HOCHSCHULE FÜR
TECHNIK UND WIRTSCHAFT
DRESDEN
UNIVERSITY OF APPLIED SCIENCES

LERNPORTFOLIO - IT1

HTML, CSS, Javascript

Marc Siggelkow

S82088
49784

Marc.siggelkow@htw-dresden.de

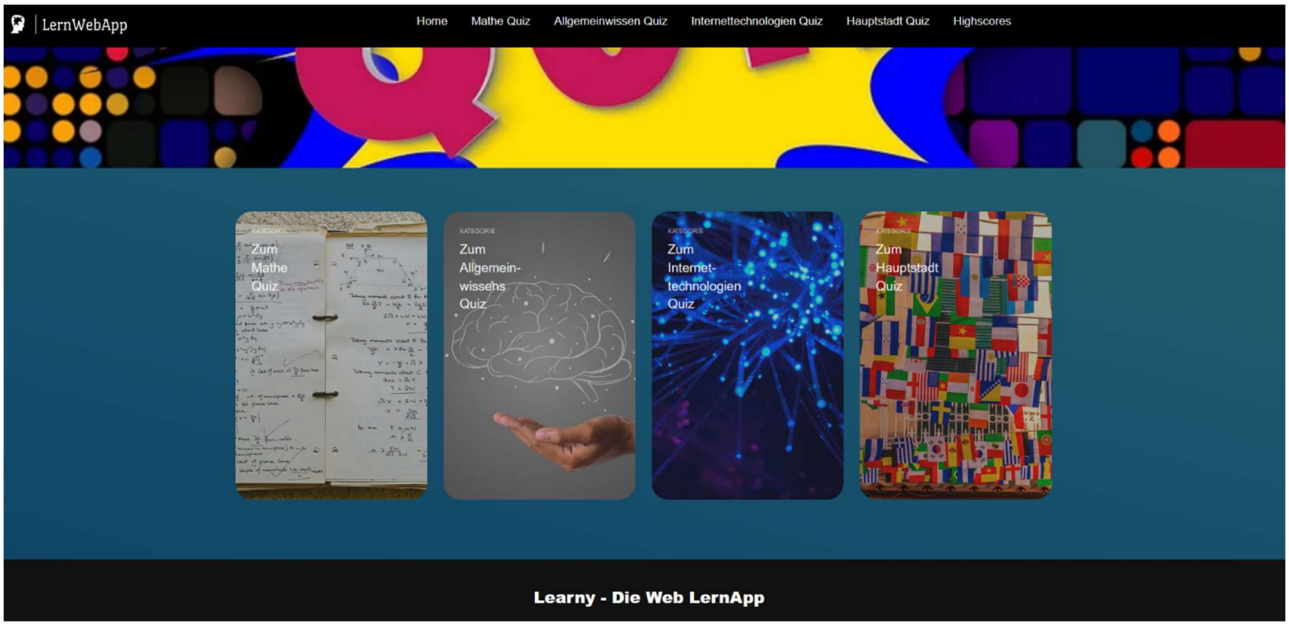
Das Programm

AUSGANGSLAGE

Als Beispiel für eine Progressive Web App (PWA) soll ein webbasiertes Programm zur Überprüfung allgemeiner bzw. Mathematikkenntnisse erstellt werden. Der Beleg dient zur praktischen Anwendung der Kenntnisse zu HTML, CSS und Javascript. Die Umsetzung als PWA ermöglicht auch die einfache und komfortable Nutzung in mobilen Geräten.Ausgangslage An Sachsen Gymnasien und Oberschulen

FUNKTIONSWEISE

Das ist die Startseite die der User beim Aufruf sieht.



Über die einzelnen Karten, welche durch

```
.card-grid{
  display: grid;
  grid-template-columns: repeat(1, 1fr);
  grid-column-gap: 24px;
  grid-row-gap: 24px;
  max-width: 1200px;
  width: 100%;
}
```

in ein Grid System eingeteilt werden, kann der User sich eine Quiz Kategorie raussuchen. Hier der Responsive first Ansatz sodass Standard mäßig 1 Grid eingeteilt ist (bei der Mobilen Ansicht würden die Karten sich also übereinander anordnen).

Mit einem @media werden dann die anderen Ansichten bedient: 2 Grid für Tablet/Laptop und 4 Grids für Desktop Ansicht.

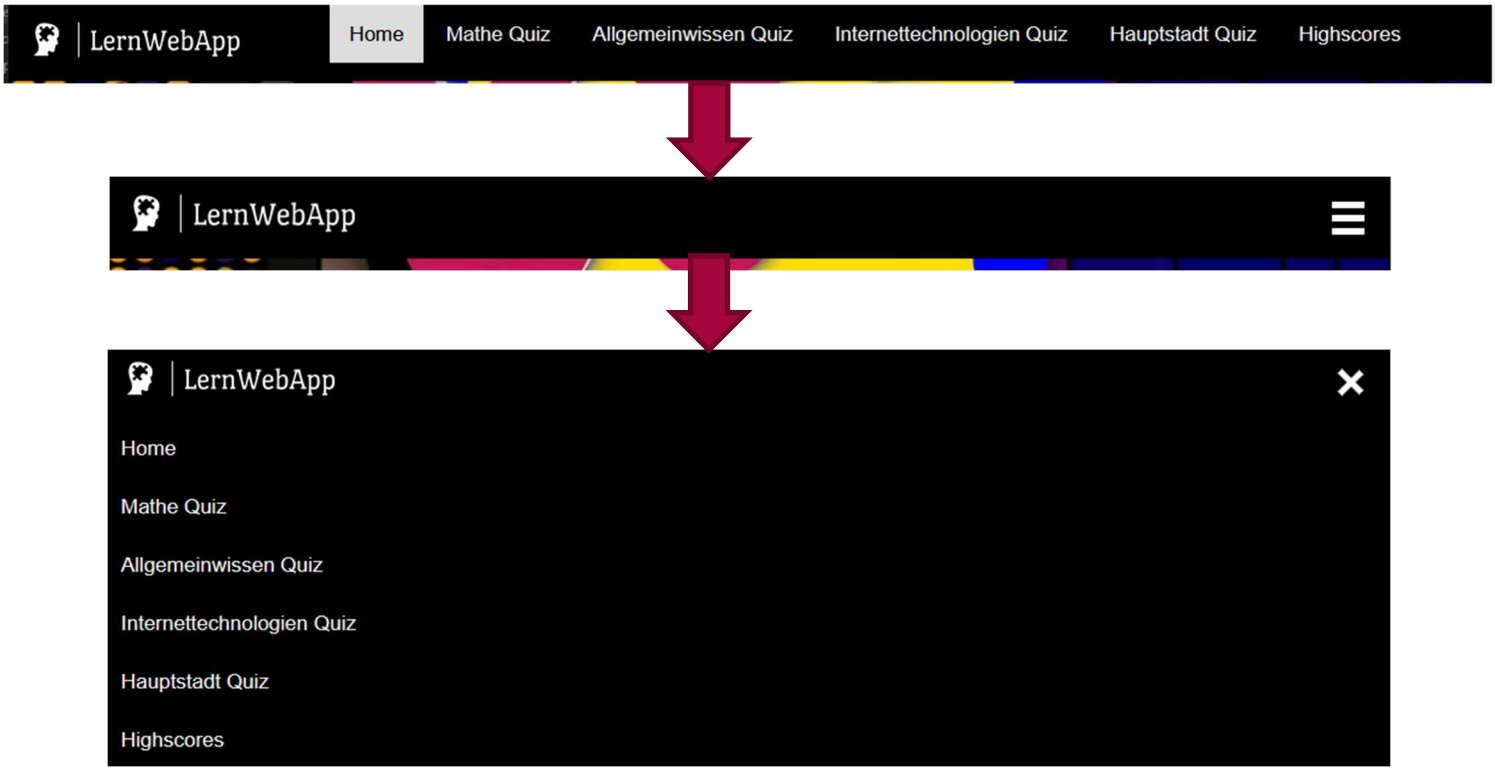
```
/*
 * Responsive: Für Bildschirm Größe kleiner als 540px
 * Nur 2 Coloms für Card
 */

@media(min-width: 540px){
  .card-grid{
    grid-template-columns: repeat(2, 1fr);
  }
}

/*
 * Responsive: Für Bildschirm Größe kleiner als 960px
 * Nur 4 Coloms für Card
 */

@media(min-width: 960px){
  .card-grid{
    grid-template-columns: repeat(4, 1fr);
  }
}
```

Der Header ist ebenfalls Responsive gestaltet welcher ab einer Screen Width von kleiner als 1130px zu dem Hamburger Menü wechselt.



Wählt der User eine Kategorie über die Navbar oder die Kategorie-Cards aus, wird er auf die „game.html“ weitergeleitet. Bei der Auswahl habe ich über javascript der Url eine ID mitgegeben um auszuwerten, welches Quiz der User spielen möchte.

```
<a id="mathe" class="card"  onclick="getIdFromUrl(this.id);">
```

```
  //Function add ID to url from what User picked for category
function getIdFromUrl(obj)
{
    //fixes problem that href works on WebServer (where first Path is /~s82088/)
    //but not on localHost anymore
    const firstPath = location.pathname.split('/')[1];
    if(obj === "home") {
        location.href = window.location.origin+"/"+firstPath+"/index.html";
    } else if (obj === "highscores") {
        location.href = window.location.origin +"/"+firstPath +"/highscores.html";
    } else {
        location.href ="";
        location.href = window.location.origin +"/"+firstPath +"/game.html?id=" + obj;
    }
}
```

Ergebnis:

https://www2.htw-dresden.de/~s82088/game.html?id=mathe

Das Quiz habe ich komplett über die WebquizAPI realisiert. Es werden sich also keine Fragen lokal gezogen und auch nicht lokal ausgewertet, dies geschieht alles über die API.
Am Anfang setze ich also eine Variable auf die jeweiligen ID’s die meine Fragen zu der jeweiligen Kategorie in der API enthalten.

```
19  //Gets the ID from the URL based on which category the User picked
20  var baseUrl = (window.location).href;
21  var koopId = baseUrl.substring(baseUrl.lastIndexOf('=') + 1);
22  //Sets the range of ID's for the Questions we will fetch later from  the  API
23  // based on which category the user picked
24  switch (koopId) {
25      case "mathe":
26          categorySwitch = [469,478];
27          break;
28      case "allgemein":
29          categorySwitch = [506,517];
30          break;
31      case "internet":
32          categorySwitch = [524,533];
33          break;
34      case "hauptstadt":
35          categorySwitch = [1563,1572];
36          break;
37      default:
38          break;
39    }
```

Dann ziehe ich mir die beiden ID's aus dem Array und hole mir alle Fragen von der API.
Der Ablauf dabei ist wie folgt:

1. Mittels einer for-loop iteriere ich über die Länge meiner Fragen
2.

```
75 for(let i = firstElement; i<= lastElement;i++) {
76     fetchAsync(i)
77         .then(data => formatQuestion(data))
78         .catch(reason => console.log(reason.message))
79 }
```

Als Beispiel mit der Kategorie Mathe:
For(i = 469; i <= 478; i++)
Dann wird die funktion fetchAsync aufgerufen und die aktuelle ID mit gegeben.
Diese Funktion ruft mittels fetch() die API auf, übergibt die aktuelle ID der Frage und als Response erhalten wir ein Promise.
Diesen schreiben wir in data und geben diesen zurück.

```
59     // only proceed once promise is resolved
60     let data = await response.json();
61     // only proceed once second promise is resolved
62     return data;
63   }
```

Da der Response nur innerhalb des Scopes in dem wir uns befinden verfügbar ist und nicht global, werden ab jetzt nur noch weiter Funktionen aufgerufen.

Der return Wert wird mittels .then in Zeile 77 an die funktion formatQuestion() übergeben, in der wir, sobald alle 10 Fragen geholt wurden, die Daten noch formatieren um schlussendlich korrekt anzuzeigen und die funktion startGame() aufgerufen.

```
82  function formatQuestion(object) {
83      //pushing all fetched Objects into result
84      result.push(object);
85      //when we have all 10 Questions -> we set them up to use them
86      if(result.length === 10) {
87          // iterating over all objects in "result"
88          questions = result.map((loadedQuestion) => {
89              const formattedQuestion = {
90                  id: loadedQuestion.id,
91                  question: loadedQuestion.text,
92              };
93
94              const answerChoices = [...loadedQuestion.options];
95              //setting ID which we will use later for calling API to solve quiz
96              // +1 because array is [0,1,2,3] but API id needs to be [1,2,3,4]
97              answerChoices.forEach((choice, index) => {
98                  formattedQuestion['choice' + (index + 1)] = choice;
99              });
100
101             return formattedQuestion;
102     })
103     startGame();
104     }
```

Dort wird dann zufällig eine der verfügbaren Frageben ausgewählt und mittels folgenden Befehlen dargestellt.

```
//Update HTML with current question
question.innerHTML =  currentQuestion.question;
//Updating HTML with answer options
choices.forEach((choice) => {
    const number = choice.dataset['number'];
    choice.innerHTML = currentQuestion['choice' + number];
});
```

Dann wird für jeden „choice“ welche ein Auswahlkästchen für die Antwort in HTML darstellt, ein EventListener eingebunden der die Richtigkeit der ausgewählten Antwort überprüft.
Dies geschieht wieder mittels fetch() und der WebQuizAPI.

Bei Falscher Antwort wird das Kästchen rot angezeigt, bei korrekter Antwort grün und Punkte gutgeschrieben und eine neue Frage aus den noch übrigen geholt.
Sind alle Fragen beantwortet, speichern wir das aktuelle Quiz und die Punktzahl im localStorage des Browsers und leiten den User zur End Seite weiter.

```
123     if (availableQuesions.length === 0 || questionCounter >= MAX_QUESTIONS) {
124         localStorage.setItem('mostRecentScore', score);
125         localStorage.setItem('mostRecentQuiz', koopId);
126         //go to the end page
127         const firstPath = location.pathname.split('/')[1];
128         let endScreen = window.location.origin+"/"+firstPath+"/end.html?id="+koopId;
129         return window.location.assign(endScreen);
130     }
```

Dort angekommen, kann der User seinen Namen eingeben und die Punktzahl und Quiz wird im Storage gespeichert und kann später verglichen werden.

```
17  saveHighScore = (e) => {
18      e.preventDefault();
19
20      const score = {
21          quiz: mostRecentQuiz,
22          score: mostRecentScore,
23          name: username.value,
24      };
25      highScores.push(score);
26      highScores.sort((a, b) => b.score - a.score);
27      highScores.splice(5);
28
29      localStorage.setItem('highScores', JSON.stringify(highScores));
30      const firstPath = location.pathname.split('/')[1];
31      let endScreen = window.location.origin+"/"+firstPath+"/highscores.html";
32      window.location.assign(endScreen);
33  };
34
```

REFLEKTION

Grundkenntnisse in HTML und CSS waren bei mir schon vor der Belegaufgabe vorhanden sodass dieser Teil für mich leichter viel. Neu bzw. anspruchsvoller war für mich der Javascript Teil, da ich dort noch nicht so viele Erfahrungen habe sammeln können.
Daher fand ich die Belegaufgabe sehr interessant und vom Umfang auch angemessen. Ich konnte vor allem in Javascript viele neue Kenntnisse für mich gewinnen. Besonders gut fand ich auch die Forderungen zur Nutzung einer API. Diese hat mich zwar wahrscheinlich die meisten Nerven gekostet, war aber auch mein größtes Highlight in dem Beleg, als nach hunderten Anfragen und komischen Promisses in meinen Arrays, endlich die korrekte Frage dargestellt wurde.