03

Isolando marcação e estilos em um novo componente

Transcrição

As chances de querermos utilizar o mesmo painel em outras páginas da nossa aplicação não são pequenas. E para que isso seja possível, somos obrigados a copiar o código HTML e o CSS. Mesmo em uma aplicação Web tradicional na qual podemos importar o mesmo CSS em várias páginas, a marcação HTML terá que ser refeita.

Durante esse processo, essa ou aquela classe pode ser omitida ou essa ou aquela tag pode não ter sido fechada o que pode ocasionar problemas. Mais ainda, se a estrutura do painel mudar, seremos obrigados a alterar em todos os lugares. A boa notícia é que isso não precisa ser assim.

Podemos tornar nosso painel um componente de `Vue` e reutilizá-lo em qualquer outro local da nossa aplicação. Dizemos que nosso painel será um componente `shared`, compartilhado. O mesmo não ocorre com `App.vue` que é específico da nossa aplicação e não faz sentido ser reutilizado por outra aplicação ou ainda dentro da mesma aplicação.

Criando um shared component

Dessa forma, vamos criar a pasta `alurapic/src/components/shared/painel`. Todos nosso componentes que criarmos a partir de agora ficarão dentro da pasta `components`. Além disso, todos aqueles que forem componentes reutilizáveis e compartilháveis ficarão dentro da subpasta `shared`. Criaremos outra subpasta por componente e dentro dela teremos o arquivo `.vue`. Sendo assim vamos seguir a convenção de criar arquivos de componente começando em pascal case:

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

<template>
</template>

<script>

export default {
}
</script>
<style>
</style>
```

Agora que temos o esqueleto básico do nosso componente constituído pelas tags `template`, `script` e `style`, vamos mover a marcação do painel e seu estilo para dentro do componente:

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

<template>

    <div class="painel">

      <h2 class="painel-titulo"></h2>
      <div class="painel-conteudo">

      </div>
```

```
    </div>

</template>

<script>

export default {
}
</script>
<style>
 .painel {
    padding: 0 auto;
    border: solid 2px grey;
    display: inline-block;
    margin: 5px;
    box-shadow: 5px 5px 10px grey;
    width: 200px;
    height: 100%;
    vertical-align: top;
    text-align: center;
  }

  .painel .painel-titulo {
    text-align: center;
    border: solid 2px;
    background: lightblue;
    margin: 0 0 15px 0;
    padding: 10px;
    text-transform: uppercase;
  }

</style>
```

Nosso componente ainda não esta pronto, mas assim que estiver poderemos utilizá-lo em `App.vue` da seguinte maneira:

```
<!-- alurapic/src/App.vue -->

<template>
  <div class="corpo">

    <h1 class="centralizado">{{ titulo }}</h1>

    <ul class="lista-fotos">

      <li class="lista-fotos-item" v-for="foto in fotos">

        <meu-painel :titulo="foto.titulo">
          <img class="imagem-responsiva" :src="foto.url" :alt="foto.titulo">
        </meu-painel>

      </li>
    </ul>

  </div>
```

```
</template>
<!-- código posterior omitido -->
```

Veja como ficou simples a marcação do de `App.vue`. No caso, estamos usando o componente `Painel.vue` como `meu-painel` da definição do template de `App.vue`. Além disso, veja que o componente recebe seu título na propriedade `titulo`. Aliás, esse é um ponto que precisamos nos debruçar.

Todo componente em Vue é uma unidade de código que pode encapsular sua marcação, estilo e comportamento, este último, ações que podem ser realizadas com ele. Por enquanto, só disponibilizamos dados para o template e não executamos nenhum comportamento, algo que veremos ainda nesse curso.