

Anleitung: Website mit Lieblings-Alben auf Github

Inhaltsverzeichnis

1	HTML	2
1.1	Erklärung HTML	2
2	GitHub Account und Projekt erstellen.....	3
2.1	Erstellen eines GitHub Accounts	3
2.2	Webseiten Grundgerüst einrichten.....	4
3	Erstellen der Webseite	6
3.1	Einführung	6
3.2	Grundgerüst HTML	7
3.3	Style	7
3.4	Alben erstellen	9
3.5	Programmieren mit JavaScript.....	12



HTML

Hyper
Text
Markup
Language

1.1 Erklärung HTML

Mit HTML stellst du Informationen in deinem Webbrowser (Firefox, Chrome ...) dar. HTML ist keine Programmiersprache, sondern eine MARKUP Sprache.

Eine Programmiersprache ist z.B. Java, PHP ... Mit einer Programmiersprache kannst du sagen was passieren soll, wenn man auf einen bestimmten Knopf klickt etc.

Das Grundgerüst eines HTML-Files sieht wie folgt aus:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Titel der Website</title>
  </head>
  <body>
    <h1>Titel</h1>
  </body>
</html>
```

- `<!DOCTYPE html>` : sagt dem Browser, dass es ein HTML File ist
- `<html>` : alles innerhalb dieses Tags ist auf HTML geschrieben
- `<body>` : hier kommt der ganze Inhalt d.h. der Text, die Bilder etc. hinein
- `<head>` : hier kommen der Titel der Website, der wird oben auf dem Tab angezeigt und die Zeichenkodierung vor.
- `<h1>` : Titel der auf der Website selber angezeigt wird
Es gibt auch `<h2>`, `<h3>` und noch einige mit denen man Untertitel schreiben kann. Je grösser die Zahl desto kleiner der Titel.
- `<p>` : mit diesem Tag werden Texte dargestellt
- `` : der Text innerhalb dieses Tags wird fett geschrieben
Z.B. `<p>Website über unsere Lieblings-Alben</p>` → Website über unsere **Lieblings-Alben**
- `<div>` : ist ein Bereich, alles das innerhalb des div-Tags steht wird in einem gemeinsamen Bereich eingeschlossen. Das braucht man um z.B. zwei Sachen nebeneinander anzuzeigen
- `` : zeigt ein Bild an, im Tag selber steht 'src=' dort gibt man den Pfad zum Bild ein.
- `` : automatisch nummerierte Liste
- `` : nicht nummerierte Liste
- `` : hier steht der Inhalt eines Elementes der Liste
Zum Beispiel:

```
<ol>
  <li>Lucky</li>
  <li>The Tourist</li>
</ol>
```

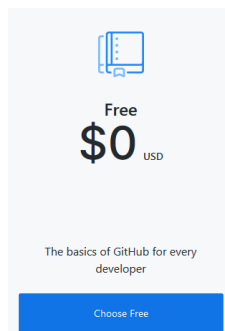
1. Lucky
2. The Tourist

2 GitHub Account und Projekt erstellen

2.1 Erstellen eines GitHub Accounts

Das Erstellen eines GitHub Accounts auf github.com ist simpel und wie auf fast jeder anderen Seite auch.

Eine wichtige Information vorab, der Benutzername wird im Webseitenname sein.



Für das weitere Erstellen des Accounts muss ein Plan ausgewählt werden. Der kostenlose Plan reicht für unsere Absichten völlig aus.

Join GitHub

Create your account

Username *
anietung ✓

Email address *
anietung@anietung.anietung ✓

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Email preferences
☐ Send me occasional product updates, announcements, and offers.

Verify your account

When the image is the correct way up touch Done!

Done

Next: Select a plan



How much programming experience do you have?

None
I don't program at all

A little
I'm new to programming

A moderate amount
I'm somewhat experienced

A lot
I'm very experienced

What do you plan to use GitHub for?
(Select up to 3)

Learn to code

Learn Git and GitHub

Host a project (repository)

Create a website with GitHub Pages

Collaborating with my team

Find and contribute to open source

School work and student projects

Use the GitHub API

Other

I am interested in:

Languages, frameworks, industries

We'll connect you with communities and projects that fit your interests.
For example: `webpack` `game-off` `ajp11a`

Complete setup

Um das Setup zu beenden kann die Programmier-Erfahrung und der persönliche Zweck von GitHub angegeben werden. Dieser Schritt ist jedoch nicht notwendig.

2.2 Webseiten Grundgerüst einrichten


Beim Erstellen der Webseite **muss darauf geachtet werden, dass der Name des «Repository» den Benutzernamen und «.github.io» enthält**. Wenn der Benutzername z.B. «anleitung» ist, muss der «Repository name» anleitung.github.io sein.

Der Rest ist bereits standardmässig eingestellt und muss nicht geändert werden.

Mit «Create repository» erstellt man nun ein neues «Repository», das als Webseite gebraucht werden kann.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner	Repository name *
 anleitung ▾	/ anleitung.github.io ✓

Great repository names are short and memorable. Need inspiration? How about [silver-disco](#)?

Description (optional)

- ☒  **Public**
Anyone can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

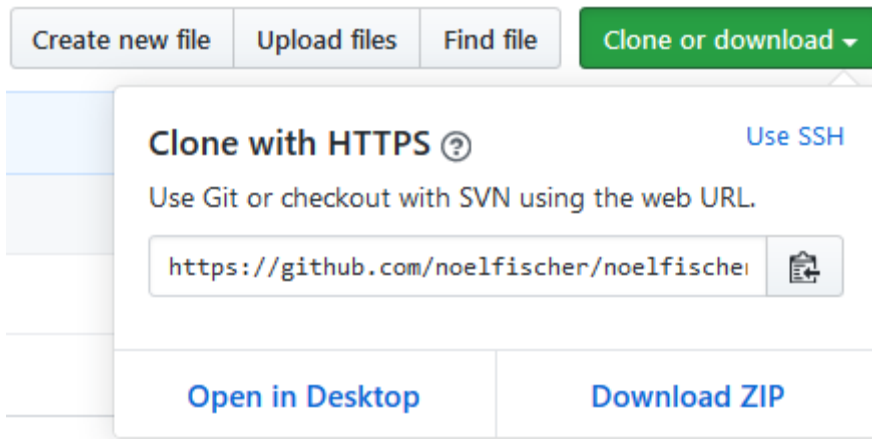
Skip this step if you're importing an existing repository.

- ☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾	Add a license: None ▾ ⓘ
-------------------------------	--------------------------------

Create repository

Um Visual Studio Code mit GitHub zu verknüpfen, muss man Git von [dieser Seite](#) herunterladen und installieren.



Auf der GitHub Seite klickt man nun auf «Clone or download» und kopiert die URL.

In Visual Studio Code gibt man nun den Befehl «git clone <Deine URL>» ein.

Damit man mit Git arbeiten kann, gibt es verschiedene Befehle, die im Terminal eingegeben werden können:

Um den Status abzurufen «git status»

Um neue Dateien zu Git hinzuzufügen: «git add *»

Um Dateien zu speichern «git commit -m «some message»»

Um Dateien auf den Server zu laden «git push»

3 Erstellen der Webseite

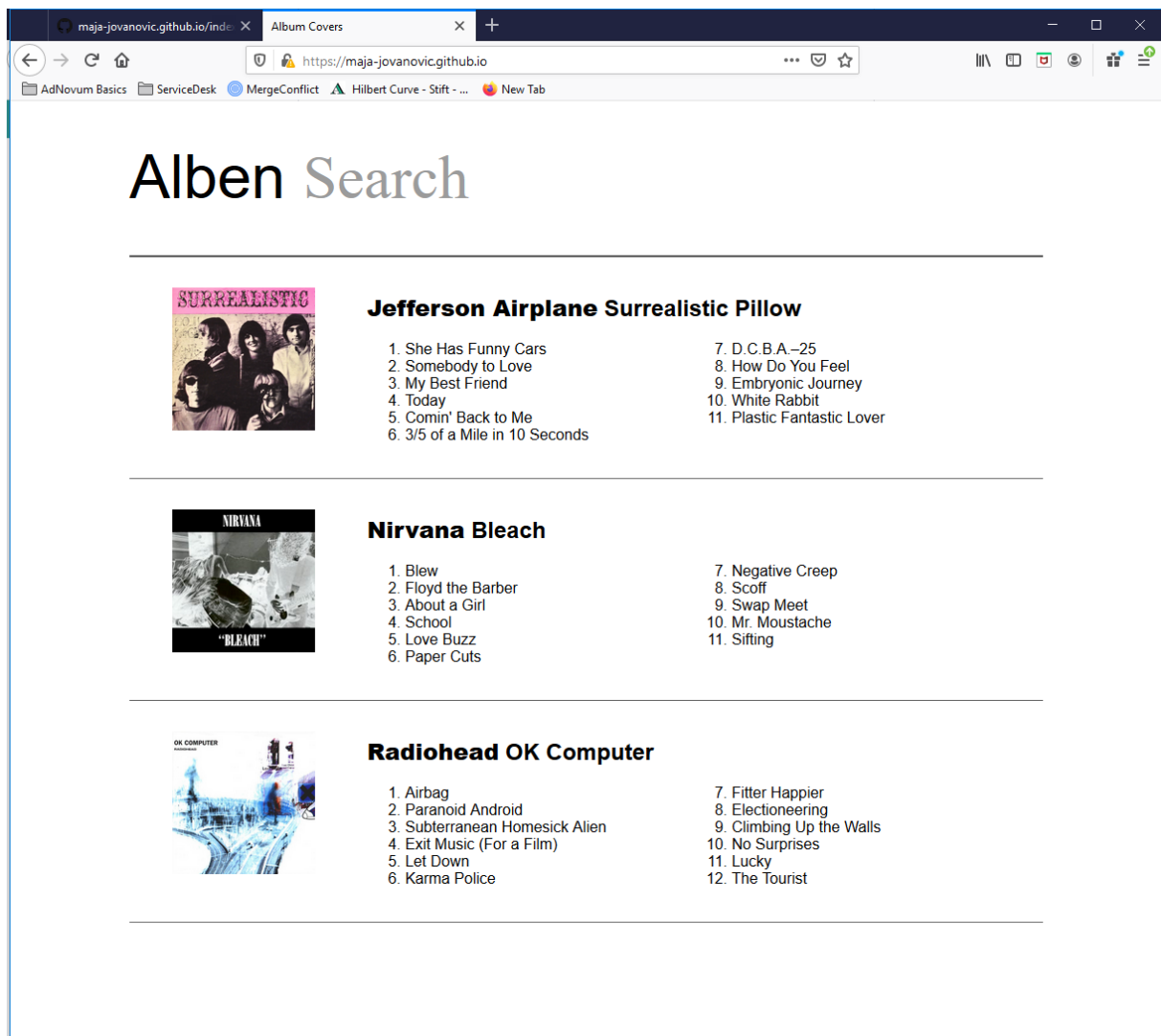
3.1 Einführung

Auf dieser Website stellst du deine Lieblings-Alben dar.

Pro Album brauchst du:

- ein Bild des Albums
- den Namen des Albums
- den Namen des Interpreten
- die Song Liste

Es gibt auch ein Suchfeld, indem du alles Suchen kannst, z.B. welche Alben zu einem Interpreten gehören oder welches Lied zu welchem Album gehört. Am Ende sollte die Website etwa so aussehen.



Die Informationen schreiben wir mit HTML, die Website stellen wir mit CSS dar und die Suche programmieren wir mit JavaScript.

Vergesse nicht! Nach jeder Änderung musst du erneut das File zuerst den Befehl git add, dann git commit -m «änderung» und am Ende pushen.

3.2 Grundgerüst HTML

Wenn du dich im Projekt befindest, das du von GitHub geklont hast, erstellst du dort ein neues File namens `index.html`. Danach schreibst du das Grundgerüst eines HTML-Files, das wie folgt aussieht.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Album Covers</title>
  </head>
  <body>
    <h1>Alben</h1>
  </body>
</html>
```

Nun kannst du es committen. Dies machst du mit dem Terminal. Als erstes musst du das File Lokal hinzufügen und dann Committen. Damit wir es sehen können, musst du es noch auf github pushen.

```
git add *
git commit -m "Add index.html"
git push
```

Nun können wir unsere Website mit «username.github.io» aufrufen. Wenn du alles richtig gemacht hast, sollte deine Webseite nun so aussehen.

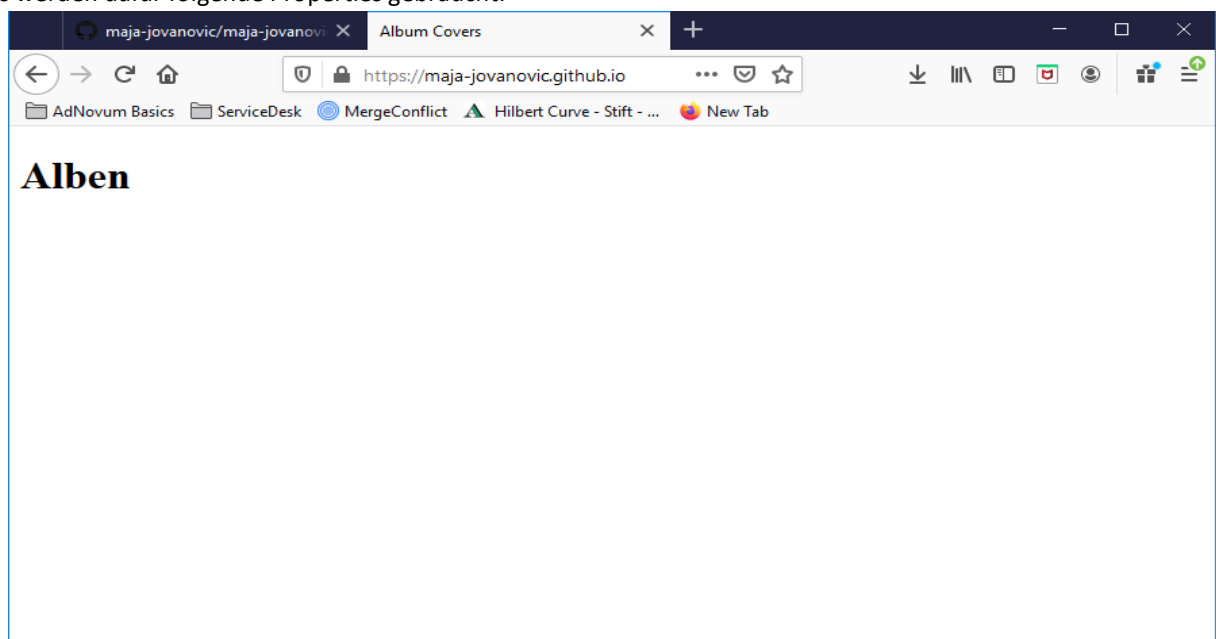
Die Suchfunktion werden wir erst am Ende machen.

3.3 Style

Als nächstes erstellst du ein neues File als Stylesheet und benennst es «index.css».

Dann kannst du den Haupt-Wrapper «pagecontent» stylen. Die Idee ist, dass die Seite bis zu einer bestimmten Grösse immer 100% der Seite ausfüllt. Wenn die bestimmte Seitengrösse erreicht ist, sollte der Wrapper seine Breite halten und immer zentriert dargestellt werden.

Es werden dafür folgende Properties gebraucht:



- **width:** Damit wird die Breite angegeben
- **max-width:** Damit wird eine maximale Breite definiert. Diese überschreibt *width*
- **margin:** Damit wird der äussere Abstand vom Wrapper definiert. Mit dem Wert *auto* kann man ein div zentrieren

```
.pagecontent {  
  width: 98%;  
  max-width: 960px;  
  margin: auto;  
}
```

Dann kommen wir zu einem weiteren, eher allgemeinen Punkt und zwar der Titel. Der Titel wird mit einem '`<h1>`' Tag definiert. Standardmässig wird er *fett* dargestellt, jedoch ist es in diesem Fall passender, wenn er dünner angezeigt wird. Die Schrift Dicke wird mit dem Property *font-weight* definiert. Dabei kommen Zahlenwerte zum Einsatz:

- 300 Light (dünn)
- 400 Regular (normal)
- 500 - 600 Medium (zwischen normal und fett)
- 600 - 700 Bold (fett)

In diesem Beispiel brauchen wir somit *300*.

```
h1{  
  font-weight: 300;  
  font-size: 48pt;  
}
```

Bei der Klasse *.line*, welche wie der Name schon sagt eine Linie zeichnet können 3 Einstellungen gemacht werden:

1. Liniendicke
2. Art der Linie (durchgehend, gestreift etc.)
3. Farbe

```
.line{  
  width: 100%;  
  border-bottom: 2px solid #4C4C4C;  
}
```

Nun kannst du das File committen. Jetzt müssen wir noch das Stylesheet mit dem HTML File verbinden und die Klassen verwenden. Im '`<head>`' Tag schreibst du nach dem '`<title>`' Tag einen neuen für das Stylesheet.

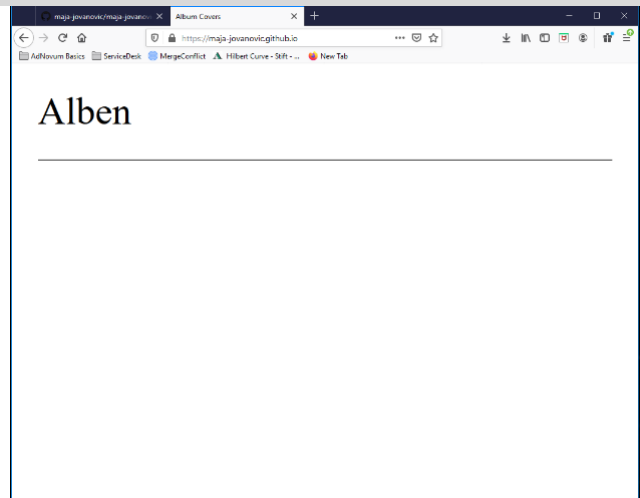
```
<link href="index.css" media="screen" rel="stylesheet" type="text/css"/>
```

Mit dem '`<link>`' Tag verbindest du weitere Files mit diesem (index.html). «href» steht für den Namen des Files bzw. den Pfad ab dem befindenden Ordner zum File.

Nun kannst du auf die Klassen zugreifen. Dazu machst du noch um den ganzen Inhalt des '`<body>`' Tags ein '`<div>`' mit der Klasse «pagecontent». Und schreibst nach dem Titel ('`<h1>`') noch einen '`<div>`' für die Linie.

3.4 Alben erstellen

```
...
<body>
    <div class="pagecontent">
        <h1>Alben</h1>
        <div
class="line"></div>
        </div>
    </body>
...
```

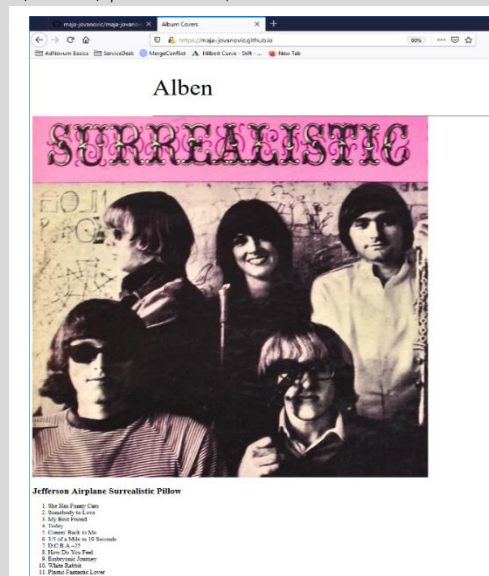


Jetzt sieht die Seite doch schon viel schöner aus. Nun kommt der wichtigste Teil, das Album. Als erstes fügst du im '`<body>`' Tag nach dem '`<div class="line"></div>`' das Bild hinzu, danach den Namen des Interpreten und den Namen des Albums und am Ende die Liste der Songs. Z.B. :

```

```

```
<h2><b>Jefferson Airplane</b> Surrealistic Pillow</h2>
<ol>
    <li>She Has Funny Cars</li>
    <li>Somebody to Love</li>
    <li>My Best Friend</li>
    <li>Today</li>
    <li>Comin' Back to Me</li>
    <li>3/5 of a Mile in 10 Seconds</li>
    <li>D.C.B.A.-25</li>
    <li>How Do You Feel</li>
    <li>Embryonic Journey</li>
    <li>White Rabbit</li>
    <li>Plastic Fantastic Lover</li>
</ol>
...
```



'``' steht für eine nummerierte Liste, wenn du eine nicht nummerierte Liste möchtest, schreibst du '``' anstatt '``'. Im '``' Tag steht bei uns jeweils der Song.

Der nächste Schritt ist die Darstellung anzupassen. Das Bild sollte kleiner und die Songs schöner dargestellt bzw. übersichtlicher sein. Dazu erstellst du als erstes wieder neue Klassen im `index.css` und darauffolgend bearbeitest du wieder das `index.html`.

Index.css:

```
...
.image-wrapper {
    width: 25%;
    float: left;
}
```

Der Image Wrapper sollte 25% der Breite ausmachen. Damit 2 '`<div>`'s nebeneinander positioniert werden können, muss man mit **float** arbeiten, so kann man ein '`<div>`' entweder links oder rechts positionieren.

```
.cover-image {
  max-width: 150px;
  width: 90%;
  padding: 5%;
  margin: auto;
  margin-top: 20px;
  display: block;
}
```

Das Bild sollte nicht die ganze Breite des Bereiches ausfüllen sondern nur 90%.
Es sollte oberhalb noch Platz haben und nicht an der oberen Linie kleben (margin-top).

```
.text-wrapper {
  width: 74%;
  float: right;
  margin-top: 20px;
  padding-bottom: 20px;
}
```

Der Text sollte sich auf der rechten Seite befinden und ebenfalls oben und unterhalb Platz haben.

Float hat auch einen Nachteil, und zwar bringt diese Einstellung alles auf der Seite durcheinander, was sich gerade unter den mit Float gestylten Objekten befindet. Verhindern kann man das mit **clear**.

```
.clear{
  clear: both;
}
```

Die Klasse .line ist die Basisklasse. Wenn eine .line gemacht wird, wird das sowieso angezeigt. Wenn man jedoch die .light Klasse hinzufügt, wird der Teil, der sich überschneidet überschrieben. D.h. die Breite wird übernommen und der **border** überschrieben.

Diese Linie sollte heller und dünner sein.

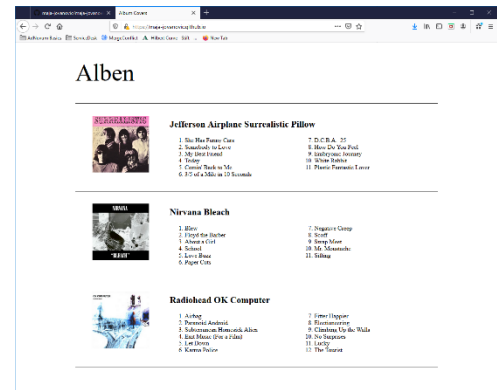
```
.line.light {
  border-bottom: 1px solid #626262;
}
```

Index.html:

```
...
<body>
  <div class="pagecontent">
    <h1>Alben</h1>
    <div class="line"></div>

    <div>
      <div class="image-wrapper">
        
      </div>
      <div class="text-wrapper">
        <h2><b>Jefferson Airplane</b> Surrealistic Pillow</h2>
        <ol>
          <li>She Has Funny Cars</li>
          ...
        </ol>
      </div>
      <div class="clear"></div>
      <div class="line light"></div>
    </div>
  </body>
...
```

Jetzt ist die Seite auch schön formatiert. Nun kannst du alles innerhalb und inklusive den grünen <div> Tags wiederholen und nur das Bild, den Namen des Interpreten, den Namen des Albums und die Song-Liste ändern. Die Seite sollte nun so ähnlich aussehen.



3.5 Programmieren mit JavaScript

Das letzte File ist das Script. Dazu erstellst du ein neues File und benennst es «index.js».

Für die Suche wird ein Suchfeld benötigt. In HTML sieht ein Textfeld, welches wir für die Suche benötigen folgendermassen aus: <input>. Das Input-Feld braucht folgende Parameter

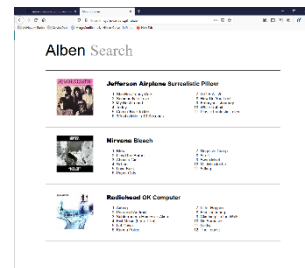
- id = "search"
- placeholder = "Search"
- type = "text"

und wird im Titel platziert:

```
<h1>Alben <input id="search" type="text" placeholder="Search"/></h1>
```

Damit die Suche auch gut aussieht, werden auch noch einige Zeilen CSS benötigt:

```
input {
  border: 0px;
  background-color: transparent;
  font-size: 48pt;
  width: 600px;
  font-family: "Helvetica Neue";
  color: #444;
  font-weight: 300;
}
```



Die Suche wird in JavaScript jQuery geschrieben. Es wird eine Funktion benötigt, welche immer dann ausgeführt wird, wenn sich im Textfeld etwas verändert:

```
$(document).ready(function() {
  $("#search").keyup(function() {
    ...
  });
});
```

Zu Beginn müssen alle Alben *eingesammelt* werden. Auch sollten sie gleich alle angezeigt werden:

```
var albums = $(".album");
$(albums).show();
```

Dann wird der Suchbegriff ausgelesen:

```
var searchTerm = $(this).val().toLocaleLowerCase()
```

Dann muss durch alle Alben geloopt und überprüft werden, ob der Suchbegriff vorkommt. Wenn er nicht vorkommt sollte dieses Album nicht mehr angezeigt werden:

```
albums.each(function(){
  if($(this).text().toLocaleLowerCase().search(searchTerm) == -1) {
    $(this).hide();
  }
});
```

Das File sieht am Ende so aus.

```
1  $(document).ready(function() {
2      $("#search").keyup(function() {
3          var albums = $(".album");
4          $(albums).show();
5
6          var searchTerm = $(this).val().toLocaleLowerCase();
7
8          albums.each(function(){
9              if ($(this).text().toLocaleLowerCase().search(searchTerm) == -1) {
10                  $(this).hide();
11              }
12          });
13      });
14  });
```

Als letztes müssen wir dieses File mit «index.html» verbinden.

Dies macht man mit dem folgenden Tag im <head> nach dem <link href ...>.

```
...
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.slim.min.js"
    integrity="sha256-pasqAKBDmFT4eHoN2ndd6lN370kFiGUfYtIUHWhU7k8="
    crossorigin="anonymous">
</script>
<script src="index.js"></script>
...
```