Websockets, Webhooks

SEBASTIAN LIPPOLD

WEB ENGINEERING SS2021

Gliederung

Ajax

Webhooks

Websockets

Was ist das Beste?

Ajax

Asynchronorous **J**avaScript **a**nd **X**ML

Ermöglicht Datenaustausch zwischen Server und Client ohne erneutes Laden der Website

Gezieltes Übermitteln von Datenkomponenten möglich

Direkte Implementierung (nur Datenaustausch) sowie indirekte Implementierung (Übertragung von HTML-Fragmenten) möglich

POST und GET Anfragen

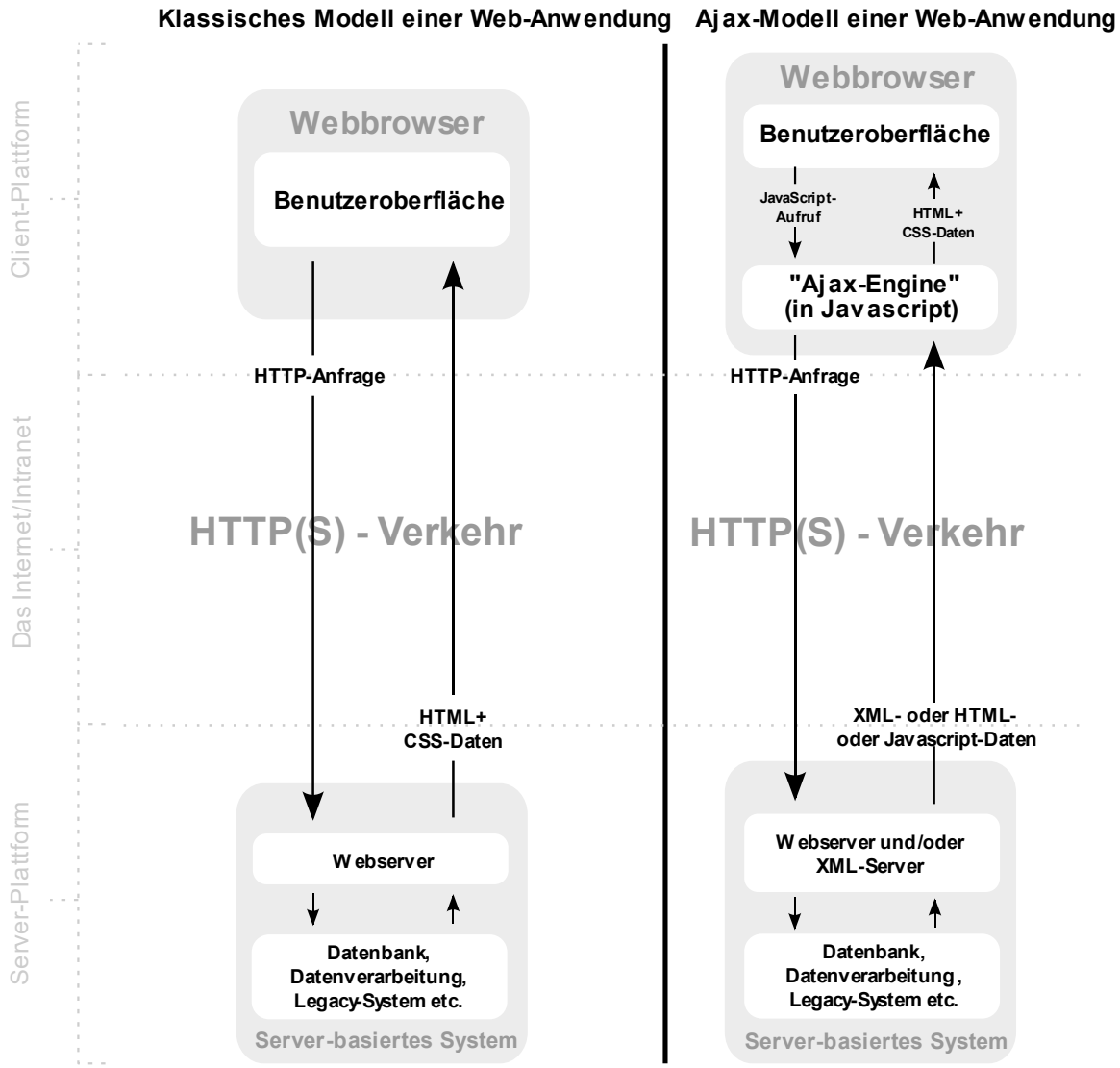
Auch synchrone Anfragen möglich (deprecated)

Ajax: Funktionsweise

https://upload.wikimedia.org/wikipedia/commons/d/d8/Ajax-vergleich.svg

Zwischengeschaltete AJAX-Engine übernimmt Kommunikation mit dem Webserver

Kombination aus beiden Modellen möglich



Ajax: Code

CLIENT (JAVASCRIPT)

```
//Neues Objekt für Ajax-Request erstellen
let r = new XMLHttpRequest();
//Objektstatusänderung erkannt
r.onreadystatechange = function ()  {
    //Antwort erfolgreich erhalten?
    if (this.readyState == 4 && this.status == 200)  {
        let msg = JSON.parse(r.responseText);
        //Antwort des Servers verarbeiten
    }
};
//Verbindung festlegen
r.open("GET", "http://url-zum-server", true);
//Request senden
r.send();
```

SERVER (PHP)

```
<?php
    header('Content-Type: application/json');
    echo json_encode(array('time' => time()));
?>
```

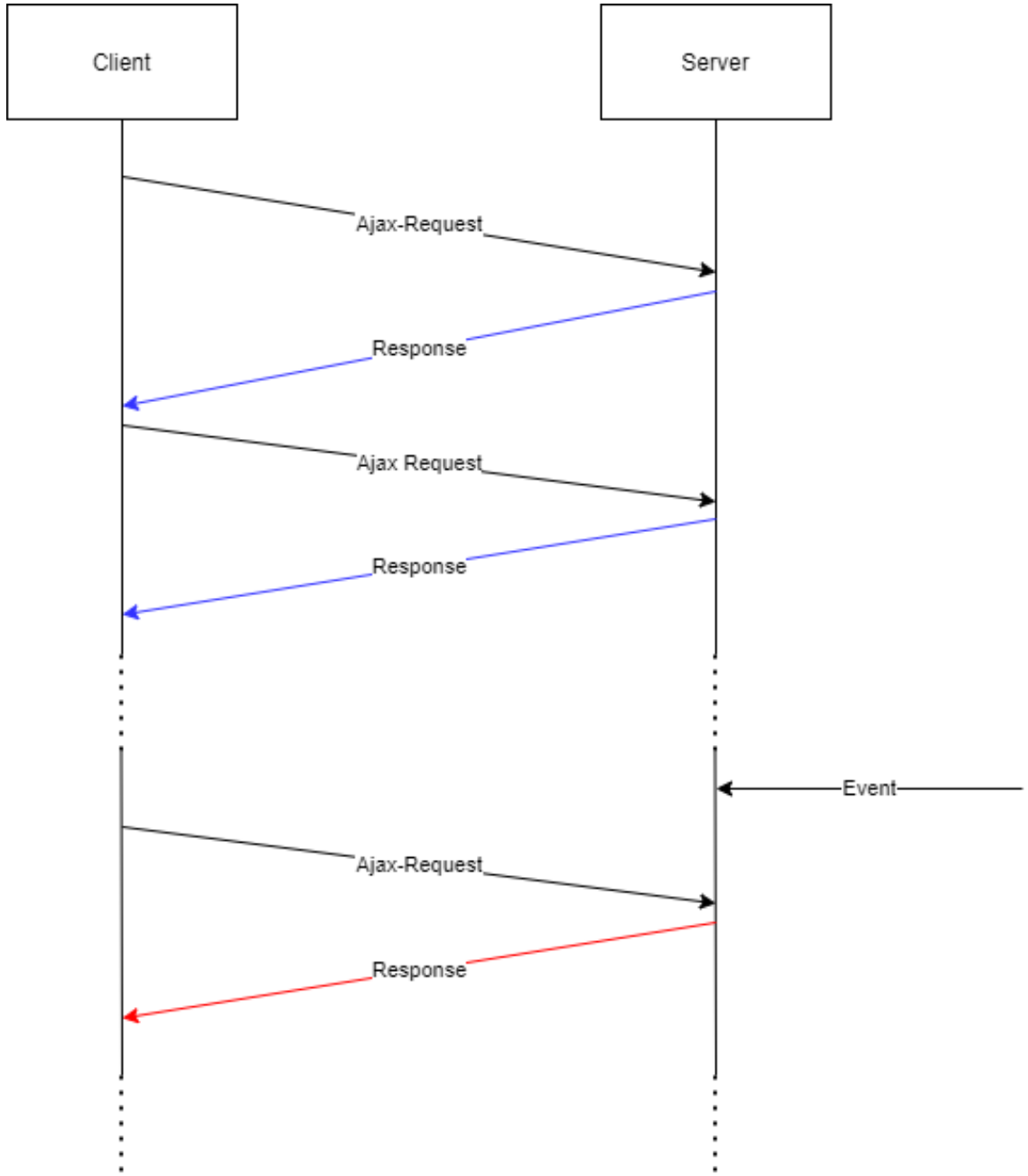
Polling-Problem

Warten auf Veränderung / Ereignis

Keine Möglichkeit der eigenständigen Benachrichtigung

Client muss Server immer erneut Anfragen und erhält die gleichen Daten

→ Resultat: Hohe unnötige Serverlast



Ajax: Vor- und Nachteile

Vorteile	Nachteile
Simple Implementierung	Polling-Problem
Hohe Verbreitung	Keine Status-Speicherung
Hohe Unterstützung	Tracking nur über Umwege möglich

WebHooks

„Reverse Api“

Ereignisbezogenes Datenversenden

Primär für Server-Server-Applikationen

Typische Anwendungsbeispiele:

Ereignis-Mitteilungen

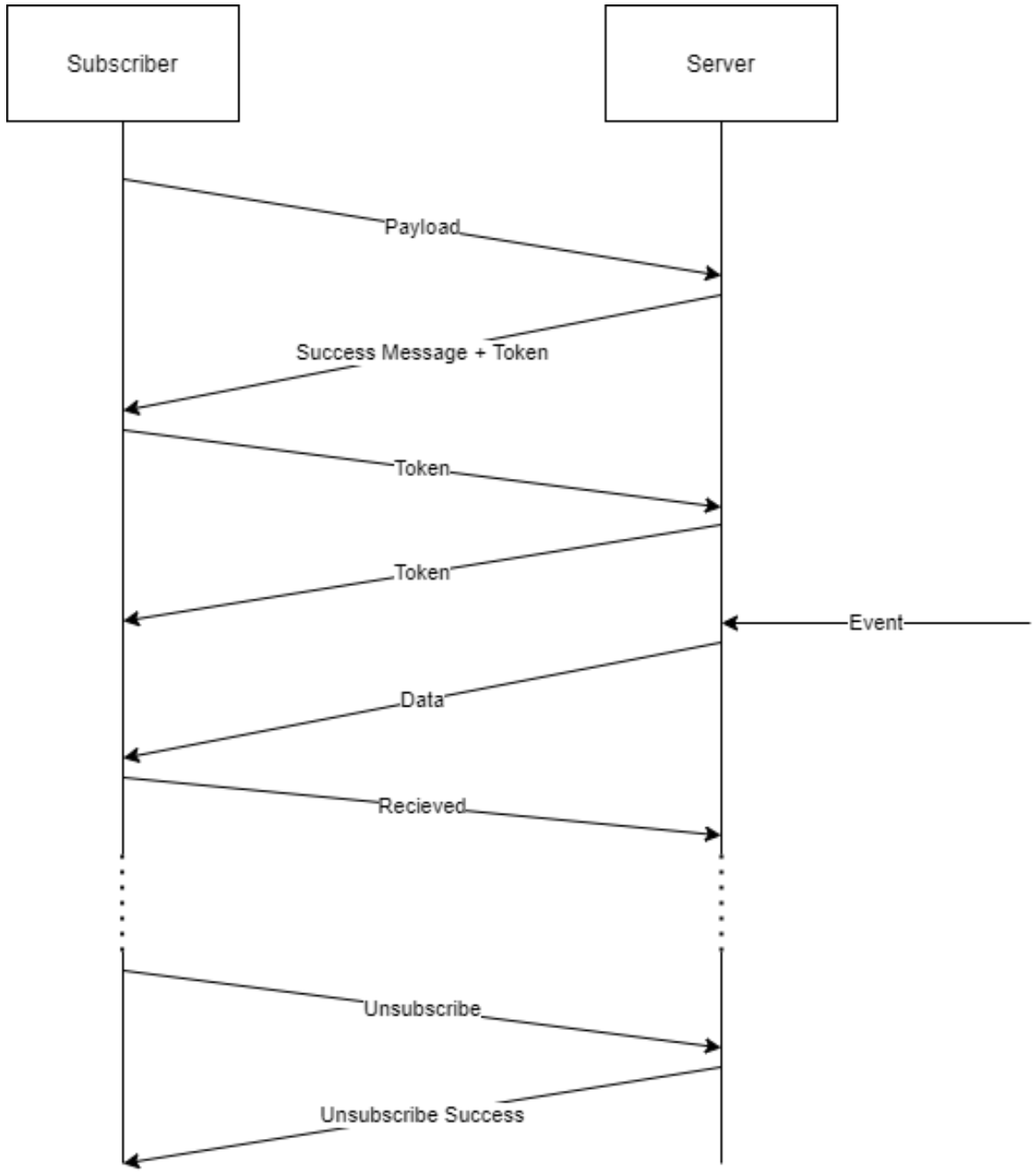
Weiterleitung von Daten

WebHooks: (typische) Funktionsweise

1. Subscriber sendet Payload
2. Server generiert Sicherheitstoken
3. Subscriber bestätigt Beitritt
4. Server sendet Erfolgsmeldung

5. Ereignis: Server sendet vorher verhandelte Daten
6. Subscriber quittiert

7. Subscriber sendet Unsubscribe
8. Server quittiert Unsubscribe



WebHooks: Vor- und Nachteile

Vorteile	Nachteile
Ereignisgetriebene Kommunikation	Keine Standarisierung
Kein Protokolloverhead	Firewall-Probleme
Entlastung von Client und Server	Einseitige Kommunikation

WebSockets

Eigenes Protokoll (ws:// bzw. wss://)

Wird über HTTP-Upgrade aufgebaut

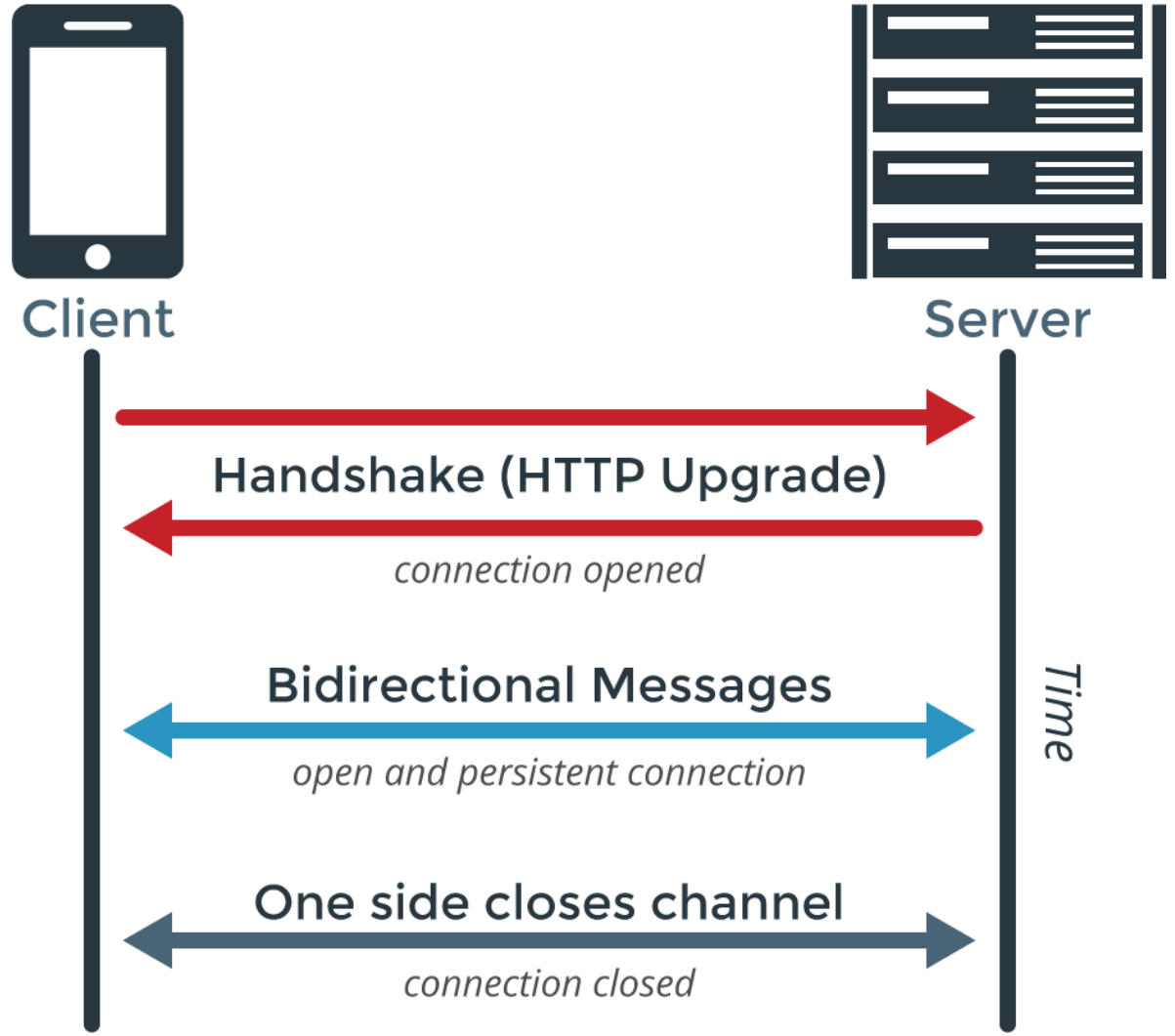
Bidirektionale Kommunikation

Persistente Verbindung

WebSockets: Funktionsweise

https://images.ctfassets.net/3prze68gbwl1/asset-17suaysk1qa1k1c/3ea3d8f59a4a701e383f01e0157083be/WebSockets-Diagram.png

1. Client fragt Server mit Upgradewunsch an
2. Server bestätigt (Handshake) → Protokollwechsel
3. Austausch von beliebig vielen Nachrichten (egal von wem initiiert)
4. Eine Partei schließt Verbindung



WebSockets: Code

CLIENT (JAVASCRIPT)

```javascript
let socket = new WebSocket('ws://url-zum-server');
socket.onopen = function () {
    //etwas tun wenn die Verbindung erstmals steht
}
socket.onmessage = function (event)  {
    //einkommende Nachrichten verarbeiten
    let data = event.data;
}
socket.close(); // beendet Verbindung
```

SERVER (PHP)

```php
$address = '0.0.0.0';
$port = 8060;

// Create WebSocket.
$server = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
socket_set_option($server, SOL_SOCKET, SO_REUSEADDR, 1);
socket_bind($server, $address, $port);
socket_listen($server);
$client = socket_accept($server);

// Send WebSocket handshake headers.
$request = socket_read($client, 5000);
preg_match('#Sec-WebSocket-Key: (.*)\r\n#', $request, $matches);
$key = base64_encode(pack(
    'H*',
    sha1($matches[1] . '258EAFA5-E914-47DA-95CA-C5AB0DC85B11')
));
$headers = "HTTP/1.1 101 Switching Protocols\r\n";
$headers .= "Upgrade: websocket\r\n";
$headers .= "Connection: Upgrade\r\n";
$headers .= "Sec-WebSocket-Version: 13\r\n";
$headers .= "Sec-WebSocket-Accept: $key\r\n\r\n";
socket_write($client, $headers, strlen($headers));
```

WebSockets: Vor- und Nachteile

Vorteile
Bidirektionale Kommunikation
Kein Protokolloverhead
Einfache Protokollstruktur
Clientseitig einfach zu implementieren
Polling entfällt

Nachteile
Schlechte Skalierbarkeit
Hohe Serverauslastung

Was ist jetzt das Beste?

Es kommt auf die Anforderungen und das Anwendungsumfeld an!

- Wer nutzt meine Anwendung?
- Wie viele Personen nutzen meine Anwendung?
- Wie oft ändert sich der abgefragte Inhalt?
- Was passiert mit abgefragten Daten?
- Wie zeitkritisch ist meine Anwendung?