FRONTEND
FREUNDE
MUENSTER

WebRTC mit PeerJS

Simon Bungartz
24.08.2023

WebRTC

1.

Was ist WebRTC?

Was ist WebRTC?

FRONTEND
FREUNDE
MUENSTER

https://workspace.google.com/intl/de/resources/video-conferencing/

Was ist WebRTC?

FRONTEND
FREUNDE
MUENSTER

- → Web Real-Time Communications
- → Web-Standard
- → Echtzeitkommunikation zwischen Browsern
- → Audio / Video
- → Peer-to-Peer

FRONTEND
FREUNDE
MUENSTER

WebRTC

2.

Historie

Ein bisschen Geschichte

FRONTEND FREUNDE MUENSTER

- 2011: W3C Arbeitsgruppe zu WebRTC beginnt
    - Peer-to-Peer-Verbindungen bei Skype und Co. im Einsatz
- November 2017: WebRTC 1.0 von “Draft” zu “Candidate Recommendation”
- Januar 2021: WebRTC 1.0 zu “Recommendation”
- Inzwischen stabile API und breiter Browser-Support

WebRTC

FRONTEND
FREUNDE
MUENSTER

3.

Wie kann ich es benutzen?

Wie benutze ich WebRTC?

FRONTEND FREUNDE MUENSTER

- → ICE
- → STUN
- → TURN
- → RTCPeerConnection
- → RTCDataChannel
- → sendOffer
- → oniceconnectionstatechange

answer, leaving out the ICE layer for the moment:

1. The caller captures local Media via `MediaDevices.getUserMedia`
2. The caller creates `RTCPeerConnection` and calls `RTCPeerConnection.addTrack()` (Since `addStream` is deprecating)
3. The caller calls `RTCPeerConnection.createOffer()` to create an offer.
4. The caller calls `RTCPeerConnection.setLocalDescription()` to set that offer as the *local description* (that is, the description of the local end of the connection).
5. After setLocalDescription(), the caller asks STUN servers to generate the ice candidates
6. The caller uses the signaling server to transmit the offer to the intended receiver of the call.
7. The recipient receives the offer and calls `RTCPeerConnection.setRemoteDescription()` to record it as the *remote description* (the description of the other end of the connection).
8. The recipient does any setup it needs to do for its end of the call: capture its local media, and attach each media tracks into the peer connection via `RTCPeerConnection.addTrack()`
9. The recipient then creates an answer by calling `RTCPeerConnection.createAnswer()`.
10. The recipient calls `RTCPeerConnection.setLocalDescription()`, passing in the created answer, to set the answer as its local description. The recipient now knows the configuration of both ends of the connection.
11. The recipient uses the signaling server to send the answer to the caller.
12. The caller receives the answer.
13. The caller calls `RTCPeerConnection.setRemoteDescription()` to set the answer as the remote description for its end of the call. It now knows the configuration of both peers. Media begins to flow as configured.

Wie benutze ich WebRTC?

FRONTEND FREUNDE MUENSTER

- ICE
- STUN
- TURN
- RTCPeerConnection
- RTCDataChannel
- sendOffer
- oniceconnectionstatechange

PEER JS

answer, leaving out the ICE layer for the moment:

1. The caller captures local Media via `MediaDevices.getUserMedia`
2. The caller creates `RTCPeerConnection` and calls `RTCPeerConnection.addTrack()` (Since `addStream` is deprecating)
3. The caller calls `RTCPeerConnection.createOffer()` to create an offer.
4. The caller calls ... `cription()` to set that offer as the *local* ... end of the connection).
5. ... N servers to generate the ice candidates
6. ... the offer to the intended receiver of the call.
7. ... `erConnection.setRemoteDescription()` to record it ... e other end of the connection).
8. The recipient does any setup it needs to do for its end of the call: capture its local media, and attach each media tracks into the peer connection via `RTCPeerConnection.addTrack()`
9. The recipient then creates an answer by calling `RTCPeerConnection.createAnswer()`.
10. The recipient calls `RTCPeerConnection.setLocalDescription()`, passing in the created answer, to set the answer as its local description. The recipient now knows the configuration of both ends of the connection.
11. The recipient uses the signaling server to send the answer to the caller.
12. The caller receives the answer.
13. The caller calls `RTCPeerConnection.setRemoteDescription()` to set the answer as the remote description for its end of the call. It now knows the configuration of both peers. Media begins to flow as configured.

Wie benutze ich PeerJS?

```
import { Peer } from 'peerjs';

const peer = new Peer();

const myStream = await navigator.mediaDevices.getUserMedia(
  { audio: true, video: true }
);

peer.on('open', (id) => console.log(`Meine ID ist: #{id}`));
peer.on('call', (call) => {
  call.answer(myStream);

  call.on('stream', (theirStream) => {
    const videoElement = document.querySelector("video");
    videoElement.srcObject = theirStream;
  });
});
```

```
import { Peer } from 'peerjs';

const peer = new Peer();

const myStream = await navigator.mediaDevices.getUserMedia(
  {audio: true, video: true}
);


const call = peer.call("<remote Id>", myStream);


call.on('stream', (theirStream) => {
  const videoElement = document.querySelector("video");
  videoElement.srcObject = theirStream;
});
```

FRONTEND
FREUNDE
MUENSTER

WebRTC

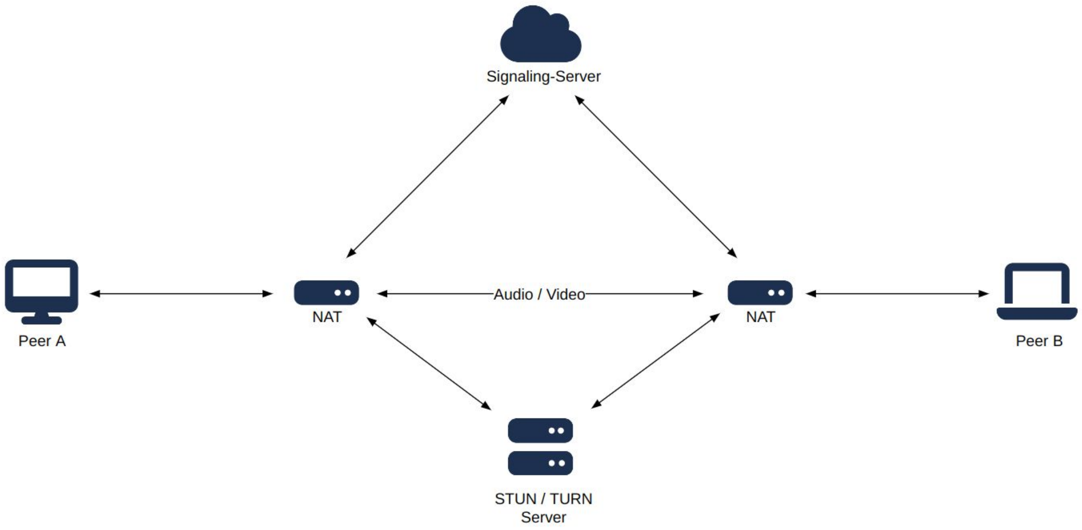
4.

Was passiert da (so in etwa)?

Wie funktioniert WebRTC?

FRONTEND FREUNDE MUENSTER

- → Signaling-Server: Vermittelt
- → STUN: “wieistmeineip.de”
- → TURN: Weiterleitung statt P2P
- → ICE: Interactive Connectivity Establishment



FRONTEND
FREUNDE
MUENSTER

WebRTC

5.

Live Demo

FRONTEND
FREUNDE
MUENSTER

WebRTC

6.

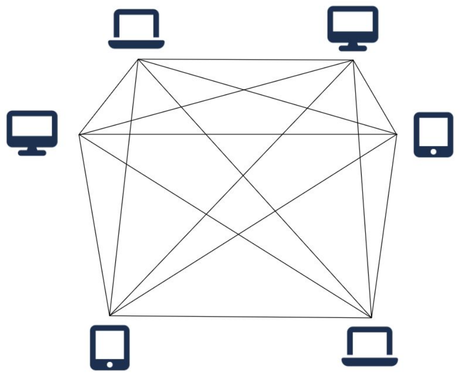
Weitere Details

Weitere Details

FRONTEND FREUNDE MUENSTER

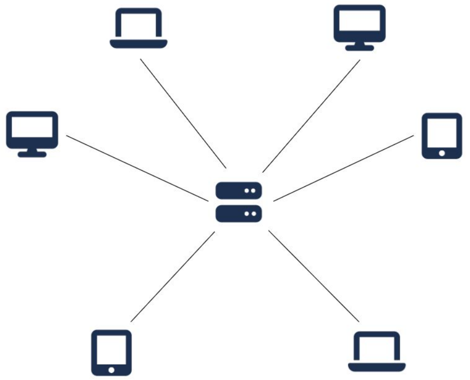
Dev-Tools

- Chrome: chrome://webrtc-internals
- Firefox: about:webrtc



Performante Gruppen-Calls

- Multipoint Control Unit (MCU)
- Selective Forwarding Unit (SFU)



Security von WebRTC

FRONTEND FREUNDE MUENSTER

Standard hat Security gleich mitgedacht

- → Verschlüsselung ist Pflicht
- → Sicherer Schlüsselaustausch
- → Auch TURN-Server kann nicht mitlesen
- → SFUs / MCUs hebeln Ende-zu-Ende-Verschlüsselung aus
- → Signalling und Darstellung sind Eigenverantwortung

Weitere Use-Cases

FRONTEND FREUNDE MUENSTER

Screen-Sharing

→ getDisplayMedia()

Data-Channels

→ Dateien schicken mit sharedrop.io

→ Filesharing mit webtorrent.io

Überblick

FRONTEND
FREUNDE
MUENSTER

- WebRTC ist mächtig
- WebRTC ist kompliziert
- PeerJS übernimmt Signalling und ICE-Austausch
- MediaStreams werden an HTML5 video-Elemente gehängt
- Eigenes Backend für non-triviale Use-Cases benötigt

```
navigator.mediaDevices.getUserMedia()

navigator.mediaDevices.getDisplayMedia()

<video autoplay></video>

peer.call()

peer.on("call", …)

call.on("stream", (stream) => {
  videoElement.objSource = stream
});
```

Links

FRONTEND FREUNDE MUENSTER

Code und Folien von heute:

→ https://github.com/sbungartz/frontend-freunde-webrtc-demo

Weitere Links:

→ https://webrtc.org/getting-started/overview

→ https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

→ https://github.com/peers/peerjs

FRONTEND
FREUNDE
MUENSTER

WebRTC

Danke