Technische Dokumentation

Responsive Webapplikation “Datenklang”

AV PRG WS 2018 - Florian Wiekhorst, Tien Huy Tonny Van

Einleitung

Unser Projekt setzt sich aus **Front-End** und **Back-End** zusammen. Zum Front-end gehören alle HTML und CSS Dateien. Diese stellen den sichtbaren Teil der Webapplikation dar. Mittels PHP und JavaScript wird der Backend-Anteil realisiert. Die Datenübertragung vom Benutzer übernimmt dabei der PHP Code. JavaScript verwenden wir um Veränderungen der sichtbaren Elemente auf der Seite zu erzeugen und um die Web Audio API anzusprechen. Da die GitHub-Pages nur statisch sind verwenden wir ein gemietetes Webhosting-Paket von der Firma “*Strato*”.

Software

Als Entwicklungsumgebungen nutzten wir “*Atom*” und “*Visual Studio Code*”, welche beide auf dem Framework ”*Electron*“ der GitHub-Entwickler aufbauen. Für die Dateiübertragung mittels FTP auf den Webserver von *Strato* verwendeten wir “*File Zilla*”. Die Grafiken auf der Webapplikation, sowie das Plakat erstellten wir mit “*Gimp*” sowie “*Photoshop*”.

Abb. 1: Verwendete Software

Aufbau

Unsere Gliederung sollte möglichst übersichtlich und nutzerfreundlich werden. Das Grundgerüst besteht aus zwei PHP-Seiten, welche jeweils ihre eigene CSS-Datei besitzen, und einer JavaScript-Datei.
Die index.php ist natürlich die Startseite. Hat der Benutzer das komplette Formular mit sinnvollen Eingaben ausgefüllt, wird er per Klick auf den Bestätigungsschalter auf die data.php weitergeleitet. Zusätzlich werden per PHP-Code seine Angaben mitgeschickt und an die music.js gesendet. Innerhalb der music.js befindet sich dann der Code für die Web Audio API, welcher mit den Benutzerangaben aus der index.php Musik erzeugt, und diese zurück an die data.php sendet.

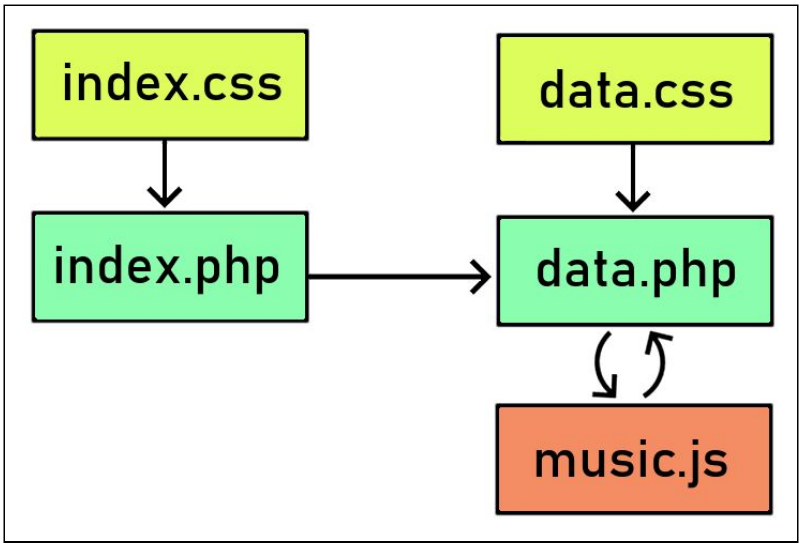

Abb. 2: Aufbau der Webapplikation

(Die Web-App wäre auch mit nur jeweils einer PHP- und CSS-Datei möglich, sodass man insgesamt nur drei Grunddateien hätte. Da wir jedoch zeitgleich verschiedene Arbeiten am Quellcode durchgeführt haben, haben wir die Aufgaben der Web-App auf zwei Seiten aufgeteilt).

Front-End

Die erste Seite ist gegliedert in: Logo, Eingabefelder und Bestätigungsschalter. Das Logo ist zugleich der Produktname.

Abb. 3: Das Datenklang Logo

Die Eingabefelder werden mittels *Bootstrap* und der internen Klasse ‘Form Group’ erzeugt. So stellen wir sicher, dass die Webapplikation responsive auf allen Endgeräten dargestellt

wird. Über prozentuale Werte in den CSS-Dateien behalten z. B. die Texteingabefelder für Vorname und Geburtsort immer eine feste Breite, statt bei den stark abweichenden Pixelbreiten der verschiedenen Geräte über den Rand hinauszuwachsen, oder zu klein zu wirken.

In den CSS-Dateien werden auch die Werte für Farben, Typografie, Abstände der Elemente untereinander und zu den Rändern des Bildschirms, usw. bestimmt.
Halt alle “Style”-Parameter. (CSS steht ja auch für “Cascading Style Sheets“).

Zusätzlich sorgen die “required”-Parameter innerhalb des Bootstraps Formulares dafür, dass der Benutzer die Eingabefelder nicht leer abschicken kann.

Back-End

Innerhalb der PHP-Seiten befindet sich neben dem HTML-Anteil auch der PHP-Code. Hier haben wir für jedes Eingabefeld aus dem Formular eine leere Variable mit entsprechendem Namen angelegt. Sobald das Formular über den Bestätigungsschalter abgeschickt wird, erhalten die Variablen den entsprechenden Wert mithilfe der “POST”-Methode.

```php
<?php
$name = $birthPlace = $gender = $eyeColor = $birthYear = $birthMonth = $birthDay = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = user_input($_POST["name"]);
  $birthPlace = user_input($_POST["birthPlace"]);
  $eyeColor = user_input($_POST["eyeColor"]);
  $gender = user_input($_POST["gender"]);
  $birthYear = user_input($_POST["birthYear"]);
  $birthMonth = user_input($_POST["birthMonth"]);
  $birthDay = user_input($_POST["birthDay"]);
}
function user_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Abb. 4: PHP-Variablen

Sobald die Variablen auf der zweiten Seite (data.php) ankommen, müssen diese dann nochmal per JavaScript-Code innerhalb des HTML-Codes an die music.js weitergeleitet werden. Erst dann kann man die Benutzereingaben für die Web Audio Api nutzen.

Zusätzlich erzeugen wir per PHP-Code unter Anderem auch HTML-Elemente wie die Select-Liste für das Geburtsjahr und sparen so einiges an unnötiger Tipparbeit.

```
<select id="birthYear_select" name="birthYear" class="custom-select" required="required">
  <?php
      for ($i=1; $i<=100; $i++){
          ?> <option value="<?php echo (2018-$i);?>"><?php echo (2018-$i);?></option> <?php
      }
  ?>
</select>
```

Abb. 5: HTML-Elemente per PHP erzeugen

Zwischenstand: Wir haben die Angaben des Benutzers von der ersten Seite mit dem Formular an die zweite Seite weitergeleitet und von dort an die JavaScript-Datei geschickt.

Innerhalb der JavaScript-Datei befindet sich unser gesamter Code für die Web Audio API, sowie einige Zeilen an Code, welche die interaktiven Elemente des Front-Ends gestalten.

Zu Beginn erstellen wir alle nötigen Variablen, welche wir für die Web Audio API verwenden.

```
var context = new (window.AudioContext || window.webkitAudioContext)();
var analyser = context.createAnalyser();
    analyser.minDecibels = -90;
    analyser.maxDecibels = -10;
    analyser.smoothingTimeConstant = 0.85;
var gainNode = context.createGain();
var convolver = context.createConvolver();
var distortion = context.createWaveShaper();
    distortion.curve = makeDistortionCurve(0);
    distortion.oversample = "4x";
var compressor = context.createDynamicsCompressor();
var filter = context.createBiquadFilter();
var audioBuffers = [];
var isPlaying = false;
var soundSource;
var source;
var stream;
var tempo;
```

Abb. 6: Web Audio API Variablen anlegen

Wir benutzen die Funktion “getAudioData” innerhalb einer IF-Schleife um eine Reihe von “XMLHttpRequests” an den Server zu senden und die gefundenen Audiodateien dann in einen Buffer zu speichern. Die Audiodateien haben wir dafür per FTP auf dem Strato-Server abgelegt. Hierbei ist die genaue Pfad-Bezeichnung entscheidend wichtig. Da man sonst nur einen 404-Fehler vom Request zurückbekommt.

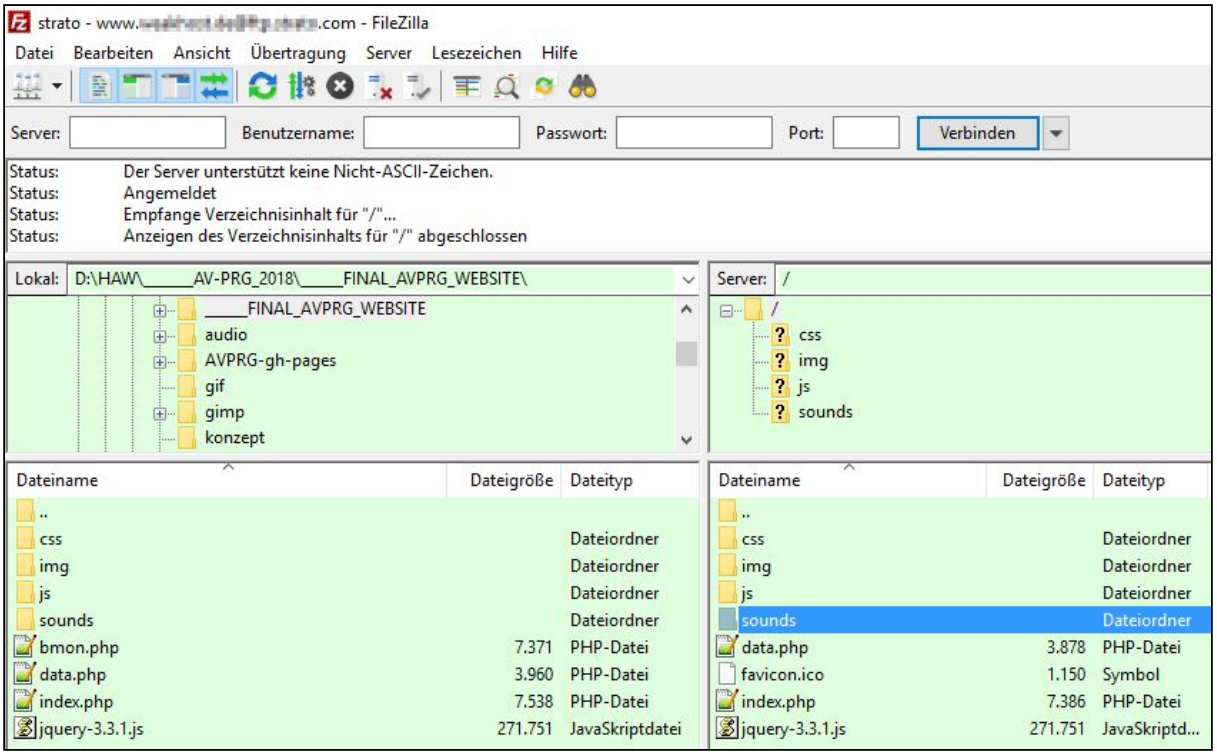

Abb. 7: FTP Zugriff per FileZilla auf den Server

Die Audiodateien im Buffer sortieren wir in zwei Arrays: Klavier und Beat. Nun können wir mit der Funktion “playBackgroundSound” auf diese zugreifen. Anhand des Geburtsjahres, welches der Benutzer angegeben hat, wählen wir ein Array mit einer Melodie in Form von Zahlen aus. Jede Zahl steht für eine Note aus dem Klavier, bzw. dem Beat Array.
Um nicht 31 verschiedene Melodien erstellen zu müssen erhalten die Geburtstage 17 bis 31 die Arrays der Tage 2 bis 16. Allerdings rückwärts durch die Arraysfunktion “array.reverse()”.

Über die Benutzerangabe “Geburtsjahr” variieren wir das Tempo des am Ende abgespielten Soundtracks. Hierzu haben wir 8 verschiedene Altersbereiche definiert, welche wieder rum jeweils ein anderes Tempo setzten. Jüngere Benutzer erhalten einen schnelleren Sound als ältere.

Für die Visualisierung bestimmen wir mithilfe von JQuery die HTML-Elemente, welche als Bereich der Darstellung genutzt werden sollen. Legen den Inhalt auf “2D” fest und definieren die Dimension der Fläche.

```
var canvas = document.querySelector('.visualisierungsBox');
var canvasCtx = canvas.getContext("2d");
var intendedWidth = document.querySelector('.mainArea').clientWidth;
    canvas.setAttribute('width',intendedWidth);
var drawVisual;
```

Abb. 8: Definieren des Visualisierungsbereiches

Erst jetzt können wir innerhalb der Funktion “visualize” die eben festgelegten Canvas-Bereiche verwenden. Innerhalb der Funktion legen wir die Farbe der Balken je nach gewähltem Geschlecht des Nutzers fest. Über eine simple If-else-Abfrage wird je nach Geschlechts-Wert ein anderer RGB-Farbwert für die Balken verwendet.

Die JavaScript-Datei regelt auch das Starten des Soundtracks per Schaltflächenklick, sowie das Erscheinen und verschwinden der Multiple-Choice Fragen.

```
document.getElementById("playMySong").addEventListener("click", function (e) {
        playBackgroundSound();
        visualize();
        $("div.bonus1").delay(1200).fadeIn(600);
        $("div.bedienelemente").fadeOut(300);
 });

// Code for show/hide Bonus Answers after Music is generated:
function end_of_Bonus1(){
  $("div.bonus1").fadeOut(400);
  $("div.bonus2").delay(900).fadeIn(500);
}
```

Abb. 9: Interaktive HTML-Elemente per JavaScript und JQuery

Wie man in Abbildung 9 sieht verwenden wir JQuery um die HTML-Elemente verschwinden oder erscheinen zu lassen. Der Play-Button startet die Funktionen für das Abspielen des Sounds, startet die Visualisierung der Balken und lässt sich selbst verschwinden. Nach einem kurzen Delay lässt er dann auch noch die erste Zusatzfrage anzeigen.

Innerhalb der Zusatzfrage ändern wir dann je nach gewählter Antwort die Filterwerte “frequency” und “distortion”. Auch die Angabe über den Geburtsmonat spielt an dieser Stelle eine kleine Rolle. Zusätzlich werden die HTML-Elemente wieder nach dem Klicken auf eine Antwort entfernt und die nächste Frage eingeblendet.

Zum Schluss ertönt dann der ganz persönliche Sound des Benutzers und dieser hat über eine eingeblendete Schaltfläche die Chance die Applikation von Vorne starten zu lassen.

```
<span id="end_text">Danke für deine Teilnahme.</span><br>
<span>Was du hörst, ist DEIN persönlicher Sound!</span><br>

<form action="https://www.weakhost.de/">
    <input type="submit" class="back2Start" id="back2Start" value="Zurück zum Start!" />
</form>
```

Abb. 10: Reset-Schaltfläche wird am Ende eingeblendet