Mousebreed Dokumentation
Play.js

October 15, 2015

1 verwendete Frameworks

1.1 HTML5

1.1.1 Bootstrap

Die Oberflaeche der Website wird durch das freie CSS-Framework Bootstrap dargestellt. Viele der sichtbaren Komponenten bestehen as diesem Framework. Zum Darstellen der Website nutzen wir das “Grid System” um unseren Inhalt aufzuteilen. Dabei erhaelt die Seitenleiste “col-md-1” und der jeweilige Inhalt der Seite “col-md-11”.

```
1     <div class="col-sm-2 col-md-1 sidebar">
2     </div>
3     <div class="col-sm-10 col-sm-offset-2 col-md-11 col-
          md-offset-1 main">
4         <?php include_once "$page.php"; ?>
5     </div>
```

Listing 1: Grundstruktur der Website

1.1.2 Canvas

Das Canvas ist ein HTML5-Element, welches die Zeichnung von Grafiken mittels Javascript ermöglicht. Es definiert einen Bereich durch Höhen- und Breitenangaben, in welchem vorhandene Grafiken positioniert oder neue Zeichnungen angelegt werden können.

```
1 <canvas id="myCanvas" width="800" height="600">
2 </canvas>
```

Listing 2: Einbinden des HTML5-Canvas-Elementes

1.1.3 LocalStorage

Als Weiterentwicklung von Cookies ermöglicht der LocalStorage eine, mit dem Benutzerprofil des lokalen Rechners verbundene, Speicherung von Daten für eine Webanwendung. Diese Daten lassen sich durch die Anwendung auslesen und bearbeiten, sodass sich erreichte Fortschritte oder gewünschte Einstellungen nachhaltig festhalten lassen.

```
1 localStorage.setItem('key', 'value');
2 localStorage.getItem('key'); //output -> value
```

Listing 3: Beispielcode für das Setzen und Lesen des LS

1.2 CreateJS

CreateJS ist ein Kollektiv von Bibliotheken um interaktiven Inhalt in HTML5 zu erstellen (Table 1). Im Rahmen des Projektes wurde ausschließlich die EaselJS Bibliothek verwendet, um eine solide Nutzung des HTML5 Canvas zur Darstellung von Grafiken zu gewährleisten.

EaselJS :	Bibliothek zur Verwendung von Grafiken in HTML5 Canvas
TweenJS :	Bibliothek zur Animation von Grafiken mittels Javascript
SoundJS :	Bibliothek zur Verwendung von Audioelementen
PreloadJS :	Bibliothek zur Verwaltung von Anwendungsdateien

Table 1: CreateJS Bibliotheken

1.2.1 Benachrichtigungen

Die Benachrichtigungen bauen auf dem JQuery-Plugin “Notify.js” auf. Die Benachrichtigungen sind in vier Klassenunterteilt:

- Erfolg
- Information
- Warnung
- Fehler

Gleichzeitig werden die Benachrichtigungen auch in der Benachrichtgungsleiste angezeigt. Zusaetzlich wird dort auch die Uhrzeit des Auftretens gespeichert.

1.3 PHP

Um die Webanwendung dynamisch gestalten zu können, haben wir uns der serverseitig interpretierten Skriptsprache PHP bedient. PHP-Code wird auf dem Server ausgeführt. Anschließend wird das Ergebnis an den Client (Browser) gesendet, der das angeforderte und zusammengestellte Dokument anzeigt. Die Folgenden Funktionen wurden mit PHP umgesetzt:

- Datenbankanbindung
- Benutzerverwaltung
- Verarbeitung von Eingabeformularen
- Informationen in Templates einfügen

1.3.1 Datenbankanbindung

Der Zugriff auf die Datenbank erfolgt mit PHP Data Objects (PDO). PDO bietet einen einheitlichen Zugang von PHP auf verschiedene SQL-Datenbanken. Der Portierungsaufwand beim Wechseln auf eine neues Datenbanksystem ist somit sehr gering.

```
1 <?php
2
3 $db = new PDO('mysql:host=localhost;dbname=mousebreed;
      charset=UTF8', 'mousebreed', 'genotyp');
4
5 ?>
```

Listing 4: Inhalt der Datei dbConnection.php

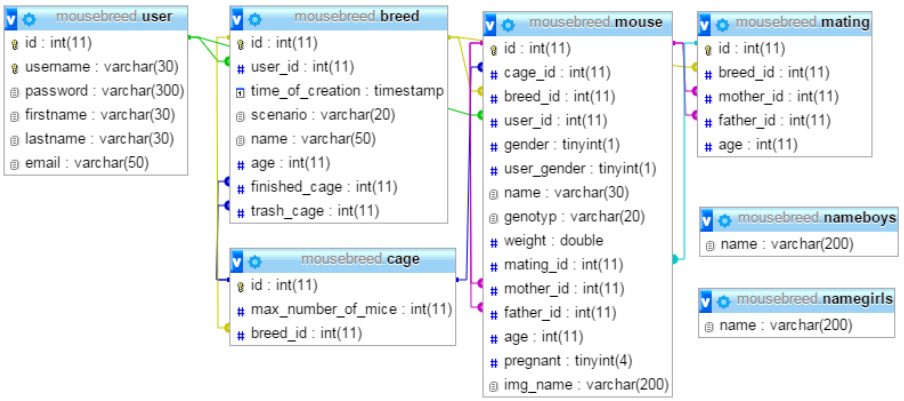

Figure 1: Struktur der Datenbank

Funktionen um auf die Datenbank zu zugreifen stellen die Dateien *userModel.php* und *breedModel.php* zur Verfügung. *userModel.php* beinhaltet Funktionen die den Benutzer betreffen, analog beinhaltet *userModel.php* Funktionen die die Zucht betreffen. Zum Beispiel liefert die Funktion *getMouse($mouseId)* aus der Datei *breedModel.php* alle Daten von der abgefragten Maus.

```
1   public function getMouse($mouseId) {
2         $stmt =  "SELECT * ".
3                  "FROM `mouse` ".
4                  "WHERE id = ?";
5         $stmt = $this->db->prepare($stmt);
6         $stmt->bindParam(1, $mouseId);
7
8         if($stmt->execute() && $mouse = $stmt->fetchAll(\
              PDO::FETCH_ASSOC)) {
9             return $mouse;
10        } else {
11            return Array();
12        }
13    }
```

Listing 5: Aufbau der Funktion getMouse

1.3.2 Informationen in Templates einfügen

Je nachdem welche Seite der Anwendung aufgerufen wird, werden unterschiedliche Informationen von der Datenbank abgefragt und mit PHP-Code in das HTML-Template eingefügt. Ruft der Benutzer sein Profil auf, werden zum Beispiel alle Daten von seinen Zuchten abgefragt und angezeigt.

```
1 <?php foreach ($_SESSION['breeds'] as $breed) { ?>
2         <tr>
3         <td><?php echo $breed['name']; ?></td>
4         <td><?php echo $breed['time_of_creation']; ?></td
              >
5         <td><?php echo $breed['scenario']; ?></td>
6         <td><?php echo $breed['numberOfMice']; ?></td>
7         <td><?php echo $breed['numberOfCages']; ?></td>
8         </tr>
9 <?php } ?>
```

Listing 6: Auflistung der Zuchtdaten

2 Arbeitsweise

Die *Play*-Seite ist dafür zuständig, den eigentlichen Spielvorgang ermöglichen. In ihr sollen die Käfige mit ihren Mäusen visualisiert werden und dem Nutzer die Möglichkeit zur Spielbeeinflussung eingeräumt werden. Dazu wurden in der Seite ein Steckbrief mit Informationen über ausgewählte Mäuse und ein Canvas als Käfigfläche eingebunden. Zusätzlich sind Buttons vorhanden, um einen Tag im Spielgeschehen verstreichen zu lassen oder männliche Mäuse aussortieren zu können. Dieses wird mit der Engine ermöglicht. Bei Abschluss des Ladevorgangs der *Play.js* wird das Canvas mit Hilfe seiner festgelegten ID als gewünschte Zeichenfläche definiert. Es kann auf Mausevents innerhalb der Fläche reagiert werden und es lassen sich Objekte in Diese einbinden. Zusätzlich wird ein *Ticker* initialisiert, welcher die Zeichenfläche in regelmäßigem Abstand neu zeichnet und somit Veränderungen sichtbar macht(siehe Listing 3).

```
1  function init(){
2  //Canvas
3  stage = new createjs.Stage("myCanvas");
4  stage.enableMouseOver();
5  stage.mouseChildren = true;
6  ...
7  //Ticker
8  createjs.Ticker.on("tick", tick);
9  createjs.Ticker.timingMode = createjs.Ticker.RAF;
10 createjs.Ticker.setFPS(60);}
```

Listing 7: init() - Funktion der Play.js

Die zu zeichnenden Objekte werden dem *LocalStorage* entnommen, welcher beim Laden des Spiels mit allen bekannten Informationen des Speicherstandes gefüllt worden ist. Diesem Speicher werden die Einzelheiten über vorhandenen Käfige, sowie die darin befindlichen Mäuse entnommen und in angemessene Datenstrukturen umgeformt. Diese Strukturen enthalten die notwendigen Kerninformationen, sowie eine visuelle Repräsentation in Form einer Bitmap und werden in Arrays festgehalten.
Im Rahmen einer Zeichenfunktion *draw()* werden alle Elemente dieser Arrays an einer bestimmten Positionen der Zeichenfläche hinzugefügt und je nach derzeitig ausgewähltem Käfig sichtbar gemacht. Sichtbare Elemente werden nun innerhalb des ausgewiesenen "Käfigbereiches" animiert.
Das Animieren übernimmt der initial definierte "Ticker". Dieser überprüft jede Maus des Mausarrays und bewegt diese entsprechend ihrer vordefinierten Bewegungsrichtung. Ob eine Maus derzeit zu bewegen ist, lässt sich mittels einer Variable innerhalb der Mausstruktur bestimmen(Listing 8).
Der Nutzer kann nun diese animierten Objekte mit dem Cursor auswählen, um verschiedene Aktionen auszuführen:

- Mausobjekte lassen sich selektieren, wobei ihre Informationen an das Steckbrief-Element weitergegeben und in Textform sichtbar gemacht werden. Anschließend kann dort die Geschlechtsbestimmung stattfinden.

```
1  //Mausobjekt
2  function Maus(){
3  ...
4  this.mousecontainer.ismove = true;
5  this.mousecontainer.isdrag = false;}
6  //Update-Funktion
7  function tick(){
8  for (i = 0; i < arrMouse.length; i++){
9  var elem = arrMouse[i].mousecontainer;
10 if (!elem.isdrag && elem.ismove){
11 elem.move();}}
12 //Aktualisieren des Canvas
13 stage.update();}
```

Listing 8: Mausstruktur und Ticker-Update

- Mausobjekte lassen sich per Drag and Drop positionieren. Werden sie auf einen Käfig mit freien Kapazitäten gezogen, so wird die Maus in diesen umgesiedelt.
- Wird ein Käfig selektiert, so wird die Ansicht auf dessen Inhalt umgeschaltet.

Im wesentlichen wird eine durch den Nutzer vollführte Änderung innerhalb der *Play*-Seite auf die Datenstruktur abgebildet und anschließend mit Hilfe der Enginefür die Spiellogik umgesetzt.

3 loadedBreed

Die Variable loadedBreed dient dazu die Informationen, der geladenen Zucht, aus der Datenbank bereitzustellen und gemäß dem Spielverlauf zu modifizieren. Die Initialisierung und Aktualisierung erfolgt über den local storage. Die Aufgabe war es die Daten persistent zu speichern, jedoch gleichzeitig zu hohe Netzwerkauslastung zwischen Nutzer und Server zu vermeiden. Hierfür benutzen wir loadedBreed, welche zunächst alle Änderungen durchführt und auf dessen Grundlage das Spiel abläuft. Einige Funktionen und letztlich das Beenden des Spiels führen zu einer Angleichung der Datenbank mit der loadedBreed und sichern den Spielfortschritt dauerhaft.

4 target

Die Variable target beinhaltet die verschiedenen Szenarien in Form der jeweiligen Spielziele. Über sie wird geprüft ob der Spieler das Ende erreicht hat oder noch weiter züchten muss. Über die Funktion engine.setTarget() wird diese Variable initialisiert.

5 engine

Engine liefert grundlegende Funktionen für den Spielablauf.

- Zum erstellen von neuen Mäusen
  Dabei ist zu unterscheiden zwischen dem einfachen erschaffen(newMouse()) und dem geboren werden(birth()), wobei birth() unter den richtigen Umständen newMouse() aufruft. Außerdem gibt es newMating() zum Erzeugen eine neuen Paarung

- Zum arbeiten mit existierenden Mäusen
  Hier wird das Alter erhöht und zwei Funktionen zum Finden bestimmter Mäuse. Die Generation kann ermittelt werden und die erste Maus eines Käfigs wird ermittelt

- Zum arbeiten mit Käfigen
  Es kann ein neuer Käfig erzeugt werden. Mäuse können zwischen den Käfigen verschoben werden/ aussortiert werden. Es können bestimmte Käfige gefunden werden. Käfige können gelöscht werden

- Speichern und Fortfahren prüfen
  Der Spielfortschritt wird gespeichert und es wird geprüft ob alle Bedingungen erfüllt sind um zum nächsten Tag zu wechseln. Update loadedBreed aus dem localStorage

- Game Over
  Prüfen ob das Spiel zu ende ist. Hierfür werden die benötigten Hilfsfunktion zur Verfügung gestellt

6 clock

Clock liefert Funktionen, welche für jeden Tag und im speziellen für die nextDay() Funktion, benötigt werden.

- Nächster Tag
  Die Hauptfunktion nextDay() und dazugehörige Hilfsfunktionen führen alle Aktionen aus welche ein Tageswechsel benötigt und prüft Bedingungen.

- Täglich Ende prüfen
  Die Hauptfunktion checkTarget() prüft, mit Hilfe der Funktionen aus dem GameOver-Bereich der engine, ob das Spiel zu Ende ist.