Übung 06

1 Vue.js

Es ist wieder soweit, der Spieleentwicklermarathon *GmTk Game Jam* steht an, das dies jährige Thema ist Kilcker-Games. Um sich auf den nur ein Wochenende langen Marathon vorzubereiten, möchten Sie jetzt schon prototypisch ein Klicker-Spiel umsetzen. Hierbei soll das Spiel aus folgenden Kerninhalten bestehen:

- Es soll einen *Mine-Button* geben, der beim Klicken die Menge der zu erspielende Kryptowährung um den Wert 1 erhöht. Über dem Button soll der Gesamtwert der so erspielten Währung angezeigt werden.
- Neben dem Gesamtwert sollen 4 *Upgrades* in Buttonform zur Verfügung gestellt werden. Beim Klicken dieser Buttons soll der Wert der Generierten Kryptowährung pro Klick des *Mine-Button* erhöht werden.
- Weiter soll die Anzahl der pro Klick erhaltenen Kryptowährung unter dem Gesamtwert angezeigt werden.
- Zuletzt soll die gesamte Spielfläche in ein Spielfeld (links) und ein Upgradefeld (rechts) sortiert werden. Der Titel des Spiels soll sich Oberhalb beider Hälften befinden.

Da sich während des Spiels dauerhaft etwas ändert und auf diese Änderungen reaktiv eingegangen werden soll, entschließen Sie sich das Vue.js Framework zu nutzen.

Lösungsweg

Beginnen wir mit der Erstellung des HTML Dokuments. Da wir das Vue.js Framework nutzen wollen, müssen wir dieses auch in unser HTML-Dokument einbinden. Vue.js stellt uns hierfür

eine einfache Methode zu Verfügung, unzwar können wir das gesamte Vue-Framework über das script-Tag laden.

```
<head>
        <meta charset="utf-8">
        <title>Klicker Game</title>
        <link rel="stylesheet" href="main.css">
        <script
         ↪  src="https://cdn.jsdelivr.net/npm/vue@2.6.12/dist/vue.js"></script>
</head>
```

Da wir nun das Vue-Framework im Header bereits geladen haben, können wir mit der Erstellung der eigentlichen Webseite, dem Body, beginnen. Zuerst benötigen wir einen Container, in dem wir unsere Vue-Instanz benutzen können. Im Normalfall handelt es sich hierbei um ein ** **-Element, dass die ID *app* besitzt. Zudem erstellen wir Zeitgleich mehrere ** **-Elemente, um die Webseite über unsere *main.css* zu gestalten.

```
<body>
  <div id="app">
    <div class="wrapper">
      <h1 class ="title">Crypto Miner 2.0</h1>
      <div class="spielfeld"></div>
      <div class="upgrades"></div>
    </div>
  </div>
</body>
```

Als nächstes erstellen wir unsere Vue-Hauptinstanz in einer separaten JavaScript Datei. Hierbei setzen wir auch direkt das **el**-Attribute, damit unsere Vue-Instanz direkt auf der HTML-Ebene gemounted wird.

```
var vm = new Vue({
  el: '#app'
})
```

Schauen wir uns nun das eigentliche Spielfeld an. Aus der Aufgabenstellung wissen wir, das hier drei Dinge umgesetzt werden sollen. Eine Anzeige, wieviel Kryptowährung erspielt wurde, eine Anzeige, wie viel Kryptowährung pro Klick erhalten wird, und ein Button der die erspielte Kryptowährung erhöht. Die ersten Beiden Punkte können mit der Moustache Schreibweise von Vue umgesetzt werden. Hierfür nutzen wir **<p>**-Tags und schreiben in doppelte geschweifte Klammern den Variablennamen der aus der Vue-Instanz getrackt werden soll. Wir starten das Spiel mit einem Gesamtwert von 0 und einer Währungseinheit pro Klick.

HTML:

```
<p> {{cryptoMenge}} Crypto Coins</p>
<p>Aktuelle Crypto Coins per Klick {{cryptoModifikator}}</p>
```

JS:

```
var vm = new Vue({
  el: '#app',
  data: {
    cryptoMenge: 0,
    cryptoModifikator: 1
  }
})
```

Im nächsten Schritt erstellen wir den Button zur Erhöhung der erspielten Kryptowährung. Hierfür erstellen wir einen normalen HTML Button über das **button**-Tag und verlinken diesen mithilfe der **v-on** Vue-Direktive zu einer Methode innerhalb der Vue-Instanz.

HTML:

```
<button v-on:click="addCrypto">Mine Cryptocoin</button>
```

JS:

```
methods:{
    addCrypto: function(){
      this.cryptoMenge += 1 * this.cryptoModifikator;
    }
```

Zuletzt wollen wir noch das Upgradefeld mit 4 Upgrades versehen, die den Wert erspielter Kryptowährung pro Klick erhöht. Hierzu nutzen wir Analog wie im vorherigen Schritt **button**-Tags und verlinken diese ebenfalls mit einer Methode aus der Vue-Instanz, mit dem Unterschied, das wir diesmal auch einen Parameter an die Methode übergeben.

HTML:

```
<div class="upgrades">
      <button v-on:click="increaseModifier('botminer')">Bot Miner</button>
      <button v-on:click="increaseModifier('rtx3090')">RTX 3090</button>
      <button v-on:click="increaseModifier('botnetzwerk')"></button>
      <button v-on:click="increaseModifier('serverraum')">Serverraum</button>
</div>
```

JS:

```
increaseModifier: function(code){
  switch(code){
    case "botminer":
      this.cryptoModifikator += 1;
      return;

    case "rtx3090":
      this.cryptoModifikator += 5;
      return;

    case "botnetzwerk":
      this.cryptoModifikator += 10;
```

```
        return;

      case "serverraum":
        this.cryptoModifikator += 50;
        return;

      default:
        alert("Something went wrong");
        return;
    }
  }
```

2 Aufgaben

Bearbeiten Sie nun die folgenden Aufgaben.

Aufgabe 1

Erweitern Sie nun den zuvor erstellten Klicker um eine Kostenfunktion für die einzelnen Upgrades. Jedes Upgrade soll einen individuellen Preis in Crypto Coins kosten (z.B. Bot Miner für 10 Crypto Coins). Die Preise für die einzelnen Upgrades sollten auf der Webseite für den Spieler gut sichtbar sein. Sollte der Spieler nicht genügend Crypto Coins für das gewählt Upgrade besitzen, soll über einen Infotext unter den Buttons eine Nachricht angezeigt werden, die den Spieler darauf hinweist. Analog soll das Spiel auch auf einen erfolgreichen Kauf reagieren. Nach einem erfolgreichen Kauf eines Upgrades, sollen die Kosten des Upgrades von der Summe der Crypto Coins abgezogen werden.

Beispielhafte Preisverteilung:

- Bot Miner - 10 Coins
- RTX3090 - 50 Coins
- Bot Netzwerk - 100 Coins
- Serverraum - 500 Coins

Aufgabe 2

Dem Spieler des Crypto-Klickers sollen alle Upgrades die er bereits erhalten hat in einer Liste angezeigt werden. Erweitern Sie hierfür den Code um die Funktionalität, eine Liste unter

dem Spielfeld mit den aktuell erspielten Upgrades und deren Anzahl, zu rendern. Nutzen Sie hierfür das Vue Directive *v-for*.

Aufgabe 3

Erweitern Sie die Webseite nun um einen Optionen-Button. Beim Klicken soll dieser sich äufklappenünd zwei weitere Buttons anzeigen werden - Light- und Darkmode. Darkmode soll hierbei den Hintergrund des gesamte Spielfeldes (inkl. Upgrades und Options) schwarz anzeigen und die Schriftfarbe auf weiß ändern. Lightmode soll die Webseite auf ihren Ursprungszustand zurückführen, weißer Hintergrund mit schwarzer Schriftfarbe.