31.05.2019

Interaktiver Liederspaß

Dokumentation des Einzelprojekts in der Übung
„Wunderbare Welt der musikalischen Akustik: Synthese“

Amelie Schneider
11715842

INHALT

1 ZIELSETZUNG

Ziel dieses Projekts war es, einen Audioplayer zu erstellen, bei dem man ein selbst hochgeladenes Lied mit Sounds erweitern kann. Diese Sounds sollten mithilfe von Buttons abgespielt werden, die in Form von dazu passenden Abbildungen rund um den Audioplayer zu finden sind.

1.1 VERWENDETE PROGRAMME

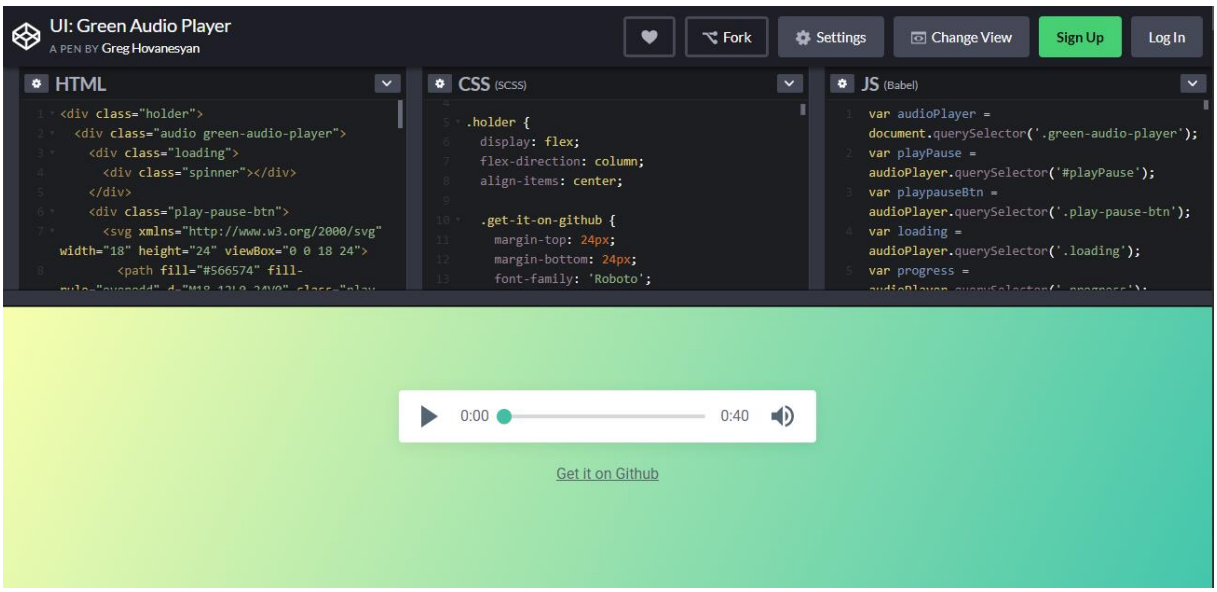
Für das Projekt habe ich primär HTML, CSS und JavaScript in der Softwareumgebung *Brackets* verwendet. Zusätzlich habe ich ein Bild mit GIMP zugeschnitten und die Seiten http://www.youtube2mp3.de/ und https://mp3cut.net/de/ verwendet, um manche Sounds im mp3-Format zu gewinnen und zuzuschneiden. Einzelne Sounds waren zu leise, also habe ich auf http://www.mp3louder.com/ die Lautstärke erhöht.

2 GRUNDLAGEN

Als Basis für mein Projekt habe ich die folgenden zwei Codeausschnitte verwendet:

2.1 UI: GREEN AUDIO PLAYER

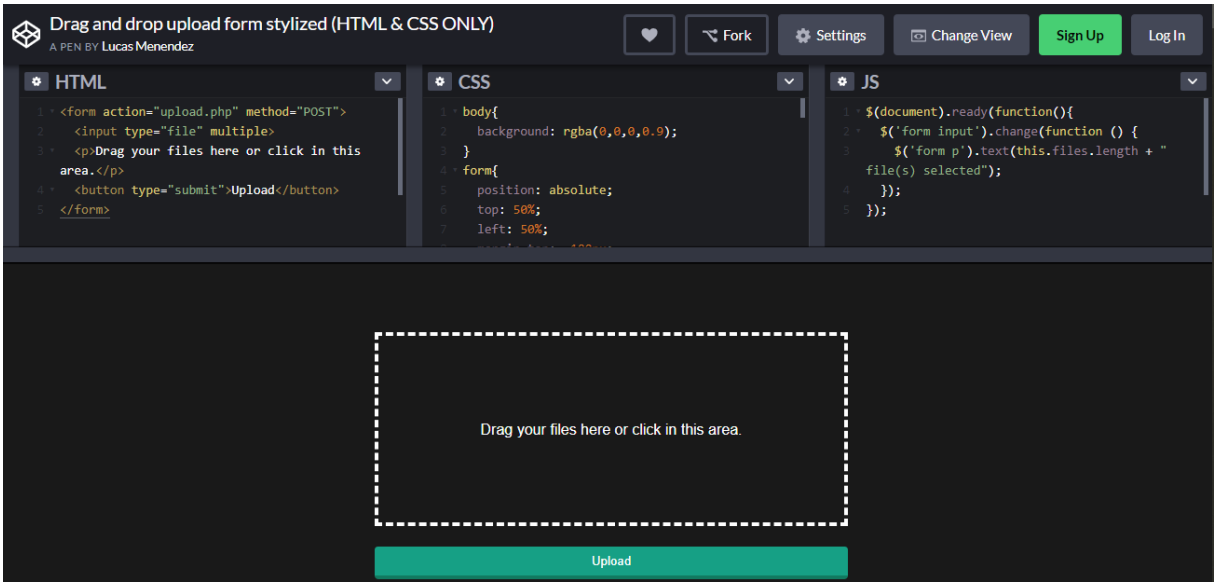
https://codepen.io/gregh/pen/NdVvbm



Mit diesem Code ist es möglich, eine bereits vorhandene Sounddatei abzuspielen. Der CSS-Code liefert bereits eine gute Grundlage für das spätere Layout. Ebenfalls ist bei dem Player bereits ein Lautstärkeregler eingebaut.

2.2 Drag and drop upload form stylized (HTML & CSS ONLY)

https://codepen.io/TheLukasWeb/pen/qlGDa



Dieser Code lieferte die Grundlage für das Drag-and-Drop-Feld, in das man später die Audiodateien einfügen sollte, die anschließend mit dem Audioplayer abgespielt werden. Auch hier wurde das Layout des Feldes übernommen, jedoch der Upload-Button entfernt und das Feld in der Höhe etwas schmäler gemacht.

3 Erstellen des Projekts

3.1 Verbinden der Codes

Der erste Schritt in meinem Projekt war das Verbinden der beiden Codesegmente, sodass sie auf einer Seite Platz finden und miteinander funktionieren. Hierzu bin ich vom Code des Audioplayers ausgegangen und habe schließlich einzelne Teile des Drag-and-Drop-Feld-Codes eingebaut, sodass nur jene Teile im Projekt sind, die ich benötige. Der nächste Schritt war es, den CSS-Code insofern zu bearbeiten, dass beide Elemente am Bildschirm Platz finden und ein stimmiges Gesamtbild erzeugen.

Der Code des Drag-and-Drop-Felds ist zum Teil in jQuery geschrieben, also habe ich das Skript mithilfe des folgenden Codes in die HTML-Datei eingebunden:

```
<script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
crossorigin="anonymous"></script>
```

Der Button beim Drag-and-Drop-Feld, sowie der Link zu GitHub beim Audioplayer-Code waren nicht notwendig, somit konnten diese Teile weggelassen werden.

3.2 Drag-and-Drop Feld

3.2.1 Erste Konfigurationen

Zunächst sollte das Feld kleiner sein und beim „Hovern“ der Mauszeiger zu einem Pointer werden. Dazu diente der folgende Code:

```
6  form {
7      width: 440px;
8      height: 100px;
9      margin-bottom: 24px;
10     border: 4px dashed #fff;
11     position: relative;
12     text-align: center;
13     cursor: pointer;
14 }
```

3.2.2 Beschriftung

Als nächstes habe ich den Text im Drag-and-Drop Feld verändert, beziehungsweise als Label klassifiziert. Dazu der folgende Code:

```
46     <form action="upload.php" method="POST">
47         <input id="fileupload" type="file" accept="audio/mp3,audio/*;capture=microphone" multiple placeholder="Gib mir dein Lied!" />
48         <label for="fileupload">Gib mir dein Lied!</label>
49     </form>
```

Dann kam die Schwierigkeit auf, das Label im Feld zu zentrieren. Dies habe ich mit folgendem Code gemacht:

```
16 form label {
17     text-align: center;
18     color: #ffffff;
19     font-family: Arial;
20     cursor: pointer;
21     position: relative;
22     bottom: 55%
23 }
```

Nachteil dieser Lösung ist, dass wenn eine Datei mehr als eine Zeile hat, der Text nicht mehr zentriert ist. Hier kommen wir auch schon zu dem nächsten Punkt.

3.2.3 Ändern der Beschriftung bei Dateieinfügen

Wenn die Datei im Feld gedroppt wurde, sollte statt „Gib mir dein Lied!“ der Dateiname stehen. Dazu habe ich folgenden Code verwendet:

```
105    <script>
106        $(document).ready(function() {
107            $('form input').change(function() {
108                $('form label').text(this.files[0].name);
109            });
110        });
111    </script>
```

3.2.4 Beschränkung auf Audiodateien

Der nächste Schritt war es, das Drag-and-Drop Feld lediglich für Audiodateien verwendbar zu machen. Hier ist es mir nur gelungen, bei Klicken auf das Feld nur Audiodateien auswählbar zu machen. Zieht man eine Datei mit Drag-and-Drop in das Feld, so wird diese Datei trotzdem akzeptiert. Der dazugehörige Code ist unter dem Punkt 3.2.2. Beschriftung zu sehen (Zeile 47).

3.3 AUDIOPLAYER

Nun sollte der Audioplayer die gedroppte Audiodatei als Quelle anerkennen. Dazu der folgende Code:

```
86              <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
87
88 ▼            <audio id="lied">
89                  <source src="" id="quelle" />
90              </audio>
```

```
183 ▼ function handleFiles(event) {
184       var files = event.target.files;
185       $("#quelle").attr("src", URL.createObjectURL(files[0]));
186       document.getElementById("lied").load();
187   }
188
189   document.getElementById("fileupload").addEventListener("change", handleFiles, false);
190
```

Der Audioplayer hatte ursprünglich ein Ladesymbol eingebaut, solange noch kein abzuspielender Song erkannt wurde. Dies war in diesem Falle nicht notwendig, also wurde der dafür zuständige Teil entfernt.

3.4 BUTTONS

Jetzt war es bereits an der Zeit, die Buttons hinzuzufügen. Dazu benützte ich zunächst GIMP, um den Jodler auszuschneiden, später habe ich fertige PNG Dateien verwendet.

Zuerst sollten also die einzelnen Bilder eingefügt werden. Dann musste jedes Bild mit einem Sound belegt werden, der auch öfter hintereinander abgespielt werden kann und jedes Mal neu lädt, anstatt erst fertig spielen zu müssen oder von sich selbst überlagert werden kann. Dazu der Code *audio.load();*. Die Sounds wurden aus Sample-Libraries und YouTube entnommen und zu mp3 konvertiert oder selbst aufgenommen. Bei Klicken auf den Button oder in dessen Nähe sollte nun der Sound gespielt werden. Das passiert mit folgendem Code (am Beispiel „doggo“):

```
13      <img id="doggo" src="doggo.png" value="PLAY" onclick="play('dog')">
14      <audio id="dog" src="woof.mp3"></audio>
```

```
 95 ▼      <script>
 96 ▼          function play(sound) {
 97                var audio = document.getElementById(sound);
 98                audio.load();
 99                audio.play();
100           }
101       </script>
```

Jetzt habe ich noch das Bild rotiert, die Position des Buttons festgelegt und den Mauszeiger als Pointer konfiguriert. Dazu der folgende Code:

```
195 ▼ #doggo {
196       position: absolute;
197       right: 70px;
198       top: 20px;
199       height: 35%;
200       transform: rotate(4deg);
201       cursor: pointer;
202   }
```

Zuletzt wurde das noch mit anderen Buttons und Sounds wiederholt, teilweise die Bilder rotiert, der Hintergrund geändert und schon ist der „Interaktive Liederspaß“ fertig, funktional und wunderschön!