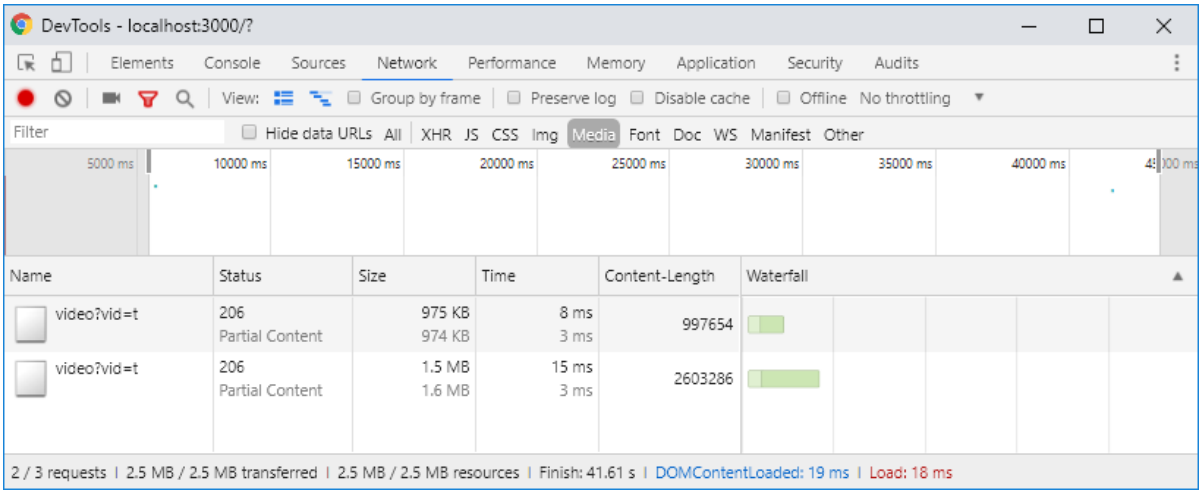
Dokumentation UE 13
von Janik Mayr

Einstiegs-URL: http://localhost:3000/
Ordner für Videos: ./public/videos/

Network Behavior in Chrome Dev Tools



Video benötigt zwei Requests um vollständig geladen zu werden.
Teil des Videos kann in diesem Fall beim Start noch gecashed sein(Da zu kleine Größe der Datei gesamt).

Server

GET auf Path: /video?vid=<<name>> Ablauf:

1. Video identifizieren

```
/* Mithilfe von simplen Entscheidungsbaum können Abkürzungen verwendet werden. */
if (param !== undefined) {
    if (param === 'yellifish' || param === 'yelli' || param === 'y') {
        path = 'public/images/yellifish.mp4';   //Video über Quallen
} else {
    if (param === 'turtle' || param === 't') {
        path = 'public/images/turtle.mp4';      //Video über Schildkröten
    } /*ADD ELSE IF FOR OTHER VIDEOS HERE*/
}
```

Wenn das Video keinen Kurznamen hat, wird nach dem Dateinamen gesucht:

```
if (path === undefined || path == null) {
    path = `public/videos/${param}`;    //Pfad wird ergänzt
}
```

2. Existiert Datei auf Pfad ?

```
fs.access(path, fs.F_OK, (err) => { //Nicht blockirender Zurgriff auf FileSystem,
    if (err) {                      // wenn File mit geg. Path nicht existiert wird ein Fehler geworden.
        //------- DATEI EXISTIERT NICHT FLOW -------
    } else {
        //------- DATEI EXISTIERT FLOW -------
    }
}
```

i. Datei Existiert

a. Größe des Files in `fileSize` Laden

b. Aus Request die Range der Bytes aus Header `const range = req.headers.range` abfragen. (Fromat Range: `bytes=%d-%d` , erste Zahl: Start Byte, Zweite Zahl: Bis Byte x maximal laden)

c. Range r mit ?

a. `r[0,_[` :

a. Stringmanipulation von Range-ObjectString

b. Byterangestart parsen zu Integer(Radix 10)

c. Byterangeende parsen zu Integer(Radix 10)

```
/*Schritt a-c*/
const parts = range.replace(/bytes=/, "").split("-");
const start = parseInt(parts[0], 10);
const end = parts[1] ? parseInt(parts[1], 10) : fileSize - 1;
```

d. Chunkgröße ermitteln `chunksize = (end - start) + 1;`

e. Dateisector laden `file = fs.createReadStream(path, {start, end})`

f. Response-Header erzeugen

```
const head = {
     /*
         Angaben in Bytes über die Übermittelten Daten: <Start>-<Ende>/<FileSizeGesamt>
     */
     'Content-Range': `bytes ${start}-${end}/${fileSize}`,
     'Accept-Ranges': 'bytes',
     /*
         Größe des Chunkes ^= end - start + 1 (+ 1 um Array 0 Indexing auszugleichen)
     */
     'Content-Length': chunksize,
     /*
         Content-Type hard-coded auf mp4, falls andere Videoformate verfügbar,
         muss dynamisch von File ableitent werden
     */
     'Content-Type': 'video/mp4',
};
```

g. Response senden
Header mit Code 206 auf Stream schreiben. Code 206 bedeutet, dass es sich um partial content handelt und der Client den Rest noch anfragen muss. Der Chunk des File wird anschließend in den Responsebody geschrieben.

```
res.writeHead(206, head);
file.pipe(res);
```

b. `r[x,_[` :

a. Response-Header erzeugen
Header mit Filesize, da das File auf einmal geladen wird. Content-Type hard-coded auf mp4, falls andere Videoformate verfügbar, muss dynamisch von File ableitent werden.

```
const head = {
     'Content-Length': fileSize,
     'Content-Type': 'video/mp4',
};
```

b. Response senden
Header mit Code 200, da alles übermittelt wird.

```
                    res.writeHead(200, head);
                    fs.createReadStream(path).pipe(res);
```

ii. Datei existiert nicht:

```
        res.status(404).send('Not found'); // HTTP status 404: NotFound senden
```

Client

1. Form:

```
<form>
   <input id="vn" type="text" value="">
   <input type="button" value="OK" onclick="loadVideo()">
</form>
```

2. Load Video Method:
   Erstellen eines Videoplayers im Video Player Container und setzen der Source auf Request URL des Servers.

```
function loadVideo() {
      let div = document.getElementById("vp");  //Video Player Container Div
      let name = document.getElementById("vn").value; //Text aus Imputfield
      if (name !== undefined && name !== null && name.length > 0) {
        div.innerHTML = `<video controls autoplay>
                            <source onerror="handleError()" src="/video?vid=${name}" type="video/mp4">
                        </video>`;
      }
}
```

3. ErrorHandling:
   Wenn in der Source Tag innerhalb des VideoPlayers einen Fehler bekommt, wird folgende Methode ausgeführt.

```
function handleError() {
      alert('Video not found');
}
```