TS-Aufgabe 4: "Es gibt keine if-Schleifen"
Spieler-Level und Monster-Stufe sind nun wichtig!

Konzept:

Dieses mal mit Konzept!
Konzipiert die neuen (und auch veränderten) Funktionen! Schreibt euch zu jeder Funktion auf, welche der kleineren Komponenten (Schleifen, Konditionen, Wert-Zuweisungen, etc etc) ihr Nutzen könntet um diese Umzusetzen. Es muss kein richtige funktionabler Code dafür geschrieben werden, lediglich die einzelnen wichtigen Code-Elemente sollten dabei sichtbar sein.

Konzept bis zum **20.06.**2019.

VARIABLEN:
[ ] Gebt dem Spieler eine Level-Variable. Diese soll sich mit steigender `playerXP` erhöhen. Siehe auch die Rechnung in der `updatePlayerLevel()`-Funktion, bei welcher momentan eine lokale Variable verändert wird.
[ ] Gebt dem Monster eine Level-Variable im Interface, falls noch nicht vorhanden. Diese kann einen zufälligen ganzzahligen Wert zwischen 0-10 annehmen.

HTML/CSS
[ ] Fügt drei weitere Buttons hinzu. Die Beschriftung ergibt sich aus den neuen Funktionen. Platziert diese neuen Buttons neben den alten Monster-Finden-Button.

FUNKTIONEN:
Neue Funktion: **"fightAllMonsters"**
[ ] Fügt eine neue Funktion zu eurem Programm hinzu. Durch einen Button-Klick wird ein Kampf mit allen vorhandenen Monstern ausgeführt (siehe vorhandene `fightMonster`-Funktion), egal welchen Levels.

Zu beachten: `fightMonster` entfernt das Monster aus dem Monster-Array! Da in `fightAllMonsters` vermutlich eine Schleife benutzt wird, muss hier besonders auf den Index des Schleifendurchlaufes geachtet werden! Wenn ein Monster besiegt wird ändert sich damit auch die Array-Länge vom Monster-Array. Unter Umständen kann dies dazu führen, dass eure For-Schleife nicht jedes Monster behandelt, da entfernte Monster zu einer Verrückung des Arrays führen.
Für eine eine einfache, saubere Lösung wäre folgendes eine Option:

```
for(let i = monsterArray.length - 1; i >= 0; i--) { // Geht ein Array von hinten nach vorne durch, umgeht dadurch das Problem der Array-Verrückung.
        // Euer Code
}
```

Neue Funktion: **"fightAllWeakMonsters"**
[ ] Fügt eine neue Funktion zu eurem Programm hinzu. Durch einen Button-Klick wird ein Kampf mit allen vorhandenen Monstern ausgeführt, welche schwächer ("kleineres Level") sind als der Spieler.
Benutzt dafür eine if-Bedingung sowie eine Schleife eurer Wahl.

Neue Funktion: **"fightWeakestMonster"**
[ ] Fügt eine neue Funktion zu eurem Programm hinzu. Diese sucht nach dem Index des schwächsten vorhandenen Monsters. Dieses eine wird danach bekämpft, falls es eine geringere Stufe als der Spieler besitzt.

Zu beachten: In der Vorlesung hatten wir einen ähnlichen Code.

Veränderte Funktion: **"fightMonster"**
Beschreibung: Anstelle von dem direkten Besiegen des Monsters wird nun ein metaphorischer Kampf durchgeführt. Wenn der Spieler auf ein Monster klickt um es zu bekämpfen wird nun auch geschaut ob der Spieler das Monster besiegen kann.
[ ] Dabei wird Monster- und Spieler-Level verglichen. Sollte das Spieler-Level geringer sein, so verliert der Spieler. Das Monster wird dabei NICHT entfernt!
[ ] Ansonsten gewinnt der Spieler. Falls der Spieler gewinnt wird das Monster wie gewohnt einfach entfernt.
[ ] Je nach Fall wird anderer Programmcode aufgerufen. Im Falle von Gewinnen wird updatePlayerLevel mit positiver Monster-XP aufgerufen, ansonsten wird updatePlayerLevel mit negativer Monster-XP aufgerufen.
[ ] Ebenfalls soll dabei die veränderte updatePlayerLevel-Funktion aufgerufen werden. Diese nimmt nun einen Parameter entsprechend der Level-Veränderung entgegen. Wird bei Gewinnen/Verlieren des Kampfes mit entsprechend positiven/negativen `monsterExperience` des Monsters aufgerufen.

Veränderte Funktion: "**updatePlayerLevel**"
Soll nun generelle Spieler-Level-Veränderung verarbeiten und nicht nur das HTML-Element dafür updaten. Verändert die neue Variable Spieler-Level.
[ ] Besitzt nun einen Parameter, welcher entweder die Level-Veränderung oder Erfahrungspunkte-Veränderung angibt. (Eure Wahl)
[ ] Die Funktion kann sowohl mit positivem als auch negativen Zahlen aufgerufen werden.
[ ] Spieler kann beim Verlieren eines Kampfes durch den Erfahrungspunkte-Verlust (oder Level-Verlust) NICHT unter Stufe 1 sinken. Sollte das Spielerlevel also auf unter 1 gehen, wird der Wert stattdessen auf 1 gesetzt.
[ ] Sollte der Spieler nach einem Kampf seine Stufe auf GENAU 20 bringen, so bekommt er eine einmaligen Alert, dass er gewonnen hat.
[ ] Andere Lösungen sind auch IO, solange sie mehrere if und/oder else verwenden.

Optionale Ziele (für Stern):
[ ] Gebt den Monstern Lebenspunkte. Bei einem Kampf wird das Monster nun nicht direkt besiegt, sondern es verliert für jeden Klick Lebenspunkte. Es kann nur Lebenspunkte verlieren solange es ein geringeres Level besitzt.
[ ] Gebt den Monster nicht zu viel Lebenspunkte. Es sollte bei angemessener Klickgeschwindigkeit besiegt werden können.
[ ] Stellt die Lebenspunkte in einem HTML/CSS-gestylten Balken dar, welcher mit sinkendem Leben ebenfalls kleiner wird.

Erwartete Funktionalität der Abgabe:
Wenn alles umgesetzt wurde sollte der Spieler nun in der Lage sein manche

Monster nicht direkt bekämpfen zu können, da diese zu stark sind.

Per Buttons können entweder alle Monster angegriffen werden (und man kann dabei auch verlieren), oder per zweiten Button kann man alle schwachen Monster besiegen oder man bekämpft per letztem Button das schwächste vorhandene Monster.

Man kann nun das Spiel gewinnen (und erhält entsprechend eine Meldung) wenn man Stufe 20 erreicht.

Abgabe: Bis zum **23.06.**