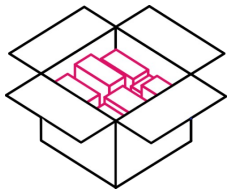
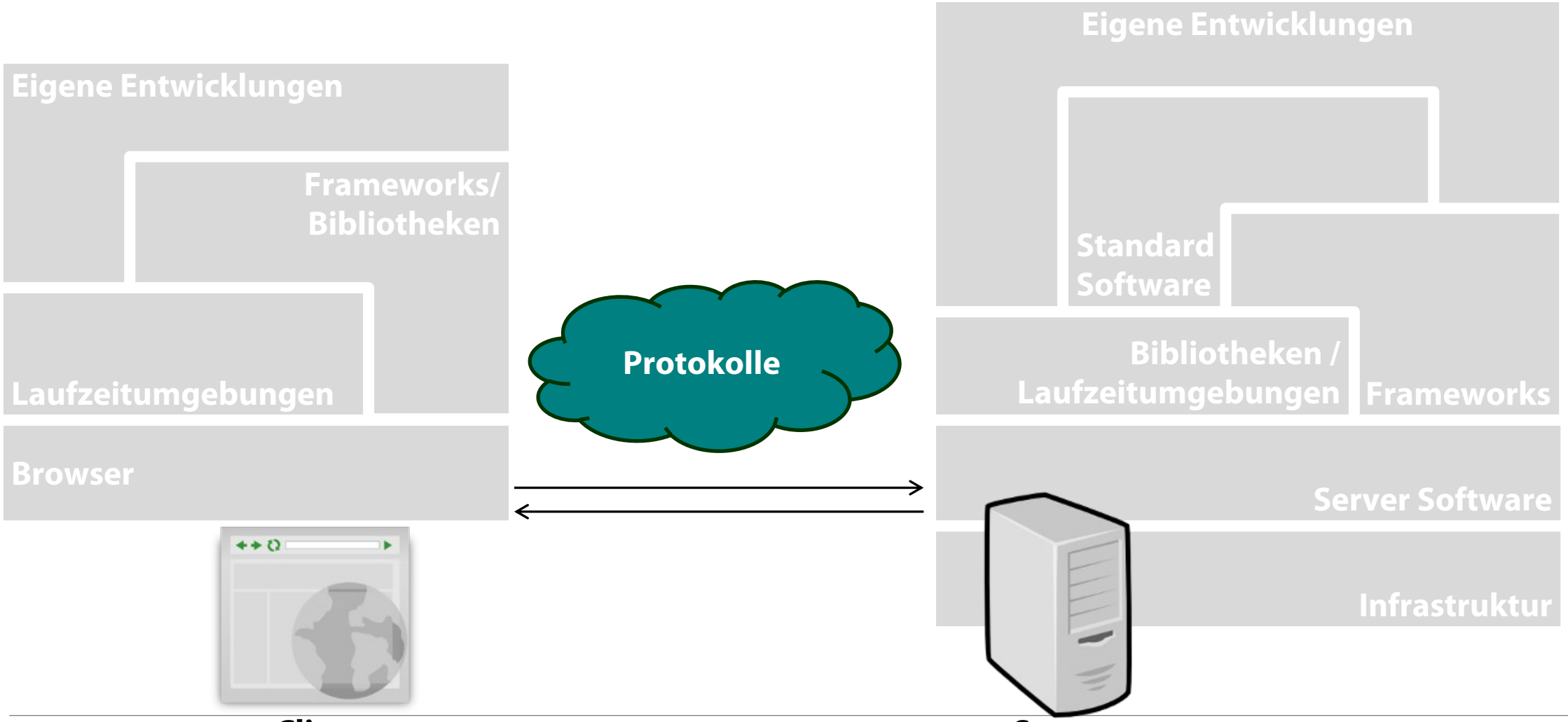
WebSockets

Hoai Viet Nguyen – TH Köln

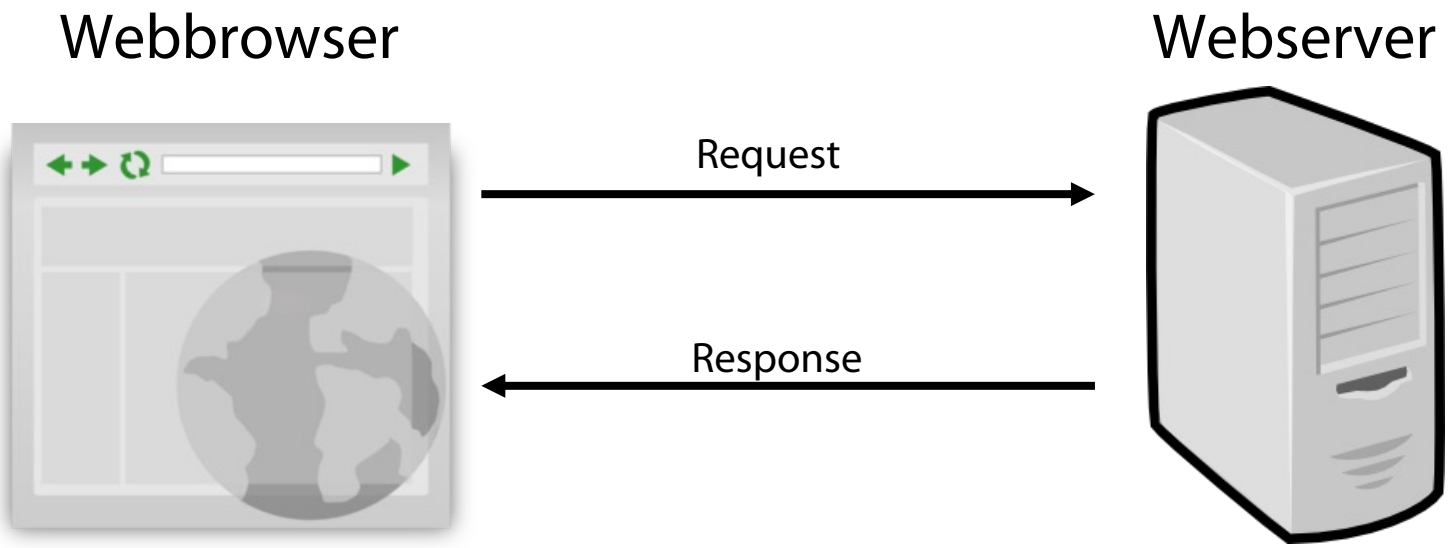
Technology
Arts Sciences
TH Köln



Protokolle


Eigene Entwicklungen
Frameworks/
Bibliotheken
Laufzeitumgebungen
Browser
Protokolle
Eigene Entwicklungen
Standard
Software
Bibliotheken /
Laufzeitumgebungen
Frameworks
Server Software
Infrastruktur
Client
Server

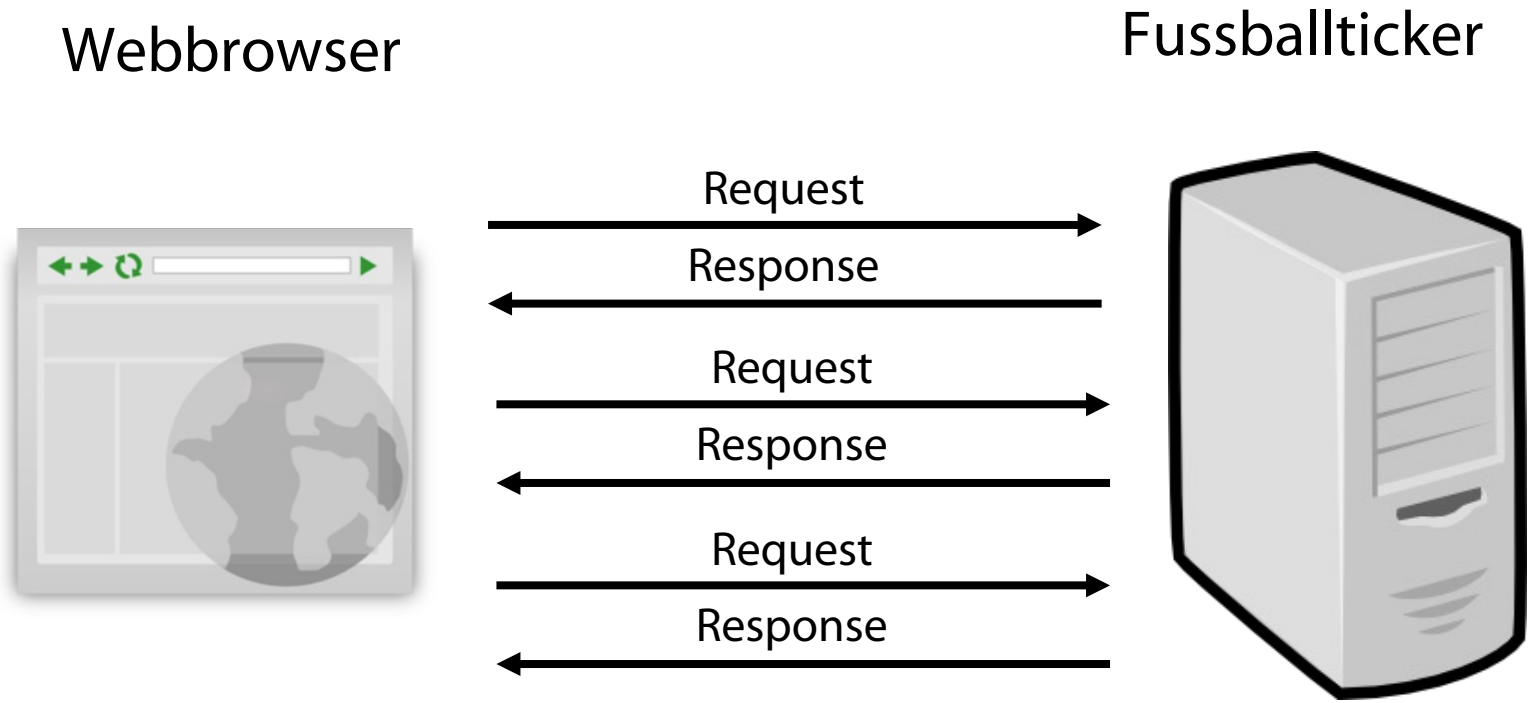
Client-Server-Kommunikation im Web



AJAX mit XHR-Requests

```
var xhr = new XMLHttpRequest()
xhr.open("POST","/users")
xhr.send("...data...");
xhr.onreadystatechange = function(){
    if(this.readyState == 4 && this.status == 200){
        //...
    }
}
```

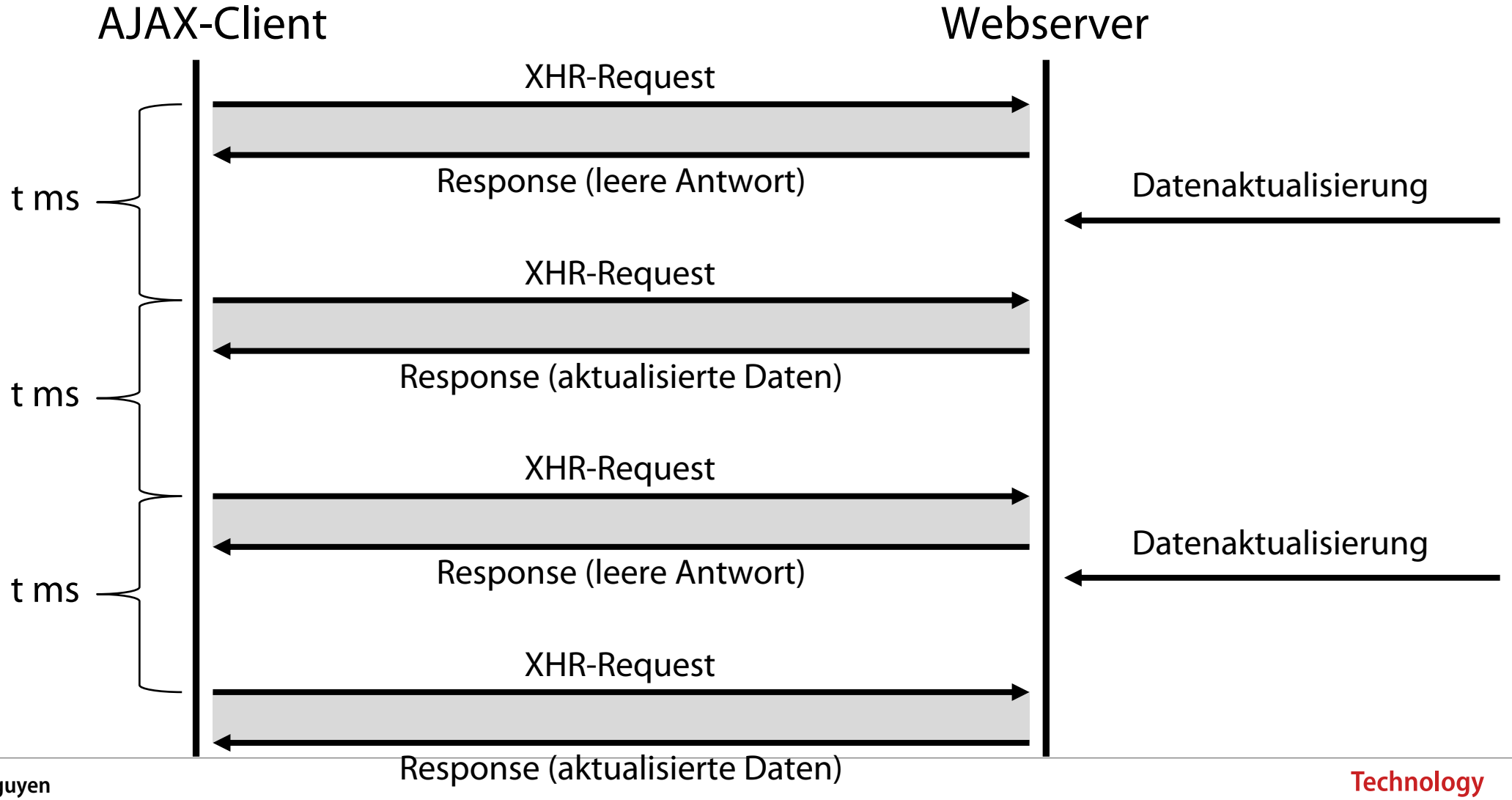
REST Request/Response-Modell für Event-basierte Daten



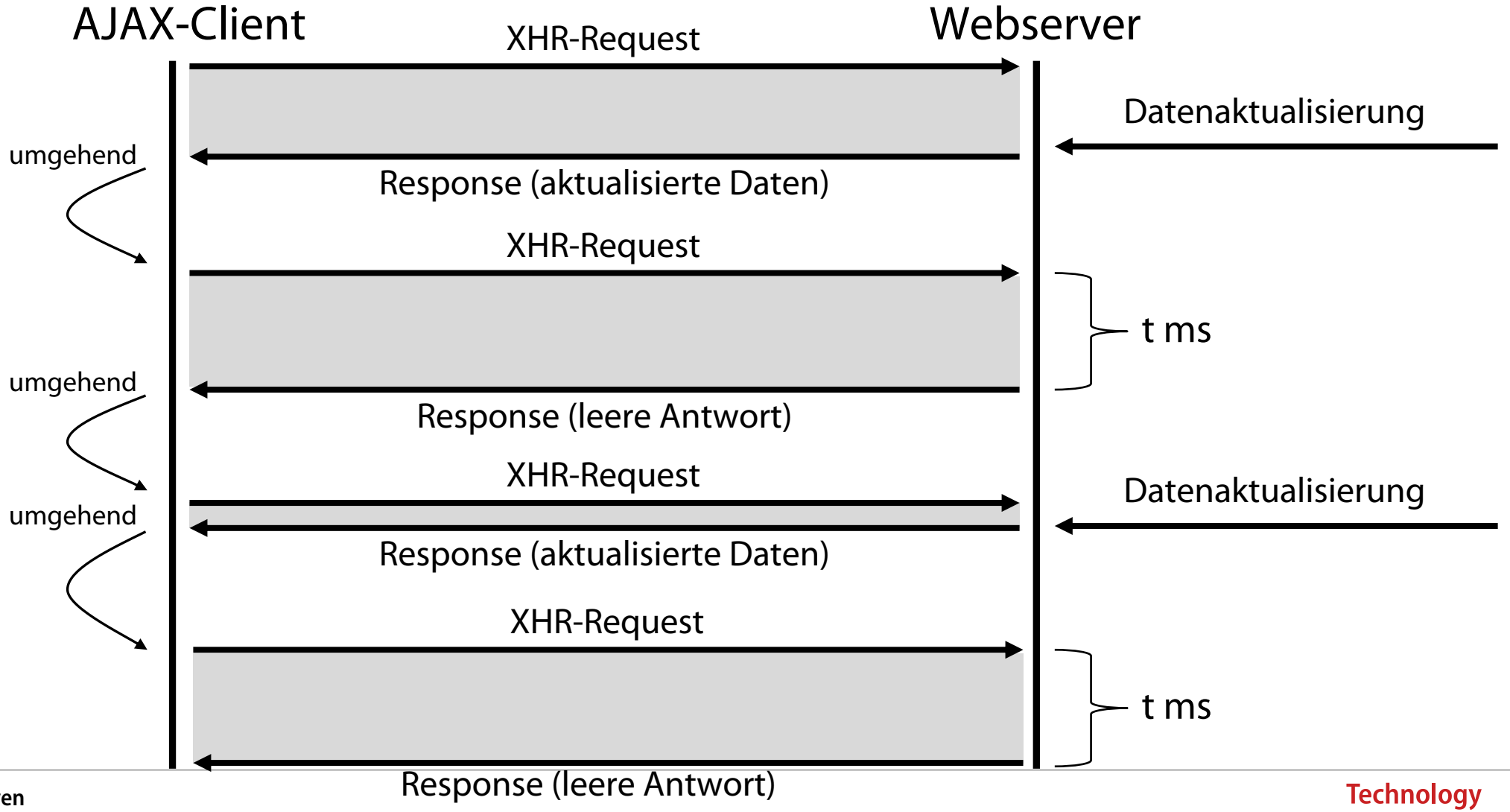
Ajax-Requests für Fussballticker

```
while(true){
    var xhr = new XMLHttpRequest()
    xhr.open("GET","/footballticker",true)
    xhr.onreadystatechange = function(){
        if(this.readyState == 4 && this.status == 200){
            // Update...
        }
    }
}
```

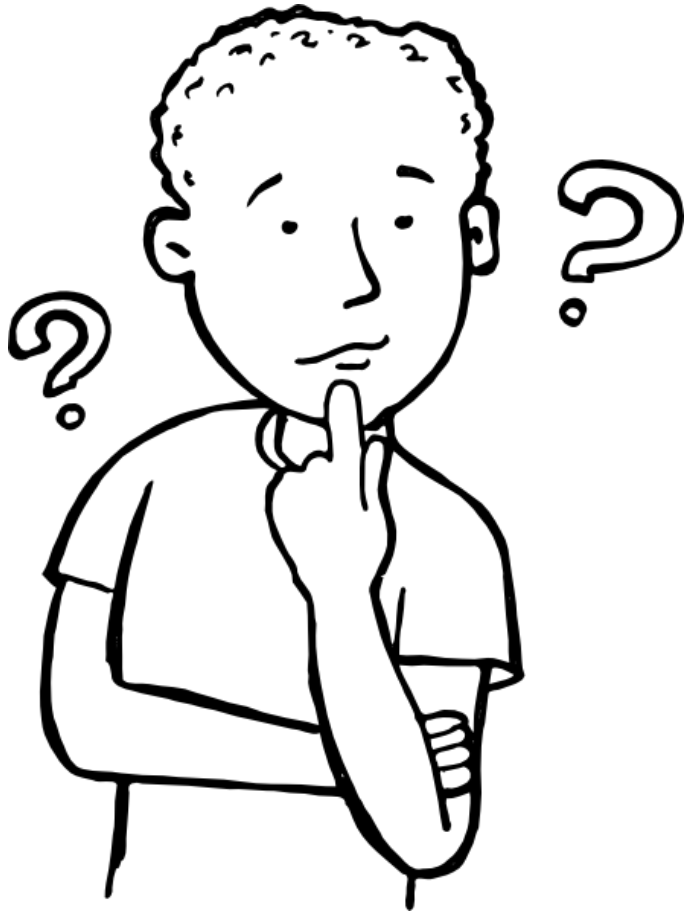
Polling mit XHR-Request für Event-basierte Daten



Long-Polling mit XHR-Request für Event-basierte Daten



Nachteile Polling/Long-Polling für Eventbasierte-Daten ?



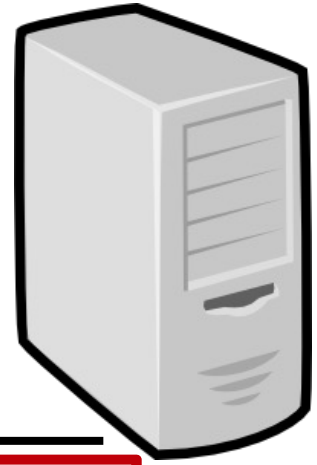
Nachteile Polling/Long-Polling für Event-basierte Daten

- Gefahr für Verlust von Events
- Initiierung des Datenaustauch geht immer vom Client aus
- Hoher Ressourcenbedarf
- Großer Overhead

Overhead in XHR-Request/Response

Webbrowser

Webserver



```
GET /events/{id} HTTP/1.1
Host: example.org
Accept: application/json
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 …
```

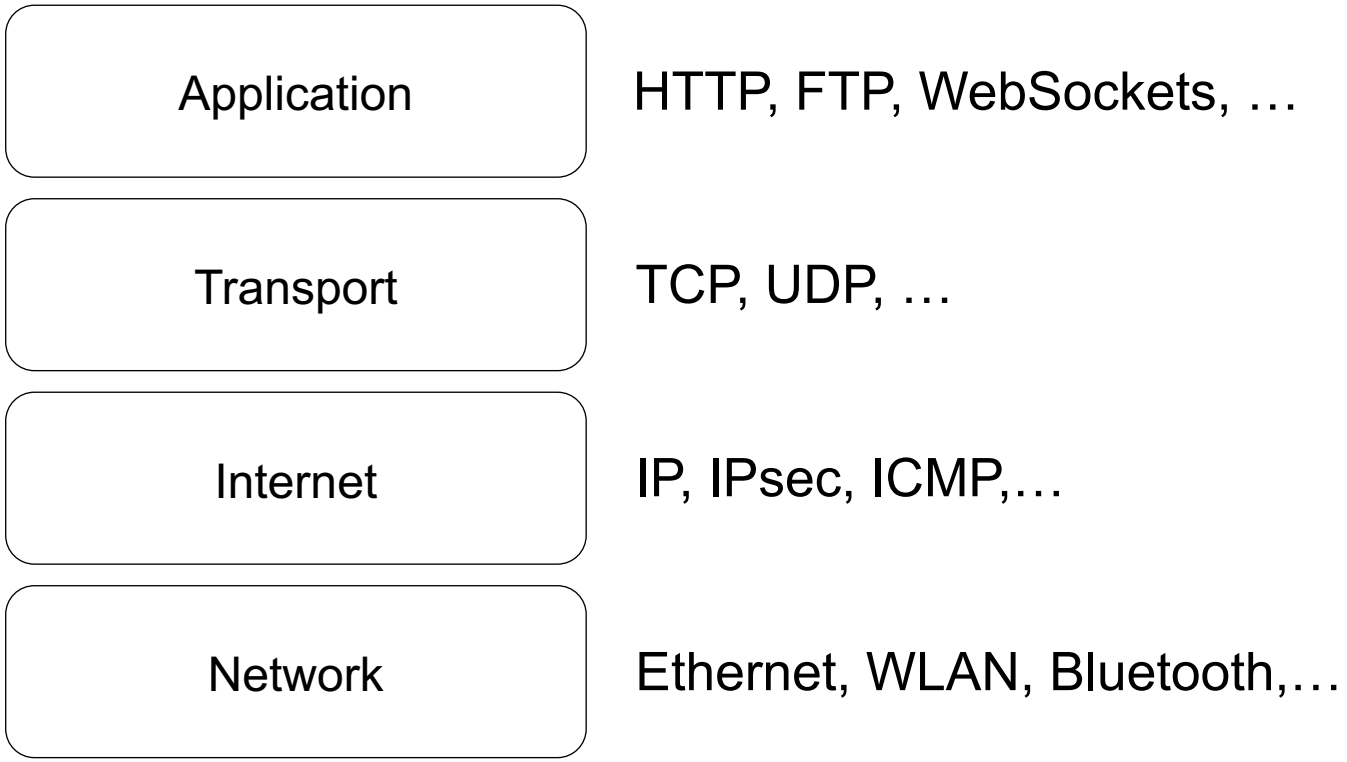
```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length:
```

{“stuttgart“:“1“,“koeln“:“0“}

WebSockets

- Bidirektionale Kommunikation in Webanwendungen
- W3C bzw. RFC-Standard
  - https://websockets.spec.whatwg.org
  - https://datatracker.ietf.org/doc/html/rfc6455
- Bestandteil aller gängigen Web Browsers und Web Frameworks
- Geringer Overhead: 2 bzw. 6 Bytes
- Default Ports: 80 und 443

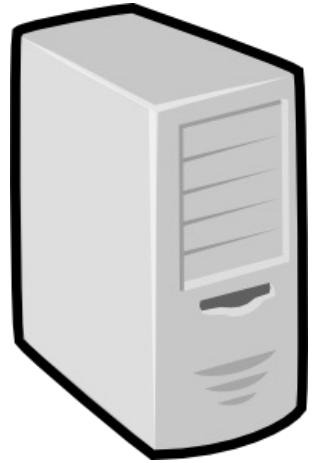
WebSockets im ISO/OSI Referenzmodell



WebSocket Handshake

Webbrowser

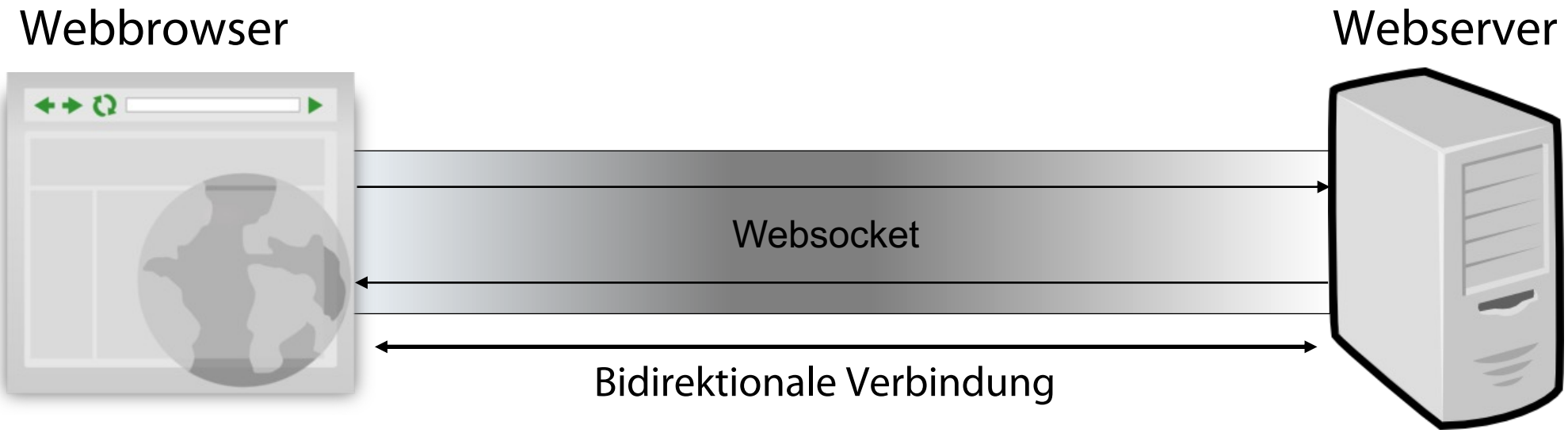
Webserver



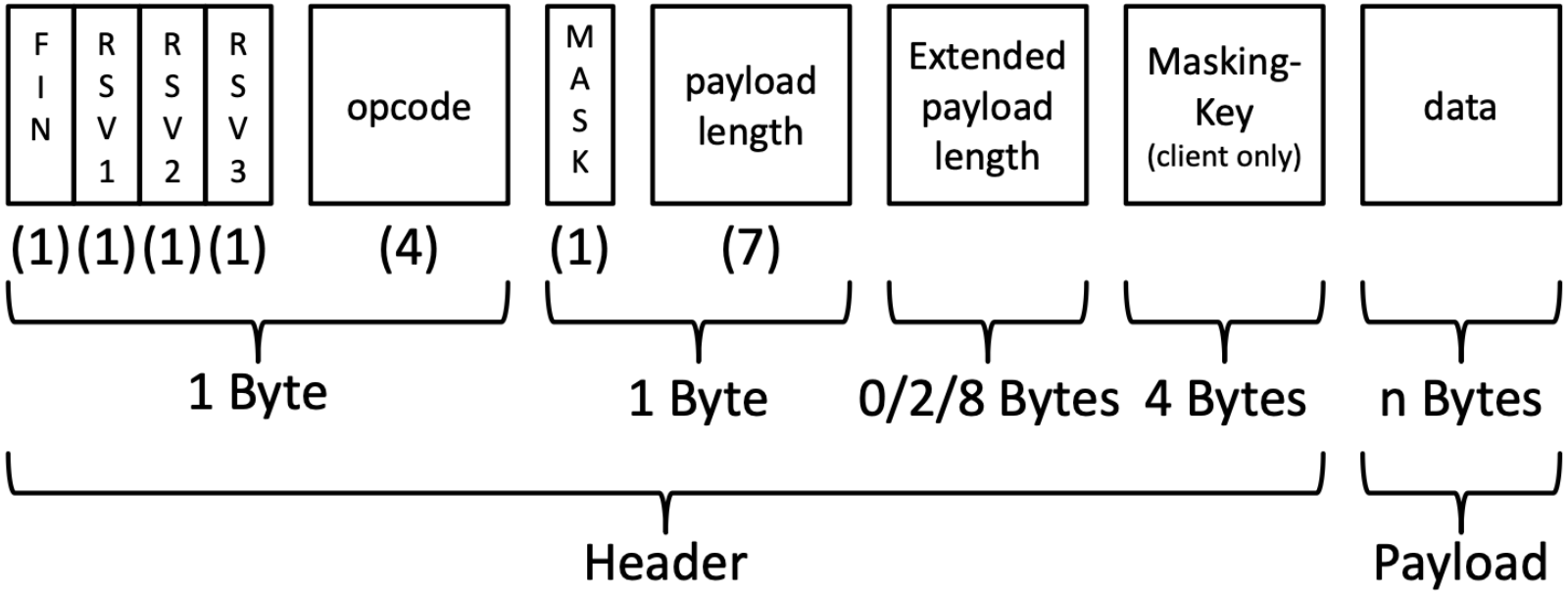
GET /chat HTTP/1.1
Host: example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo=

Kommunikation mit WebSockets



Aufbau WebSocket-Frame vereinfacht [PLN2015]



[PLN2015] P. L. Gorski, L. Lo Iacono, H. V. Nguyen, WebSockets – Moderne HTML5-Echtzeit-anwendungen entwickeln, Carl Hanser Verlag, 2015

Daten senden im Webbrowser

- Konstruktor

```
var ws = new WebSocket(url, [, protocols])
```

- Textdaten

```
void send(DOMString data);
```

- Binärdaten

```
void send(Blob data);
void send(ArrayBuffer data);
void send(ArrayBufferView data);
```

WebSocket Events

- `onopen`
- `onerror`
- `onmessage`
- `onclose`

Daten empfangen

```
ws.onmessage = function(message) {
    var data = message.data;
    if(data === "someCommand") {
        doSpecificAction()
    } else {
        // Ignore unknown command message
    }
}
```

Verbindung schließen im Webbrowser

Aus Sicht der W3C Javascript Browser-API

- Man selbst schließt den WebSocket

```
void close();
void close(unsigned short code);
void close(unsigned short code, DOMString reason);
```

- Die Gegenstelle hat den WebSocket geschlossen

```
ws.onclose = function(event) {
    if(!event.wasClean) {
        console.log("WS-Fehlercode: " + event.code);
        console.log("WS-Fehlergrund: " + event.reason);
    }
};
```

WebSocket EchoClient

```
<h1>Echo Demo</h1>
<input type="text" id="message">
<button type="button" onclick="sendMessage()">Send</button>
<div id="echo"></div>
<script>
    var ws = new WebSocket("wss://echo.websocket.org")
    ws.onopen = function (event) {
    alert("WebSocket successfully established")
    }

    function sendMessage(){
    var message = document.getElementById("message").value
        ws.send(message)
    }

    ws.onmessage = function (message) {
        echo.innerHTML += "<p>" + message.data + "</p>"
    }
</script>
```

Lernzielkontrolle

- Was ist Polling und Long Polling?
- Was sind die Nachteile von Polling und Long Polling?
- Was sind WebSockets?
- Was sind die Vorteile von WebSockets?

Zusammenfassung

- **Polling/Long Polling:** Request in Dauerschleife, um Server nach aktuellen Daten anzufragen
- **Bei Polling/Long Polling wird ständig ein neuer Request erstellt und bei jeder Anfrage bzw. Antwort werden viele Metadaten mitgeschickt**
- **WebSockets ermöglichen eine bidirektionale Verbindung zwischen Client und Server**
- **Der Overhead von WebSockets ist deutlich geringer als bei HTTP**