03

Criando nossa primeira diretiva

Transcrição

Vamos iniciar os trabalhos criando o arquivo `alurapic/src/directives/Transform.js`. Veja que o arquivo não é um single file template com a extensão `.vue`, mas um arquivo `.js` como outro script qualquer.

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

import Vue from 'vue';
```

COPIAR CÓDIGO

Veja que a primeira instrução da nossa diretiva é a importação do global view object `Vue`. É através dele que registraremos nossa diretiva para que seja acessível pelas views da nossa aplicação.

O próximo passo é invocar o método `Vue.directive` que recebe como parâmetro o nome da diretiva e como segundo um objeto JavaScript que conterá as configurações da diretiva:

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

import Vue from 'vue';

Vue.directive('meu-transform', {

  /* ainda sem qualquer configuração */
```

```
});
```

COPIAR CÓDIGO

Nossa diretiva ainda não esta completa, mas quando formos utilizá-la em nossos template somos obrigados a utilizar o prefixo `v-` . Veja que diretivas como `v-for` e `v-show` possui o mesmo prefixo. Sendo assim, para utilizarmos nossa diretiva precisaremos fazer `v-meu-transform` .

A função bind e acesso ao elemento do DOM

Agora que você já entendeu como usaremos a diretiva que estamos criando em nossos templates podemos partir para sua configuração. O objeto passado como segundo parâmetro precisa ter o método `bind` , um *hook* chamado toda vez que a diretiva é associada ao elemento do DOM. O método recebe três parâmetros. O nome desses três parâmetros podem ser qualquer um, no entanto vamos utilizar aqueles que deixaram claro o papel de cada um:

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

import Vue from 'vue';

Vue.directive('meu-transform', {

    bind(el, binding, vnode) {

    }

});
```

COPIAR CÓDIGO

Por enquanto, vamos no ater apenas ao primeiro parâmetro da diretiva, que nada mais é do que o elemento do DOM no qual a diretiva esta sendo aplicad Vamos imprimir no console esse parâmetro:

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

import Vue from 'vue';

Vue.directive('meu-transform', {

    bind(el, binding, vnode) {

        console.log('Diretiva iniciada');
        console.log(el);

    }

});
```

COPIAR CÓDIGO

Por enquanto vamos deixar essa implementação e experimentar utilizá-la em nossa aplicação.

Importando nossa diretiva

Em nenhum momento carregamos o módulo `Transform.js`. Todo arquivo `.js` e `.vue` criados são módulos que precisam ser importados. Faremos isso em nosso arquivo `alurapic/src/main.js`:

```
// alurapic/src/main.js

import Vue from 'vue'
import App from './App.vue'
import VueResource from 'vue-resource';
import VueRouter from 'vue-router';

import { routes } from './routes';

// importando o arquivo `Transform.js`.
```

```
import './directives/Transform';

Vue.use(VueRouter);

// código posterior omitido
```

COPIAR CÓDIGO

Veja que a cláusula `import` precisa receber apenas o caminho do arquivo. É desta forma porque o módulo `Transform.js` não exporta código algum, precisamos apenas carregá-lo para que nossa diretiva seja registrada através do global view object. Excelente! Isso já é suficiente para que nossa diretiva esteja ativa.

Agora, vamos abrir `alurapic/src/components/home/Home.vue` e adicionar a diretiva no elemento `<imagem-responsiva/ >`:

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

<!-- código anterior omitido -->

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

<!-- código posterior omitido
```

COPIAR CÓDIGO

Não esqueça que o Vue CLI precisa estar de pé para que possamos visualizar o resultado em nosso navegador. Assim que a página for atualizada, abra o console do navegador e veja o resultado. Veja que para cada componente ele exibirá o elemento do DOM correspondente àquele componente.

Agora que já vimos que nossa diretiva esta atuando para cada componente `ImagemResponsiva` criado dinamicamente através da diretiva `v-for` podemos dar início a implementação da lógica de rotação.

Um ponto importante é que cada diretiva é aplicada isoladamente para cada elemento, sendo assim, cada diretiva pode ter suas variáveis de controle específicas para cada elemento. No caso, vamos declarar a variável `current` que será iniciada assim que nossa diretiva for aplicada no elemento. Ela guardará o eixo atual utilizado para rotacionarmos o elemento. Todos começarão de zero:

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

import Vue from 'vue';

Vue.directive('meu-transform', {

    bind(el, binding, vnode) {
      let current = 0;

    }

});
```

COPIAR CÓDIGO

Implementando nossa lógica

A cada duplo clique que realizamos, vamos incrementar a variável current em `90` graus. Mas como trabalharemos com evento dentro da nossa diretiva? Se fosse no componente, poderíamos usar @dblclick no template, mas veja que aqui não temos template. A boa notícia é o fato de `el` ser o elemento do DOM no qual a diretiva esta associada e, sendo um elemento do DOM, podemos adicionar evento da maneira tradicional no mundo JavaScript através de `addEventListener`:

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

import Vue from 'vue';
```

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

    bind(el, binding, vnode) {
      let current = 0;
      el.addEventListener('dblclick', function() {
        current+=90;
      });
    }

});
```

COPIAR CÓDIGO

A cada clique o valor de `current` da imagem que estamos clicando será incrementado. Agora, basta aplicamos uma transformação do tipo `rotate` do CSS para conseguimos o evento desejado. Via DOM, aplicamos um estilo diretamente no elemento através de `elemento.style.nomeDoPropriedade` :

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

import Vue from 'vue';

Vue.directive('meu-transform', {

    bind(el, binding, vnode) {
      let current = 0;
      el.addEventListener('dblclick', function() {
        current+=90;
        this.style.transform = `rotate(${current}deg)`;
      });
    }

});
```

COPIAR CÓDIGO

Aqui usamos *template string* do ES2015 para nos ajudar a construir a string dinamicamente com base no valor atual da variável `current`. Muito mais interessante do que se tivéssemos feito uma concatenação. Templates strings são criadas usando crase (backstick) e no lugar da concatenação, usamos `${nomeDaVariavel}` para interpolar dentro da string valores de variáveis. Se você quer saber mais sobre ES2015 e template strings, sugiro cursar meu curso de JavaScript avançado aqui da Alura.

Vamos testar o resultado? Com a página recarregada, quando clicamos duas vezes nas imagens elas são rotacionadas de 90 em 90 graus. No diretiva é tão genérica que podemos adicioná-la ao título da página, permitindo sua rotação no duplo clique:

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

<template>
    <div>
        <h1 class="centralizado" v-meu-
transform>Alurapic</h1>

<!-- código posterior omitido -->
```

COPIAR CÓDIGO

Faça um teste e veja o resultado!

Excelente. Mas se quisermos incrementar de 50 em 50? Para isso nossa diretiva precisará receber a quantidade em graus do incremento. É isso que veremos no próximo vídeo.