01

Adequando o template gerado

Transcrição

Agora que temos uma visão geral de como cada parte do projeto se encaixa, inclusive a aprendizagem do conceito de data binding através de interpolação, podemos começar a dar ao projeto a cara da aplicação que desejamos construir.

Template e view root

Vamos apagar o conteúdo do template de `alurapic/src/App.vue` deixando apenas o esqueleto necessário para criarmos nossa primeiro componente:

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

<template>
</template>

<script>
export default {

}
</script>

<style>
</style>
```

Nosso componente `App` , o primeiro a ser exibido na view `index.html` exibirá uma lista de fotos. Nesse sentido, vamos adicionar uma tag `img` no template de `alurapic/src/App.vue` para exibir uma imagem qualquer da nossa escolha, inclusive você pode pesquisar essa imagem no próprio Google e utilizar seu endereço.

Alterando `src/App.vue` :

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

<template>

  <img src="https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH_m(

</template>

<script>
export default {

}
</script>

<style>
</style>
```

Pelo mecanismo de live reloading, assim que salvarmos o arquivo nosso navegador será carregado exibindo nossa imagem.

Faltou o título da página, vamos colocar uma tag `h1` :

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

<template>

  <h1>Alurapic</h1>

  <img src="https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH_m(

</template>

<script>
export default {

}
</script>

<style>
</style>
```

Surpresa! Nada é exibido no navegador. Se olharmos o terminal temos a seguinte mensagem:

```
ERROR in ./~/vue-loader/lib/template-compiler.js?id=data-v-6a8232ee!./~/vue-loader/lib/selector
template syntax error Component template should contain exactly one root element:

<h1>Alurapic</h1>
<img src="https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH_mOr

If you are using v-if on multiple elements, use v-else-if to chain them instead.
 @ ./src/App.vue 11:23-143
 @ ./src/main.js
 @ multi main
```

Extraindo dessa mensagem de erro a mensagem mais importante:

```
App.vue
template syntax error Component template should contain exactly one root element:
```

Isso acontece, porque se quisermos exibir mais de um elemento dentro da tag `template` , os elementos devem estar dentro de um elemento pai. Nesse caso, uma tag `div` resolve esse problema:

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

<template>

  <div>
```

```
    <h1>Alurapic</h1>

    <img src="https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH

  </div>

</template>

<script>
export default {

}
</script>

<style>
</style>
```

Perfeito, conseguimos exibir uma imagem, mas se fosse para exibir a imagem assim como fizemos não precisaríamos do Vue. Queremos exibir a imagem dinamicamente baseada nos dados da foto que lhe forem passados. Mas antes de passarmos os dados para a a tag `img`, vamos fazer o seguinte. Forneceremos para o template de `App.vue` o título da página.

Data binding unidirecional através de interpolação

Aprendemos que é a função `data` que deve retornar um objeto JavaScript com os dados de que o template da nossa view precisa:

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

  <div>
    <h1>{{ titulo }}</h1>

    <img src="https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH

  </div>

</template>

<script>
export default {

  data() {
    return {
      titulo: 'Alurapic'
    }
  }
}
</script>

<style>
</style>
```

Excelente, recarregando nossa página o título `Alurapic` continua sendo exibido. Se trocarmos por outro, automaticamente o navegador recarregará e usará a nova informação exibindo-a para o usuário. Vale ressaltar que interpolações com `{{ }}` realizam uma associação unidirecional da fonte de dados para o template sempre nesse sentido. Sendo assim, mudanças nos dados acarretam mudanças no template do nosso componente.

Agora, vamos retornar os dados da foto. No caso, vamos usar os mesmos que usamos antes:

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

  <div>
    <h1>{{ titulo }}</h1>

    <img src="" alt="">

  </div>

</template>

<script>
export default {

  data() {
    return {
      titulo: 'Alurapic',
      foto: {
        url: 'https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH_
        titulo: 'Cachorro'
      }
    }
  }
}
</script>

<style>
</style>
```

Veja que movemos o valor de src e de alt da tag `img` para a propriedade do objeto `foto` retornando também pela função `data` . E agora, como acessaremos essas informações no template. Será que podemos utilizar interpolação assim como fizemos com o `titulo` ?

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

<template>

  <div>

    <h1>{{ titulo }}</h1>

    <img src="{{ foto.url }}" alt="{{ foto.titulo }}">

  </div>

</template>
```

```
<script>
export default {

 // código omitido

}
</script>

<style>
</style>
```

Nossa página esta em branco no navegador! Olhando no terminal temos a seguinte mensagem:

```
ERROR in ./~/vue-loader/lib/template-compiler.js?id=data-v-6a8232ee!./~/vue-loader/lib/selector
template syntax error src="{{ foto.url }}": Interpolation inside attributes has been removed. Us
 @ ./src/App.vue 11:23-143
 @ ./src/main.js
 @ multi main

ERROR in ./~/vue-loader/lib/template-compiler.js?id=data-v-6a8232ee!./~/vue-loader/lib/selector
template syntax error alt="{{ foto.titulo }}": Interpolation inside attributes has been removed
 @ ./src/App.vue 11:23-143
 @ ./src/main.js
 @ multi main
```

A parte que nos interessa é:

```
template syntax error src="{{ foto.url }}": Interpolation inside attributes has been removed. Us
```

E agora?

A diretiva v-bind

Não podemos usar interpolação em atributos! Precisamos fazer de outra maneira, aliás essa maneira possui duas formas. A primeira é usarmos `v-bind` :

```
<template>

  <div>
    <h1>{{ titulo }}</h1>

    <img v-bind:src="foto.url" v-bind:alt="foto.titulo">

  </div>

</template>

<script>
export default {
```

```
  // código omitido
}
</script>

<style>
</style>
```

Assim que realizamos a alteração, nossa página volta a ser exibida, inclusive a tag `img` com os dados da foto. Usamos a sintaxe `v-bind:nomeDoAtributo`. O valor é atribuído diretamente, sem `{ {}}` como `v-bind:src="foto.url"`. É o Vue que fará a interpolação por debaixo dos panos.

O que acabamos de ver em ação é o uso de uma **diretiva** do Vue. Diretivas nada mais são do que um código interpretado pelo Vue que encapsula determinada funcionalidade ensinando novos truques para o navegador. Tanto isso é verdade que no mundo HTML não existe `v-bind`, logo, esta diretiva esta sendo interpretada pelo Vue.

Um atalho elegante para v-bind

No entanto, pode parecer um tanto verbo usar a sintaxe `v-bind` para realizar uma associação unidirecional que vai da fonte de dados para a view. Nesse caso podemos trocar `v-bind` pelo seu atalho dois pontos:

```
<template>

  <div>
    <h1>{{ titulo }}</h1>

    <img :src="foto.url" :alt="foto.titulo">

  </div>

</template>

<script>
export default {

 // código omitido

}
</script>

<style>
</style>
```

É uma sintaxe muito mais enxuta equivalente ao `v-bind`. Aliás, se não quisermos usar interpolação para o conteúdo de uma tag, podemos usar a diretiva `v-text`, em nosso exemplo ficaria `<h1 v-text="titulo"></h1>`. No entanto, eu usarei interpolação até o final do treinamento para fornecer o conteúdo de tags.