08

Global View Object

Transcrição

Em index.html, quando nossa aplicação é carregada, a tag `<div id="app"></div>` é substituída pelo componente `App` que acabamos de escrutinar, mas isso não ocorre por padrão. Alguém precisa explicitar isso na inicialização da nossa aplicação, a tarefa realizada por `alurapic/main.js` . Vamos verificar esse arquivo.

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

import Vue from 'vue'
import App from './App.vue'

new Vue({
  el: '#app',
  render: h => h(App)
})
```

O arquivo `alurapic/src/main.js` , assim como `App.vue` é um módulo do ES2015:

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

import Vue from 'vue'
import App from './App.vue'

// código posterior omitido
```

Sendo um módulo do ES2015, para termos acesso a outros artefatos de outros módulos da nossa aplicação, precisamos explicitar qual artefato de qual módulo desejamos importar. Como este é o arquivo responsável em exibir nosso componente `App` precisamos importá-lo, o que é feito na segunda instrução do arquivo. Como só temos um componente em `App.vue` e a sintaxe `export default` foi utilizada, usamos como nome do artefato a ser importado o mesmo nome do arquivo, sem a extensão `.vue` .

No entanto, antes de importar `App` , o módulo `alurapic/main.js` importa `Vue` (maiúsculo) do módulo `vue` . O módulo vue esta localizado dentro da pasta `alurapic/node_modules` e o CLI já tem tudo configurado para que a pasta do módulo seja enxergada pela instrução import. O artefato `Vue` é o **Global View Object**, um objeto especial do Vue.js, seu core.

View Instance

Para que possamos carregar, ou melhor, renderizar nosso componente `App` em `<div id="app"></div>` de `index.html` , precisamos criar uma **view instance** com auxílio do global view object que importamos:

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

// código anterior omitido

// criando uma view instance
new Vue({
```

```
 /* código omitido */
})
```

Uma instância de view é uma ponte entre nossos componentes e view. Em nosso caso, nossa view é o `index.html`. Mas a view instance não fez "curso mãe Diná" para saber qual componente deve carregar e o local onde deve inseri-lo em `index.html`, sendo necessário passar essa informação para a ela.

Renderizando um template

Vejamos nossa view instance devidamente configurada:

```
// alurapic/src/main.js
// código anterior omitido

new Vue({
  el: '#app',
  render: h => h(App)
})
```

A propriedade `el` recebe como parâmetro o seletor do elemento que será substituído pelo nosso componente, já na função `render` passamos o componente que desejamos renderizar.

Agora que já entendemos como as coisas se encaixam em nossa aplicação, vamos voltar para nosso componente `App` para compreender um conceito importante logo neste primeiro capítulo.