und CSS-Dateien vom gleichen Server, der auch die REST-API zur Verfügung stellt. Hinweise zu *Cross Origin* Problemen finden Sie am Ende dieser Praktikumsbeschreibung.

Aufgaben:

- Zunächst zum Server. Wie schon im früheren Praktikum verwenden wir *Express* für den Server. Um auch statische Dateien wie unsere HTML- und CSS-Dateien übertragen zu können, ist eine zusätzliche Middleware nötig. Ausserdem erlauben wir einen zusätzlichen API-Key *c4game*. Sehen Sie sich *index.js* mit den zugehörigen Änderungen an.
- Ergänzen Sie die Datenbank des Servers (Variable *data*) um den Anfangszustand für unser Spiel. Wählen Sie einen geeigneten Schlüssel für den Zugriff.
- Unter *public* ist bereits ein Gerüst von *connect4.html* und das Stylesheet. Ergänzen Sie die Dateien um die nötigen Scripts aus den letzten beiden Praktika.
- Installieren Sie die zusätzlich benötigten Module (*npm install*). Wenn Sie den Server nun starten (*node index.js*) sollten Sie das Spiel laden können (<http://localhost:3000/connect4.html>).
- Nun zum Client. Ergänzen Sie die Seite um zwei Buttons «Laden» und «Speichern».
- Beim Klick auf «Speichern» soll der aktuelle Spielzustand mit einem PUT-Request auf dem Server gespeichert werden. Zum Zugriff auf den Server können Sie *fetch* verwenden, welches ein Promise-Objekt zurückgibt. Das sieht etwa so aus:

```
fetch(url + "api/data/" + datakey + "?api-key=c4game", {  
  method: 'PUT',  
  headers: { 'Content-type': 'application/json' },  
  body: data  
})
```

Im *body* müssen Sie den zu einem JSON-String konvertierten Zustand übergeben.

- Beim Klick auf «Laden» wird der auf dem Server gespeicherte Spielzustand geladen und das Spielfeld entsprechend neu aufgebaut:

```
fetch(...)  
  .then(response => response.json())  
  .then(data => { /* use data */ })
```

- Das Spiel sollte nun wieder spielbar sein. Zusätzlich können Sie den Spielzustand auf dem Server speichern und – zum Beispiel in einem anderen Browser – wieder laden.

Aufgabe 2: Spiel-Ende erkennen (Miniprojekt)

Schreiben Sie eine Funktion *connect4Winner*, die nach jedem Spielzug aufgerufen wird und überprüft, ob einer der Spieler gewonnen hat. Am besten entfernen Sie dazu die anderen Steine aus dem Spiel (nach einem Zug von Rot kann nur Rot gewonnen haben), so dass die belegten Felder *truthy* und die unbelegten *falsy* sind.¹ Es gibt eine ganze Reihe verschiedener Möglichkeiten, das Erkennen des Spiel-Endes von «Vier gewinnt» zu programmieren.

Testen Sie die Funktion mit Jasmine. Erstellen Sie dazu einige Testfälle.

Integrieren Sie die Funktion in das Spiel. Wenn ein Spieler gewonnen hat, soll ein Hinweis ausgegeben und weitere Klicks blockiert werden.

Ergänzende Hinweise

Same Origin Policy

Aus Sicherheitsgründen darf ein Script normalerweise nur mit Ressourcen auf derselben Quelle interagieren, von der es selbst heruntergeladen wurde. Sonst könnte ein Script einer Website, nennen wir sie <http://malicious.attacks/you-have-won.html>, schlimmstenfalls mit anderen Seiten interagieren, in die wir gerade eingeloggt sind (Facebook, Bank). Der Server der anderen Seite kann einen solchen Zugriff jedoch zulassen, indem er mit einem *Access-Control-Allow-Origin*-Header alle oder bestimmte Quellen zulässt. Dies wird als **Cross-Origin Resource Sharing (CORS)** bezeichnet.

- Seite zu CORS des Mozilla Developer Networks:
<https://developer.mozilla.org/de/docs/Web/HTTP/CORS>

¹ Damit ist gemeint, dass ein Vergleich mit Typkonvertierung wahr oder falsch ist: `0 == false` liefert ebenfalls `true` wie `"b" == true`.