2025/12/03 19:36 1/4 WebSocket

Warum WebSockets?

Bei klassisches HTTP sendet ein Client (z.B. der Browser) eine Anfrage (Request) die dann vom Server (z.B. einem 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://. Der Netzer unterstützt 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 (sofern sie implementiert wurden).

```
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 an 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 schließen 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.

2025/12/03 19:36 3/4 WebSocket

Ein einfaches Beispiel

Anzeigen des aktuellen Wertes von IO-0.



```
<html>
   <head>
        <meta http-equiv="content-type" content="text/html; charset=UTF-8">
        <script type="text/javascript">
            var meinWebSocket,
            // verbinden
            verbinden = function() {
                // bisherige Verbindung bereinigen
                if(meinWebSocket)
                    meinWebSocket.close();
                }
                // URL extrahieren
                url = document.getElementById("netzer url").value;
                // neue Verbindung erstellen
                if("WebSocket" in window)
                {
                    meinWebSocket = new WebSocket(url);
                else if("MozWebSocket" in window)
                    meinWebSocket = new MozWebSocket(url);
                }
                else
                    alert("Keine Unterstützung für WebSockets vorhanden.");
                    return;
                }
                // Handler für eingehende Nachrichten setzen
                meinWebSocket.onmessage = nachrichtEmpfangen;
                // wenn die Verbindung offen ist, wird ein Kommando
gesendet, das den Trigger setzt
                meinWebSocket.onopen = function()
{meinWebSocket.send("tuv0=3");};
            },
            // Nachricht empfangen
            nachrichtEmpfangen = function(event) {
                nachrichtJSON = JSON.parse(event.data);
                // Ist es ein Update des Wertes?
                if(nachrichtJSON && nachrichtJSON.u && nachrichtJSON.u.v &&
nachrichtJSON.u.v["0"])
```

```
{
                    document.getElementById("io wert").innerHTML =
nachrichtJSON.u.v["0"];
                // andere Nachrichten werden ignoriert
            },
            // Funktion für die initiale Konfiguration der Seite
            seiteEinrichten = function() {
                // Hinweis löschen
                document.getElementById("js hinweis").innerHTML = '';
                // Event-Handler setzen
                document.getElementById("anwenden button").onclick =
verbinden:
            };
        </script>
   </head>
    <body onload=seiteEinrichten()>
        <div id="js hinweis">Bitte aktivieren Sie JavaScript!</div>
        <input value="ws://test.local/ws" id="netzer url" type="url">
        <input id="anwenden button" value="Anwenden" type="button">
        <div id="io wert"></div>
   </body>
</html>
```

From:

http://mobacon.de/dokuwiki/ - MoBaCon

Permanent link:

http://mobacon.de/dokuwiki/doku.php?id=de:netzer:websockets&rev=1361545257



