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

Unidade Curricular: Desenvolvimento Web – Front-End — 1 º 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>
```

Cofinanciado por:

MADEIRA 14-20 — REGIÃO AUTÓNOMA DA MADEIRA — PORTUGAL 2020 — UNIÃO EUROPEIA Fundo Social Europeu

```
            </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>
                </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

```
    <script src="app.js"></script>
    </body>
    </html>
```

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

```
new Vue({
    el: '#app',
    data: {
        playerHealth: 100,
        monsterHealth: 100,
        gameIsRunning: false
    }
});
```

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

```
                <div class="healthbar">
                    <div class="healthbar text-center" style="background-color: green;
                        {{ playerHealth }}
                    </div>
                </div>
```

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

Se tudo estiver correto deverá visualizar a pontuação nas barras

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

```
<div class="healthbar">
    <div
            class="healthbar text-center"
            style="background-color: green; margin: 0; color: white;"
            :style="{width: playerHealth + '%'}">
        {{ playerHealth }}
    </div>
</div>
```

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

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

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

```
<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

```
new Vue({
    el: '#app',
    data: {
        playerHealth: 100,
        monsterHealth: 100,
        gameIsRunning: false
    },
    methods: {
        startGame: function() {
            this.gameIsRunning = true;
            this.playerHealth = 100;
            this.monsterHealth = 100;
        }
    }
});
```

Adicionamos agora os métodos para as ações do jogo

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

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

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

```
    methods: {
        startGame: function() {
            this.gameIsRunning = true;
            this.playerHealth = 100;
            this.monsterHealth = 100;
        },
        attack: function() {

        },
        specialAttack: function() {

        },
        heal: function() {

        },
        giveUp: function() {

        }
    }
});
```

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

```
        attack: function () {
            var max = 10;
            var min = 3;
            var damage = Math.max(Math.floor(Math.random() * max) + 1, min)
            this.monsterHealth -= damage;

            if (this.monsterHealth <= 0) {
                alert('You won!');
                this.gameIsRunning = false;
                return;
            }
```

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

```
    data: {
        playerHealth: 100,
        monsterHealth: 100,
        gameIsRunning: false,
        turns: []
    },
    methods: {
        startGame: function () {
            this.gameIsRunning = true;
            this.playerHealth = 100;
            this.monsterHealth = 100;
        },
        attack: function () {
            var damage = this.calculateDamage(3, 10);
            this.monsterHealth -= damage;
            this.turns.unshift({
                isPlayer: true,
                text: 'Player hits Monster for ' + damage
            });
            if (this.checkWin()) {
                return;
            }
```

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu

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

```
<section class="row log" v-if="turns.length > 0">
    <div class="small-12 columns">
        <ul>
            <li v-for="turn in turns">
                {{ turn.text }}
            </li>
        </ul>
    </div>
</section>
```

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

```
.log ul li {
    margin: 5px;
}

.log ul .player-turn {
    color: blue;
    background-color: #e4e8ff;
}

.log ul .monster-turn {
    color: red;
    background-color: #ffc0c1;
}

button {
    font-size: 20px;
```

Efetuamos então o bind à classe correspondente.

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

O jogo terá então o seguinte comportamento

Cofinanciado por:

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA Fundo Social Europeu