Warum WebSockets?

Bei klassisches HTTP sendet ein Client (z.B. der Browser) eine Anfrage (Request) die dann vom Server (z.B. ein Netzer) beantwortet wird (Response). Der Server kann nicht von sich aus Daten an den Client senden. Dies ist zum Beispiel dann von Nachteil, wenn Daten in unregelmäßigen Abständen aufgefrischt werden müssen. In solchen Fällen muss der Client auf gut Glück Anfragen stellen, die der Server jedesmal beantworten muss. Dies erzeugt viel unnötigen Datenverkehr.

WebSockets sind eine Möglichkeit für bidirektionale Kommunikation zwischen Server und Client. D.h. sowohl Client, als auch Server können jederzeit Daten an das Gegenüber senden. Beim Auffrischen von Daten reicht es dann aus, wenn der Server neue Daten sendet, sobald diese verfügbar sind. Der Client muss nicht ständig Anfragen stellen.

WebSocket-URI

Im Allgemeinen beginnen unverschlüsselte WebSocket-URI mit `ws://` und verschlüsselte mit `wss://`. Allerdings unterstützt der Netzer zur Zeit nur unverschlüsselte WebSocket-Verbindungen.

Die WebSocket-URI eines Netzers folgt dem Schema `ws://NETZER/ws`, wobei NETZER entweder die IP oder der mDNS-Name des Netzers ist. Die WebSocket-Funktion eines Netzers mit der IP `192.168.0.2` und dem mDNS-Namen `meinNetzer.local` ist unter `ws://192.168.0.2/ws` und `ws://meinNetzer.local/ws` erreichbar.

WebSockets in JavaScript

Verbindung öffnen

Eine neue WebSocket-Verbindung wird geöffnet indem ein neues Websocket-Objekt erzeugt wird.

```
var meinWebSocket = new WebSocket(meineWebSocketURI);
```

In manchen Firefox-Versionen (bis Version 11) heißt das WebSocket-Objekt “MozWebSocket”. Alle anderen Bezeichnungen sind gleich (aber nicht immer implementiert).

```
var meinWebSocket = new MozWebSocket(meineWebSocketURI);
```

Daten senden

WebSockets unterstützen grundsätzlich zwei Übertragungsarten: Text (UTF-8) und binär. Daten werden mit Hilfe von `send(meineDaten)` gesendet, wobei die Übertragungsart durch den Datentyp von `meineDaten` bestimmt wird.

Da der Netzer nur Text-Übertragungen benutzt, wird an dieser Stelle auch nur diese weiter besprochen.

Die Übertragungsart Text wird nur dann verwendet, wenn die and `send` übergebene Variable ein String ist.

```
var meineDaten = "Mein Text steht hier.";
meinWebSocket.send(meineDaten);
```

Daten empfangen

Das Empfangen von Daten geschieht über den Event-Handler `onmessage`. Die empfangen Daten sind in `event.data` hinterlegt.

```
meinWebSocket.onmessage = function(event) {alert("Empfangene Daten:
"+event.data);};
```

Verbindung schließen

Um die WebSocket-Verbindung zu shcließen wird, die Funktion `close()` aufgerufen. Optional kann noch ein Close-Code und eine Begründung (Reason) angegeben werden. Allerdings werden beide vom Netzer nicht weiter ausgewertet.

```
meinWebSocket.close();
```

Weitere Event-Handler

onopen

onopen wird ausgelöst, wenn eine WebSocket-Verbindung geöffnet wird.

onclose

`onclose` wird ausgelöst, wenn eine WebSocket-Verbindung geschlossen wird. Den Close-Code und die Begründung, die der Server angegeben hat, können mit `close` bzw. `reason` abgefragt werden.

```
meinWebSocket.onclose = function(event) {alert("Verbindung geschlossen.
Code: "+event.code+" Begründung: "+event.reason);};
```

onerror

“onerror” wird ausgelöst, wenn ein Fehler auftritt.

Ein einfaches Beispiel