

Curso Técnico Superior Profissional em Tecnologias e Programação de Sistemas de Informação

Unidade Curricular: Desenvolvimento Web – Front-End

1º Ano/2º Semestre

Docente: Marco Miguel Olival Olim

Data 16/03/2018

ESTE EXERCÍCIO PRETENDE ILUSTRAR O FUNCIONAMENTO DO VUE JS REALIZANDO UM MINI-JOGO

Iniciar uma nova aplicação VueJS com esta template fornecida:

```
<!DOCTYPE html>
<html>

<head>
  <title>Monster Slayer</title>
  <script src="https://npmcdn.com/vue/dist/vue.js"></script>
  <link rel="stylesheet" href="foundation.min.css">
  <link rel="stylesheet" href="app.css">
</head>

<body>
  <div id="app">
    <section class="row">
      <div class="small-6 columns">
        <h1 class="text-center">YOU</h1>
        <div class="healthbar">
          <div class="healthbar text-center" style="background-color: green;
margin: 0; color: white;">

          </div>
        </div>
      </div>
      <div class="small-6 columns">
        <h1 class="text-center">MONSTER</h1>
        <div class="healthbar">
          <div class="healthbar text-center" style="background-color: green;
margin: 0; color: white;">

          </div>
        </div>
      </div>
    </section>
    <section class="row controls">
      <div class="small-12 columns">
        <button id="start-game">START NEW GAME</button>
      </div>
    </section>
  </div>
</body>
</html>
```

Cofinanciado por:

```

    </div>
  </section>
  <section class="row controls">
    <div class="small-12 columns">
      <button id="attack">ATTACK</button>
      <button id="special-attack">SPECIAL ATTACK</button>
      <button id="heal">HEAL</button>
      <button id="give-up">GIVE UP</button>
    </div>
  </section>
  <section class="row log">
    <div class="small-12 columns">
      <ul>
        <li>
          </li>
        </li>
      </ul>
    </div>
  </section>
</div>
</body>

</html>

```

Pode inicializar a instância do VueJS numa tag script ou criar um novo ficheiro de javascript e então indicá-lo no html

```

52   <script src="app.js"></script>
53   </body>
54   </html>

```

Em qualquer um dos casos inicializa-se a pontuação inicial e o estado inicial do jogo na propriedade data

```

1  new Vue({
2    el: '#app',
3    data: {
4      playerHealth: 100,
5      monsterHealth: 100,
6      gameIsRunning: false
7    }
8  });

```

Colocamos agora esta pontuação inicial nas barras de progresso correspondentes usando a interpolação de string

```

14   <div class="healthbar">
15     <div class="healthbar text-center" style="background-color: green;
16       {{ playerHealth }}
17     </div>
18   </div>

```

Cofinanciado por:



Se tudo estiver correto deverá visualizar a pontuação nas barras



Para alterar também as próprias barras realizamos o bind à propriedade de STYLE

```
14 <div class="healthbar">
15   <div
16     class="healthbar text-center"
17     style="background-color: green; margin: 0; color: white;"
18     :style="{width: playerHealth + '%'}">
19     {{ playerHealth }}
20   </div>
21 </div>
```

Para alternar entre começar novo jogo e os botões com as ações recorreremos à diretiva v-if e v-else do VueJS

```
35 <section class="row controls" v-if="!gameIsRunning">
36   <div class="small-12 columns">
37     <button id="start-game">START NEW GAME</button>
38   </div>
39 </section>
```

Para então começar o jogo alteramos este estado no botão com a diretiva v-on

```
37 <button id="start-game" @click="gameIsRunning = true">START NEW GAME</button>
```

Como este não é a única operação a realizar no início do jogo, devemos realizar um refactoring desta operação e inicializa-la num método próprio

```
1 new Vue({
2   el: '#app',
3   data: {
4     playerHealth: 100,
5     monsterHealth: 100,
6     gameIsRunning: false
7   },
8   methods: {
9     startGame: function() {
10      this.gameIsRunning = true;
11      this.playerHealth = 100;
12      this.monsterHealth = 100;
13    }
14  }
15 });
```

Adicionamos agora os métodos para as ações do jogo

```
41 <div class="small-12 columns">
42   <button id="attack" @click="attach">ATTACK</button>
43   <button id="special-attack" @click="specialAttack">SPECIAL ATTACK</button>
44   <button id="heal" @click="heal">HEAL</button>
45   <button id="give-up" @click="giveUp">GIVE UP</button>
46 </div>
```

Cofinanciado por:

Estes métodos têm agora de ser implementados na propriedade methods do VueJS

```
8      methods: {
9        startGame: function() {
10          this.gameIsRunning = true;
11          this.playerHealth = 100;
12          this.monsterHealth = 100;
13        },
14        attack: function() {
15
16        },
17        specialAttack: function() {
18
19        },
20        heal: function() {
21
22        },
23        giveUp: function() {
24
25        }
26      }
27    });
```

O Ataque, a título de exemplo, precisa de infligir um dano aleatório entre 3 e 10 ao monstro (10 e 20 no ataque extra) e entre 12 e 5 ao jogador, sendo a condição de paragem quando a pontuação for inferior a 0

```
14      attack: function () {
15        var max = 10;
16        var min = 3;
17        var damage = Math.max(Math.floor(Math.random() * max) + 1, min);
18        this.monsterHealth -= damage;
19
20        if (this.monsterHealth <= 0) {
21          alert('You won!');
22          this.gameIsRunning = false;
23          return;
24        }
25      }
```

Como utilizaremos esta fórmula para cálculo de danos nas diferentes ações, recomenda-se que efetue novo refactoring e coloque toda esta lógica num novo método **calculateDamage(min, max)** . Para manter registo destes ataques, inicializamos um Array **turns** cuja implementação poderá ter esta configuração

```
3      data: {
4        playerHealth: 100,
5        monsterHealth: 100,
6        gameIsRunning: false,
7        turns: []
8      },
9      methods: {
10        startGame: function () {
11          this.gameIsRunning = true;
12          this.playerHealth = 100;
13          this.monsterHealth = 100;
14        },
15        attack: function () {
16          var damage = this.calculateDamage(3, 10);
17          this.monsterHealth -= damage;
18          this.turns.unshift({
19            isPlayer: true,
20            text: 'Player hits Monster for ' + damage
21          });
22          if (this.checkWin()) {
23            return;
24          }
25        }
26      }
```

Cofinanciado por:

Agora para listar este registo utilizamos a diretiva **v-for** do VueJS

```
48 <section class="row log" v-if="turns.length > 0">
49   <div class="small-12 columns">
50     <ul>
51       <li v-for="turn in turns">
52         {{ turn.text }}
53       </li>
54     </ul>
55   </div>
56 </section>
```

Como estão disponíveis estilos apropriados para distinguir os ataques no ficheiro app.css disponibilizado

```
34 .log ul li {
35   margin: 5px;
36 }
37
38 .log ul .player-turn {
39   color: blue;
40   background-color: #e4e8ff;
41 }
42
43 .log ul .monster-turn {
44   color: red;
45   background-color: #ffc0c1;
46 }
47
48 button {
49   font-size: 20px;
```

Efetuamos então o bind à classe correspondente.

```
51 <li v-for="turn in turns"
52     :class="{ 'player-turn': turn.isPlayer, 'monster-turn': !turn.isPlayer}">
53   {{ turn.text }}
54 </li>
```

O jogo terá então o seguinte comportamento



Cofinanciado por: