

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 21/03/2018

ESTE EXERCÍCIO PRETENDE ILUSTRAR O PROCESSO PARA ANIMAR ELEMENTOS COM VUE JS

Iniciar uma nova aplicação VueJS com esta template fornecida:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Exercício 7</title>
</head>
<body>
  <script src="https://nmpcdn.com/vue/dist/vue.js"></script>
  <div id="exercicio7"></div>
  <script>
    new Vue({
      el: '#exercicio7',
      data: {}
    });
  </script>
</body>
</html>
```

Vamos animar um elemento a surgir no html após ação de um botão. Este é um procedimento já realizado antes pelo que vamos repeti-lo da mesma forma:

```
<div id="exercicio7">
  <button @click="toggle = !toggle">botão</button>
  <div v-if="toggle">Texto</div>
</div>
<script>
  new Vue({
    el: '#exercicio7',
    data: {
      toggle: false
    }
  });
</script>
```

Cofinanciado por:



UNIÃO EUROPEIA
Fundo Social Europeu.

Para animar ou efetuar transições usamos o componente apropriado do VueJS designado por `<transition>` . Atenção que este só permite a animação de um único elemento de html:

```
<transition>
  <div v-if="toggle">Texto</div>
</transition>
```

Atribuímos um nome à animação e ficam automaticamente disponíveis pelo VueJS os estados *enter* e *leave* para indicarmos estas fases da animação.

```
<transition name="desvanecer">
  <div v-if="toggle">Texto</div>
</transition>
</div>
<style>
.desvanecer-enter {
  /* estado inicial da animação */
}
.desvanecer-enter-active {
  /* animação em curso */
}
.desvanecer-leave {
  /* estado final da animação */
}
.desvanecer-leave-active {
  /* animação em curso até estado inicial*/
}
</style>
```

A transição que pretendemos afeta apenas a visibilidade do elemento, pelo que usamos a propriedade opacity do CSS. A animação deverá ter uma duração de 3 segundos e é ativada pela ação do botão:

```
.desvanecer-enter {
  /* estado inicial da animação */
  opacity: 0;
}
.desvanecer-enter-active {
  /* animação em curso */
  transition: opacity 3s;
}
.desvanecer-leave {
  /* estado final da animação */
  opacity: 1;
}
.desvanecer-leave-active {
  /* animação em curso até estado inicial*/
  transition: opacity 3s;
  opacity: 0;
}
```

Cofinanciado por:



UNIÃO EUROPEIA
Fundo Social Europeu

Para o caso das animações usamos também os estados `enter` e `leave` mas temos também de definir os keyframes da animação. Criamos então uma nova animação designada por `deslizar`

```
<transition name="deslizar">
  <div v-if="toggle">Texto 2</div>
</transition>
</div>
<style>
  .deslizar-enter {
    transform: translateY(20px);
  }
  .deslizar-enter-active {
    animation: desliza-entrada 1s ease-out forwards
  }
  .deslizar-leave {
    /*o forwards mantem a posição do elemento que tinha
     no fim da animação, em vez de regressar à posição inicial*/
  }
  .deslizar-leave-active{
    animation: desliza-saida 1s ease-out forwards
  }
  @keyframes desliza-entrada {
    from {
      transform: translateY(20px);
    }
    to {
      transform: translateY(0);
    }
  }
  @keyframes desliza-saida {
    from {
      transform: translateY(0);
    }
    to {
      transform: translateY(20px);
    }
  }
}
```

No entanto, por vezes queremos animar elementos de uma lista dinâmica, como no exemplo do jogo em que temos um log de jogadas. Nestes casos utilizamos o componente `<transition-group>` que tem basicamente o mesmo comportamento que `<transition>`. Neste exemplo definimos primeiro o CSS

```
<style>
  .fade-enter-active {
    transition: opacity 0.5s;
  }

  .fade-enter {
    opacity: 0;
  }
</style>
```

Cofinanciado por:



UNIÃO EUROPEIA
Fundo Social Europeu

De seguida envolvemos as linhas da lista do log com a tag transition-group que aplicará a transição fade a cada novo elemento da lista. O appear serve apenas para o primeiro elemento aparecer já com esta transição.

```
<section class="row log" v-if="turns.length > 0">
  <div class="small-12 columns">
    <ul>
      <transition-group name="fade" appear>
        <li v-for="turn in turns"
            :class="{'player-turn': turn.isPlayer,
                      'monster-turn': !turn.isPlayer}"
            :key="turn.id">
          {{ turn.text }}
        </li>
      </transition-group>
    </ul>
  </div>
</section>
```

Em todos estes exemplos foram criadas por nós todas as transições e animações, mas existem bibliotecas que disponibilizam já um vasto número de animações, como o Animate.css

Animate.css

Just-add-water CSS animations

bounce

Animate it

[Download Animate.css](#) or [View on GitHub](#)

Another thing from [Daniel Eden](#).

Podemos utilizar diretamente esta biblioteca a partir de um CDN:

```
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.5.2/animate.min.css">
```

Cofinanciado por:



UNIÃO EUROPEIA
Fundo Social Europeu

Para usar na tag transition do VueJS não podemos atribuir um nome a esta como tem sido feito, mas introduzir diretamente os estados *enter* e *leave* na tag transition

```
<transition enter-active-class="animated bounce" leave-active-class="animated shake">
  <div v-if="toggle">Texto 3</div>
</transition>
```

Há também a possibilidade de utilizar javascript em vez de CSS para animações. Neste caso os hooks são:

```
<transition
  @before-enter="beforeEnter"
  @enter="enter"
  @after-enter="afterEnter"
  @enter-cancelled="enterCancelled"

  @before-leave="beforeLeave"
  @leave="leave"
  @after-leave="afterLeave"
  @leave-cancelled="leaveCancelled">
  <div style="width: 100px; height: 100px;">
</transition>
```

A programação das animações é efetuada em methods, como de costume:

```
    elementWidth: 100
  }
},
methods: {
  beforeEnter(el) {
    console.log('beforeEnter');
    this.elementWidth = 100;
    el.style.width = this.elementWidth + 'px';
  },
  enter(el, done) {
    console.log('enter');
    let round = 1;
    const interval = setInterval(() => {
      el.style.width = (this.elementWidth + round * 10) + 'px';
      round++;
      if (round > 20) {
        clearInterval(interval);
        done();
      }
    }, 20);
  },
  afterEnter(el) {
```

Cofinanciado por:

