

# Walkthrough

## Why this code was created?

To download videos from Mediathek that can't be accessed normally without some vpn extension helper. In this case I used this series as guinea pig [Falk] (<https://www.fernsehserien.de/falk/episodenguide>).

## How this code works?

First install a browser, in my case Chromium (it's an open source browser based Chrome but without the bad parts of it). You can change and use what browser you like.

After installer you need to download some free VPN extension for you browser (setupVPN in my case).

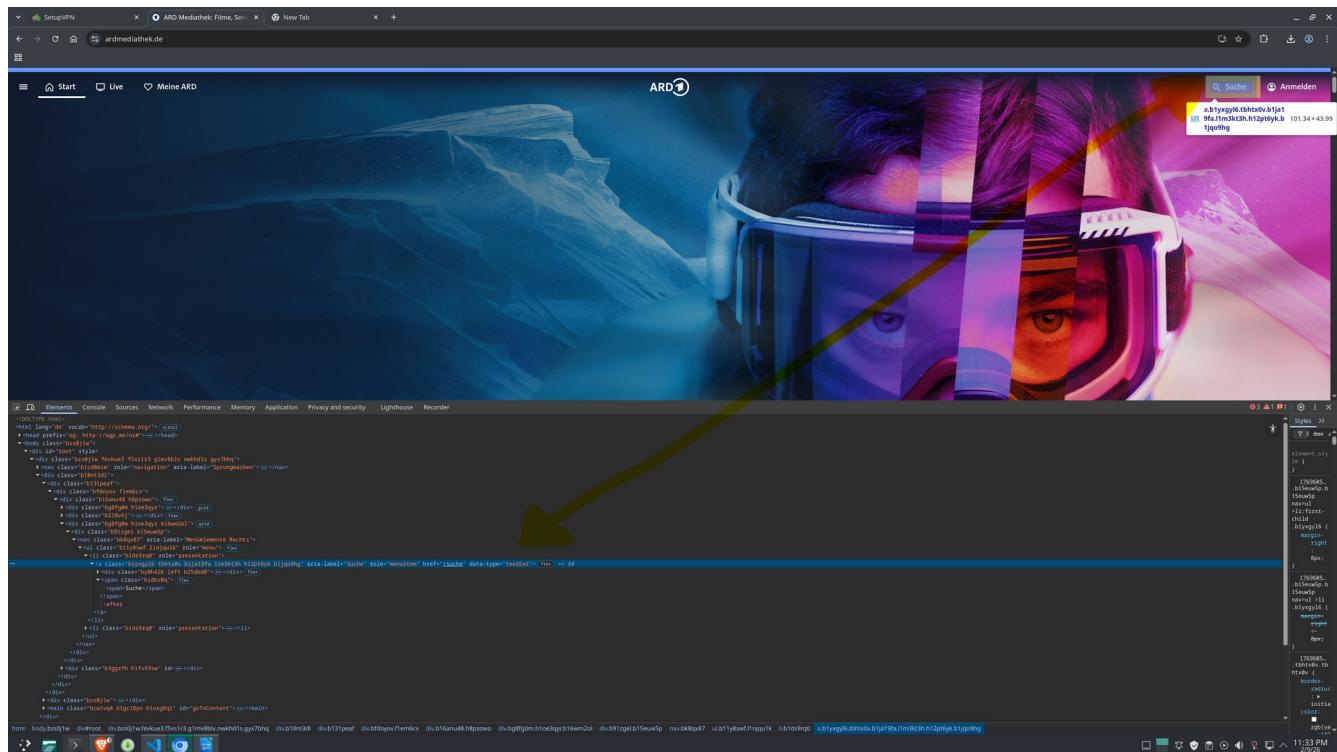
MAKE SURE THAT THE VPN EXTENSION IS OPENED AND WORKS BEFORE YOU OPEN MEDIATHEK OTHERWISE THE SIDE WILL SAY “sORrY tHe ViDeO iS nOt AvAiLaBLe iN YoUr cOuNtRy”.

When you open the Mediathek, in the search button you type the series that is available to watch (as sometimes they might delete and put the series back after a while).



Before seeing the code let's see how the site looks like and see how to save a video.

# Manual walkthrough



The search bar is this element:

```
<a class="b1yxgyl6 tbhtx0v b1ja19fa l1m3kt3h h12pt6yk b1jqo9hg active b10oyi86 t130opgm" aria-label="Suche" role="menuitem" href="/suche" data-type="textExt"><div class="by0h42k left b25dbd8"><div><svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" class="b12o3cf2"><path d="M20.72 19.34-5.52-5.59a6.82 6.82 0 0 0 1.24-3.93c0-3.76-3.01-6.81-6.72-6.81S3.605 3.981 3.01 6.81 6.72 6.81c1.57 0 3.01-5.54 4.16-1.47l5.49 5.56c.19.19.43.29.68.29s.49-.168-.29c.38-.38.38-1 0-1.38zM4.92 9.81c0-2.68 2.15-4.86 4.8-4.86s4.8 2.18 4.8 4.86-2.15 4.86-4.8-2.18-4.8-4.86"></path></svg></div></div><span class="bzdhsoq"><span>Suche</span></span></a>
```

But it can be searched with just the attribute `aria-label="Suche"` for more simplicity

The screenshot shows the ARD Mediathek homepage. At the top, there's a search bar with the placeholder "Suche nach Videos und Sendungen in der ARD Mediathek". Below it, a navigation bar includes links for "Start", "Live", and "Meine ARD". On the right, there's a login link and a search icon. The main content area is titled "Sendungen A-Z" and features a grid of categories: "Rubriken" (with icons for Film, Serien, Dokus, tagesschau & Information, and Sportschau), "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", and "#". Below this, there's a detailed view of the "Serien" category, showing a thumbnail of a TV show and some descriptive text. The browser's developer tools are open, showing the DOM structure of the page, including the search input element.

when clicking on the element, it will redirect to this url <https://www.ardmediathek.de/suche/> where an input field is working so you just need to type the series you want to watch

The screenshot shows the search results for the query "Falk Die Qual der Wahl". The search bar at the top contains the query. Below it, there are several filters: "Alle Sender", "Untertitel", "Audiodeskription", "Gebärdensprache", and "Kinder". The results are listed under the heading "Videos". The first result is "Folge 6: Die Qual der Wahl (501/E06)" from "Falk - WDR" (airdate: 24.02.2023). The second result is "Frag Valet: Eigenmarken – wann sich der Einkauf lohnt" from "Falk 4 Tage" (airdate: 14.02.2024). There are also other results like "dm EXKLUSIV bei uns" and "Mindestens 1 Tag". The browser's developer tools are visible on the right side of the screen.

so we type the name of the tv show + name of episode; the url will also have the query params to search for it. 99% of the cases it is the first element. I am saying 99% because there is also this episode when the episode is the second in the search list

ARDmediathek.de/suche/Falk%20Die%20Arzt-Patienten-Grenze

Start Live Meine ARD

Alle Sender Untertitel Audiodeskription Gebärdensprache Kinder

Videos 2

5 Jahre Corona: Als das Virus nach Deutschland kam 4 Min.

Folge 4: Die Arzt-Patienten-Grenze (S02/E04) 47 Min.

5 Jahre Corona: Als das Virus nach Deutschland kam 27.01.2025 Abendschau - BR

Folge 4: Die Arzt-Patienten-Grenze (S02/E04) 28.02.2023 Falk WDR

Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder

b1ja19fa.h1oki4ga

Color: #000000; Fallback for TheSans-Css; Margin: 0px 0px 8px; Font: 20px "TheSans-Css";

Contrast: 1; Name: Folge 4: Die Arzt-Patienten-Grenze (S02/E04); Role: heading; Keyboard-focusable: true;

Relevanz ▾

Styles >

both have the same class “`b1ja19fa h1oki4ga`” .

ok. so back to first example:

ARDmediathek.de/suche/Folge%206%20Die%20Qual%20der%20Wahl

Start Live Meine ARD

Alle Sender Untertitel Audiodeskription Gebärdensprache Kinder

Videos 2

Wer die Wahl hat, hat die Qual - 20.11.12 | Folge 1028 47 Min.

Folge 6: Die Qual der Wahl (S01/E06) 29 Min.

Wer die Wahl hat, hat die Qual - 20.11.12 | Folge 1028 24.02.2023 - Falk - WDR

Folge 6: Die Qual der Wahl (S01/E06) 04.11.2025 - Dahoam is Dahoam - BR

Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder

b1ja19fa.h1oki4ga

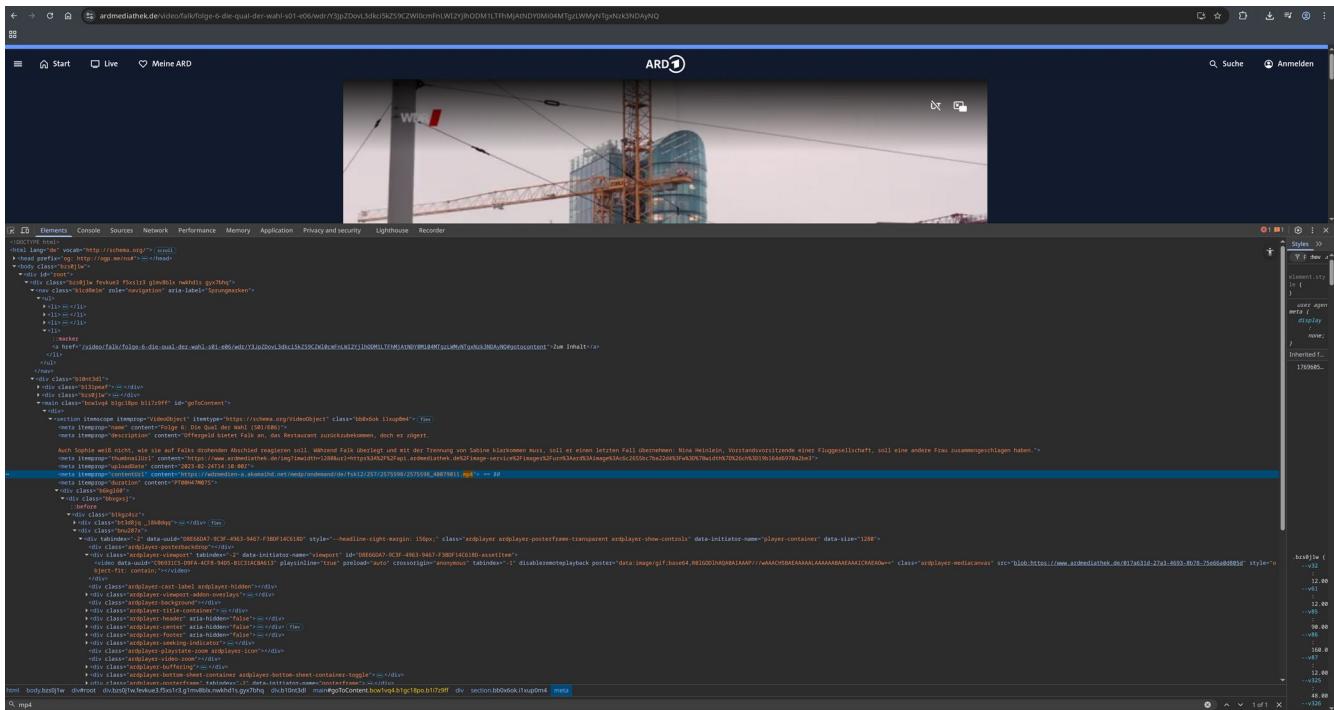
Color: #000000; Fallback for TheSans-Css; Margin: 0px 0px 8px; Font: 20px "TheSans-Css";

Contrast: 1; Name: Folge 6: Die Qual der Wahl (S01/E06); Role: heading; Keyboard-focusable: true;

Relevanz ▾

Styles >

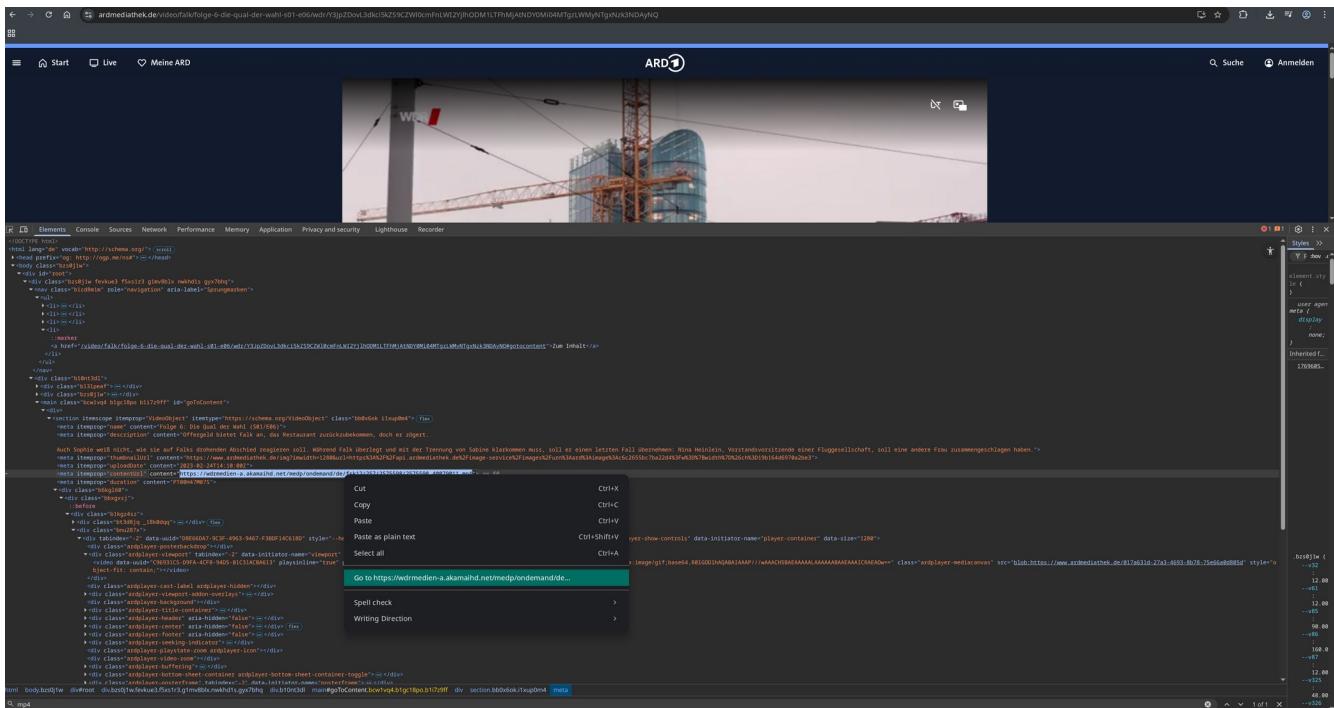
again we can see the same class “`b1ja19fa h1oki4ga`” . Now if we click on this element, we will be redirected to the video.



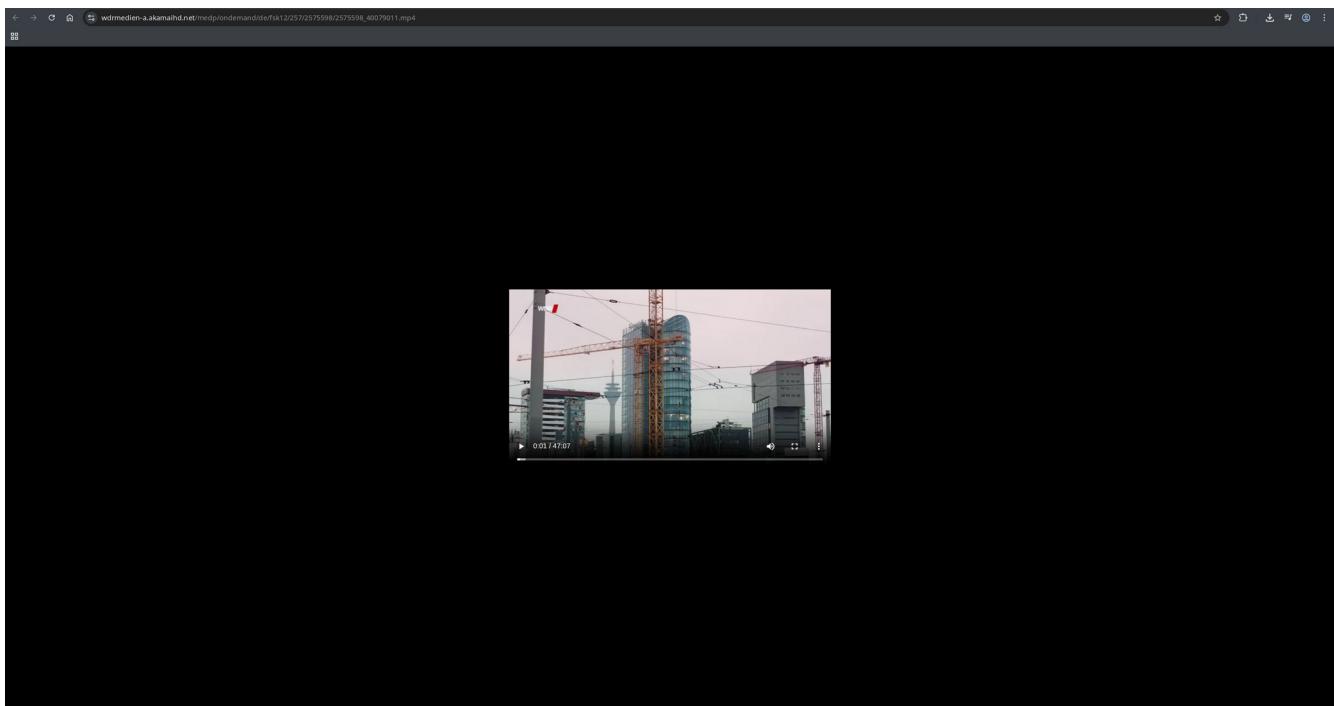
If we click inspect and search for mp4 we see this interesting content with a mp4 extension

<meta itemprop="contentUrl"

content="[https://wdrmedien-a.akamaihd.net/medp/ondemand/de/fsk12/257/2575598/2575598\\_4007901\\_1.mp4](https://wdrmedien-a.akamaihd.net/medp/ondemand/de/fsk12/257/2575598/2575598_4007901_1.mp4)">



we can even go to this link, and we'll be redirected to:



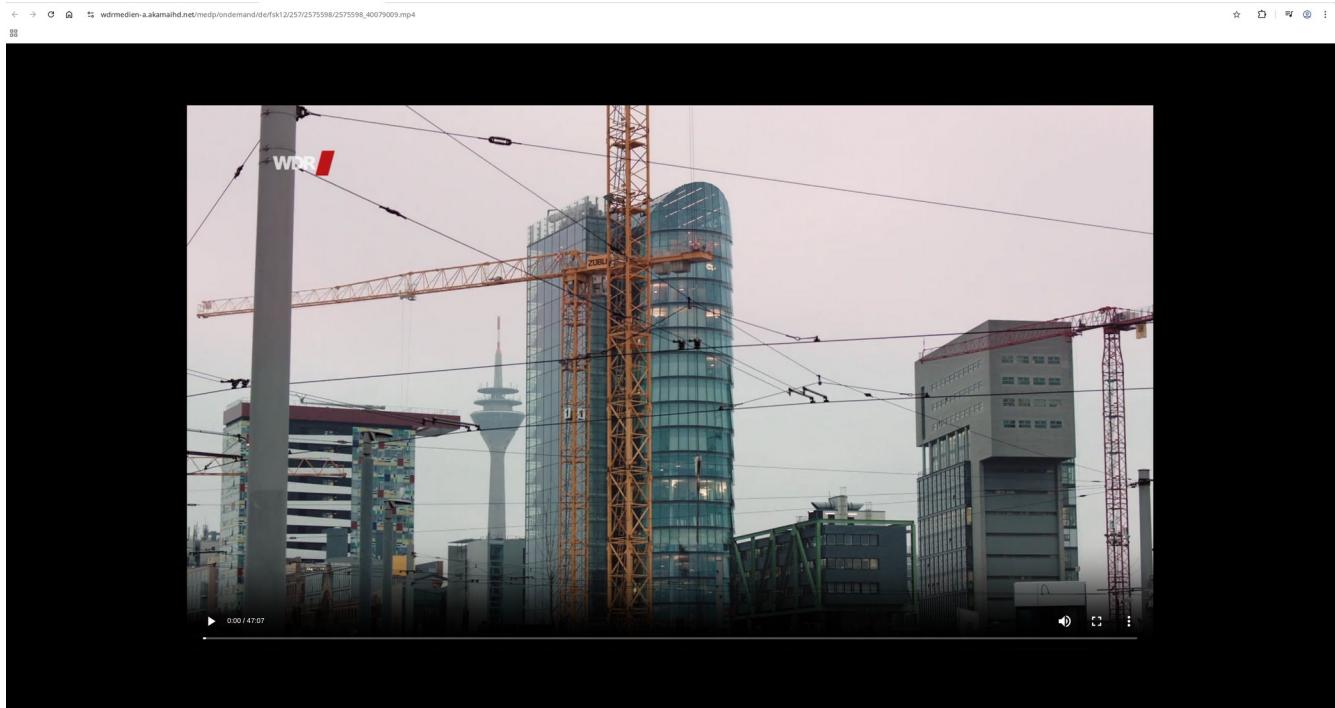
to our video. A video with bad quality though. So what's the problem? Turns out, when we go back, if we didn't set ourselves Settings → HD Quality 1080p, it will be stuck on Auto or last video quality. And here's also an interesting part, it appears that when inspecting again on Element from browser after we set the video quality from Settings, we will see two mp4's!

The screenshot shows the browser's developer tools (Elements tab) with the video player's internal HTML structure. A dropdown menu for 'Qualität' (Quality) is open, showing options: Qualität, Auto, SD 360p, SD 540p, HD 720p, and HD 1080p. The 'HD 1080p' option is highlighted. The background shows the video player interface with a thumbnail of a construction site.

The second video is the quality that we selected from Settings



and if we open the link of the second video, now we get the best quality

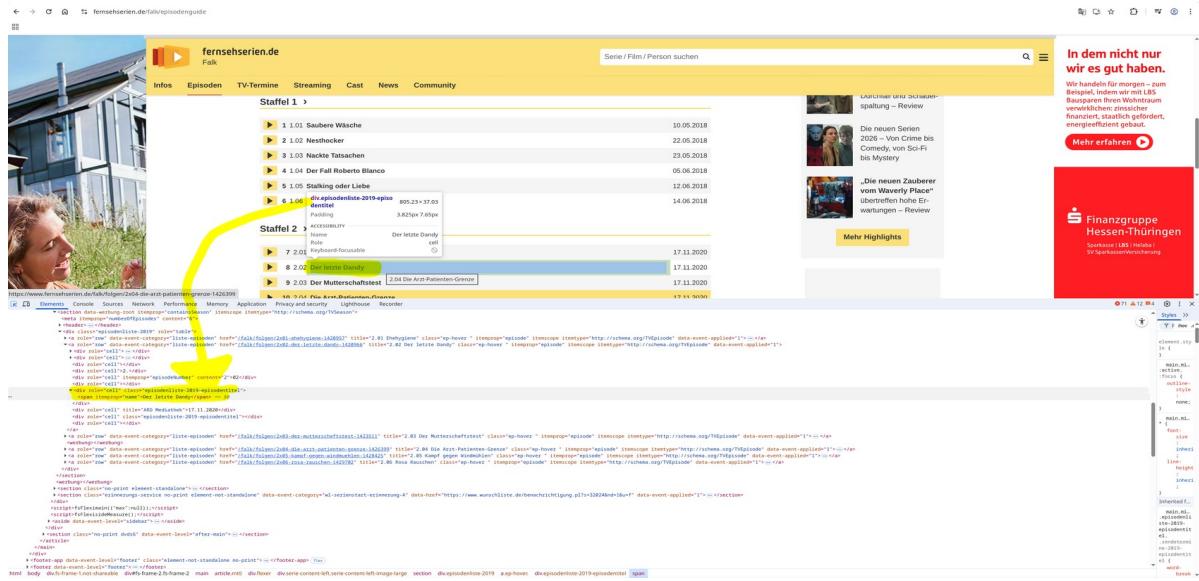


This is what we want to code. We want to webscrap until we find this link and download the videos in a folder.

But it won't work as stated before without the vpn extension on the browser. This is why we use Puppeteer, due to Geolocation and the fact that we can't use a simple fetch api through Node JS in this context.

## Code explanation:

## test.js:



We fetch this URL to get all the episode names. Each episode name is inside a span element that is inside a div with class name .episodenliste-2019-episodentitel. Then we export the function to our second script.

```
const { JSDOM } = require("jsdom");
const { pipeline } = require("node:stream/promises");
const fs = require("node:fs");

const infoURL = "https://www.fernsehserien.de/falk/episodenguide";

async function getAllEpisodeTitles() {
  try {
    const response = await fetch(infoURL);
    if (response.ok) {
      const data = await response.text();
      const dom = new JSDOM(data);
      const document = dom.window.document;

      const divs = document.querySelectorAll(
        ".episodenliste-2019-episodentitel",
      );

      const episodeTitles = [...divs]
        .map((div) => div.querySelector("span")?.textContent.trim())
        .filter(Boolean);

      console.log(episodeTitles);
      return episodeTitles;
    } else {
      console.log("Failed to find titles");
      return [];
    }
  } catch (err) {
    console.error(err);
    return [];
  }
}

module.exports = { getAllEpisodeTitles };
```

## puppet1.js

We export the function from the first script into this script, as well puppeteer module, fs module, path module and pipeline module

```
const downloadDir = path.join(__dirname, "downloadDir");
```

is for telling the script where we want to put the downloaded videos. In this case in **downloadDir**

And now the big **(async ()**

**1)** First we retrieve the episode names by using the function exported from the first script

```
const episodes = await getAllEpisodeTitles();
```

**2)** We explicitly say to puppeteer that we are using this type of browser:

```
const chromeBrowser = {
  browserURL: "http://127.0.0.1:9222",
  defaultViewport: null,
  protocolTimeout: timeout,
  headless: false,
};
```

```
const browser = await puppeteer.connect(chromeBrowser);
```

**3)** inside const download there is an async function that takes **episode** as param

before downloading videos we check if there are already downloaded videos so that they are not download again. If they do not exist, return

```
const filePath = path.join(downloadDir, episodeFile);
console.log(
  filePath,
  fs.existsSync(filePath) || fs.existsSync(`${filePath}.crdownload`),
);
if (fs.existsSync(filePath) || fs.existsSync(`${filePath}.crdownload`))
  return;
```

**4)** we open the page using puppeteer and create a client **CDPSSession** in order to let browser download the video in downloadDir

```
const page = await browser.newPage();
page.setDefaultTimeout(timeout);

const client = await page.target().createCDPSSession();

await client.send("Page.setDownloadBehavior", {
  behavior: "allow",
  downloadPath: downloadDir,
});
```

## 5) This is the automated Manual walktrough explained earlier

```
await page.goto("https://www.ardmediathek.de/");

await page.setViewport({ width: 1024, height: 1024 });

const findSuche = await page.locator('[aria-label="Suche"]').waitHandle();

await findSuche?.click();

const sucheInput = await page.locator("#SearchInputWidget").waitHandle();
await sucheInput?.click();

await page.keyboard.type(`Falk ${episode}`, { delay: 50 });

await page.keyboard.press("Enter", { delay: 50 });

const titleVideo = await page.locator(".b1ja19fa.h1oki4ga").waitHandle();
await titleVideo?.click();

await delayExecutionFor(1000);
const videoSettings = await page
  .locator([
    'button[class="ardplayer-button-settings ardplayer-icon ardplayer-icon-settings ardplayer-icon-settings-hd"]' +
    '[data-initiator-name="addon-button(SettingsSheetAddon:settings)"]' +
    '[tabindex="0"]' +
    '[title="Einstellungen an / aus"]' +
    '[data-display-title="Einstellungen"]' +
    '[aria-pressed="false"]' +
    '[aria-label="Einstellungen an / aus"]',
  ])
  .waitHandle();

await delayExecutionFor(1000);
await videoSettings?.click();

await page
  .locator(
    "span" +
    '[tabindex="0"]' +
    '[role="option"]' +
    '[data-index="4"]' +
    '[data-initiator-name="select(Qualität)=>HD 1080p"]',
    //'[aria-selected="false"]',
  )
  ?.click();

const videoSelector = await page
  .locator('source[type="video/mp4"]')
  .waitHandle();

const videoSRC = await videoSelector?.evaluate((el) => el.src);
```

small time delays were added in order for the page to have time to upload;

6) In const **videoSRC** is the URL with the **mp4** file with the best quality, but we also need to download the video, for that we do this

```
await page.evaluate(  
  (url, episodeFile) => {  
    const a = document.createElement("a");  
    a.href = url;  
    a.download = episodeFile;  
    document.body.appendChild(a);  
    a.click();  
    a.remove();  
  },  
  videoSRC,  
  episodeFile,  
);
```

7) The video is being downloaded and now we just close the page so that browser won't crash when going to the next episode

```
await page.close();  
console.log(`Completed: ${episode}`);  
};
```

8) And we do a for loop

```
//await download(episodes[0]);  
for (const episode of episodes) {  
  await download(episode);  
}  
}());
```

Thank you for coming to coming to my tedtalk  
100% no AI written