08

Lidando com argumentos

Transcrição

Excelente, nossa diretiva de transformação funciona rotacionando qualquer elemento na tela. Aliás, você a adicionou no `meu-painel` ao invés de utiliza-la em `imagem-responsiva` ? Bem, eu prefiro seu uso em `imagem-responsiva` e o design, amigo imaginário, também.

Mas vejamos o seguinte. E se quisermos, no lugar de rotacionar a imagem quisermos ampliá-la? Veja que o modificador `animacao` faz sentido, já o reverse, não muito. Para isso, precisamos trabalhar com argumentos.

Vamos alterar a chamada da nossa diretiva no template de `Home` :

```
<!-- alurapic/src/components/home/Home.vue -->

<imagem-responsiva :url="foto.url" :titulo="foto.titulo" v-
meu-transform:rotate.animate.reverse="15"/>
```

COPIAR CÓDIGO

Veja que logo após o nome da diretiva foi adicionando dois ponto seguido do nome do parâmetro. No caso, estamos passando `rotate` . Aliás, esse será o efeito padrão quando nenhum argument for passado. Agora, vamos alterar nossa diretiva para levar em consideração esse argumento:

```
// alurapic/src/directives/Transform.js

import Vue from 'vue';
```

```
Vue.directive('meu-transform', {

    bind(el, binding, vnode) {

      let current = 0;

      el.addEventListener('dblclick', function() {

        let incremento = binding.value || 0;

        let efeito;

        if(!binding.arg || binding.arg == 'rotate') {

          if(binding.modifiers.reverse) {
            current-=incremento;
          } else {
            current+=incremento;
          }
          efeito = `rotate(${current}deg)`;
        }

        this.style.transform = efeito;

        if (binding.modifiers.animate) this.style.transition
= "transform 0.5s";

      });
    }

});
```

COPIAR CÓDIGO

É através de `binding.arg` que verificamos a existência ou não de um argumento. Um teste indica que tudo continua funcionando. Por fim, vamos adicionar a condição que aumenta o tamanho da imagem:

```
// alurapic/src/directives/Transform.js

import Vue from 'vue';

Vue.directive('meu-transform', {

    bind(el, binding, vnode) {

      let current = 0;

      el.addEventListener('dblclick', function() {

        let incremento = binding.value || 0;

        let efeito;

        if(!binding.arg || binding.arg == 'rotate') {

          if(binding.modifiers.reverse) {
            current-=incremento;
          } else {
            current+=incremento;
          }
          efeito = `rotate(${current}deg)`;

        } else if(binding.arg == 'scale') {

          efeito = `scale(${incremento})`;
        }

        this.style.transform = efeito;

        if (binding.modifiers.animate) this.style.transition
= "transform 0.5s";

      });
    }
```

```
});
```

COPIAR CÓDIGO

Veja que o modificador `reverse` será ignorado quando o argumento for `scale`. Isso não é feio não, podemos ter vários modificadores que se aplicam a um tipo de argumento. Outro ponto que mudamos é o valor padrão de 90 para 0. Não faz sentido usar um scale com 90!

Hoje, nosso componente esta assim:

```
<!-- alurapic/src/directives/Transform.js -->

<imagem-responsiva :url="foto.url" :titulo="foto.titulo" v-meu-transform:rotate.animate.reverse="30"/>
```

COPIAR CÓDIGO

Agora, vamos mudar para scale. Podemos remover o modificador `reverse` já que ele não terá afeito algum quando usarmos `scale`:

```
<!-- alurapic/src/directives/Transform.js -->

<imagem-responsiva :url="foto.url" :titulo="foto.titulo" v-meu-transform:scale.animate="1.2"/>
```

COPIAR CÓDIGO

Podemos também não usar animação omitindo o modificador `animate` e por ai vai. Por fim, para que a imagem rotacionada e ampliada não fazer de dentro do nosso painel podemos alterar nosso componente `Painel` e adicionar o estilo `overflow: hidden`:

```
<!-- alurapic/src/components/painel/Painel.vue -->
```

```
<template>

 <!-- código omitido -->

</template>

<script>

export default {

   // código omitido
}
</script>

<style>

  /* estilos anteriores omitidos */

  .painel-conteudo {
    overflow: hidden;
  }

</style>
```

COPIAR CÓDIGO

O mais bacana é que essa alteração no estilo do nosso componente se refletirá em todos os lugares que ele for utilizado.