Para usar o Vue JS

Instalar o node.js

o cli ajuda a criar os arquivos necessários para rodar o vue

Instalar o npm install [vue-cli@2.7.0](mailto:vue-cli@2.7.0) -g

Cria o template que será usado para o projeto ( Tem outros no site do vue )

Vue init webpack alurapic

Volta no diretório

Cd alurapic

Instala todas as depedencias

Npm install

Vai rodar o projeto no servidor e abrir o navegador

Npm run dev

Entrar na pasta da api

Executar npm start

//Vai retornar a mensagem API Escutando na porta: 3000, depois só ver a página

O VUE através faz a transcompilação de todo o código escrito usando ES2015 para ES5 garantindo maior compatibilidade da nossa aplicação, mais notadamente em browsers desatualizados de smartphones o Vue CLI utiliza respectivamente **Babel** e **WebPack**

Componentes nada mais são do que um objeto que possui sua apresentação, dado e comportamento.

É por isso que nosso componente é definido através de três grandes blocos: template (apresentação), script (comportamento e dados) e style (o estilo da apresentação)

Importamos do módulo vue o global view object Vue. Inclusive, é por intermédio de uma view instance criada a partir do global view object que renderizamos um componente em index.html.

//Este Vue é global do próprio vue

import Vue from 'vue'

import App from './App.vue'

new Vue({

el: '#app', //aonde ele vai renderizar

render: h => h(App) // daonde esta vindo

})

//A função data sempre precisa retornar um objeto javascript.

<script>

export default {

name: 'app',

data () {

return {

msg: 'Seja bem-vindo à sua Vue.js App'

}

}

}

</script>

As propriedades destes objetos são acessessiveis no template utilizando as tags {{ }}

Interpolação é um tipo de data binding

Não podemos usar interpolação em atributos

<template>

<div>

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

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

//v-bind ou : nos atributos

</div>

</template>

<script>

export default {

// código omitido

data() {

return {

titulo: 'Alurapic',

foto: {

url: 'https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTwV4kVzT5McBdGSgqlVeRzubrNH\_mOrrkKseDOGFURq20HmsrelEfMU7It',

titulo: 'Cachorro'

}

}

}

}

}

</script>

<style>

</style>

se tivermos mais de um elemento dentro da tag template eles devem vir dentro de um elemento raiz (root), por exemplo, uma div

Exemplo da entrada de dados



a diretiva v-for que permite realizar laço dentro do nosso template,



Pode se usar dessas duas formas

v-for="nome of nomes"

v-for="nome in nomes"

O VueResource é responsável pelas requisições igual o fetch api , xlHTTPRequet etc

npm install vue-resource@1.0.3 --save

Após instalar é precisso registrar no arquivo main.js

import VueResource from ‘vue-resource’

// registrando o módulo/plugin no global view object

Vue.use(VueResource)

props. Nela podemos passar uma lista de propriedades que podem ser recebidas pelo componente

É por isso que dentro da tag que representa o título do componente, usamos

Aqui estamos dizendo que o painel vai receber o titulo e vai passar para os elementos filhos neste caso o h1 onde esta o titulo la no painel eu vou criar um props e ai vou poder usar {{ titulo }}. No h1 dentro do painel

<meu-painel :titulo="foto.titulo">

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

        </meu-painel>

No arquivo App vue vamos importar o arquivo é preciso usar o mesmo nome que esta no arquivo

import Painel from './components/chared/painel/Painel.vue';

export default {

Eu preciso dentro de export default criar uma nova propriedade e dizer qual é o componente e a origem dele

components: {

      'meu-painel' : Painel

  },

O SLOT ele permite dizer aonde será recebido o conteúdo, para que o elemento não seja apagado quando for criado o painel.

<template>

    <div class="painel">

        <h2 class="painel-titulo">{{ titulo }}</h2>

        <slot class="painel-conteudo">

        </slot>

    </div>

</template>

Foi usado isso porque dentro do Painel tinha uma img que estava sendo apagada

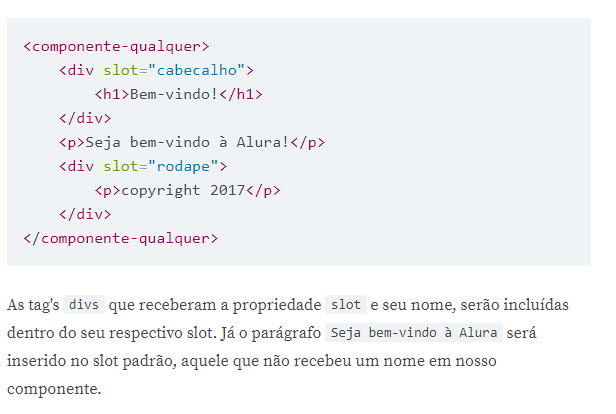
   <meu-painel :titulo="foto.titulo">

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

        </meu-painel>

Todos os estilos da páginas eles são globias, ou seja afetam não só o componente, mas o arquivo principal, para tornar o componente utilizavel é preciso usar o scoped dentro da tag style

<style scoped>



v-on:input="filtro = $event.target.value"

Toda vez que este evento for disparado

Ele pegaára o filtro dentro do objeto data o campo tem o nome de filtro e este filtro receberá $event.target.value"

Para verificar podemos soltar dentro do template um {{ filtro }} este é um databind que o vue jogará o valor neste campo

A diretiva v-bind realiza uma associação que flui da fonte de dados para a view. Qualquer mudança na fonte de dados fará com que a view seja atualizada para representar o novo valor. Já a diretiva v-on realiza uma associação de evento que flui da view para a fonte de dados. Para que as mudanças realizadas por v-on sejam atualizadas na view, é necessário combinar v-on com v-bind.

Um substituo para v-on é @ basta usar @ que funcionará do mesmo jeito

tipo assim @dblclick="visivel

Queremos esconder o conteúdo do painel, não removê-lo. Para isso há a diretiva v-show. Quando adicionada em um elemento, quando seu valor for true, o elemento será exibido, se for false, será ocultado. Por debaixo dos panos a diretiva realiza um display: none.

<div class="painel-conteudo" v-show="visivel">

<slot></slot>

</div>

data(){

return{

visivel: true

}

}

Para utilizar o evento doubleClick

Usar a diretiva

v-on:dblclick=”visível = !visivel”

neste caso se visível estiver true ele recebe false

O Transition é utilizado para animar, quando utilizamos com o name name="painel-fade", é utilizado delimita o elemento, que o vue adicionará alguma classe nos elementos que ele esta abraçando, este name é importante por que os nomes das classes que forem ser criadas levará em consideração os names no css

<transition name="painel-fade">

<div class="painel-conteudo" v-show="visivel">

<slot></slot>

</div>

<transition>

Como ficou o transition

. .painel-fade-enter{ }

.painel-fade-enter-active{ }

.painel-fade-leave-active{ }

painel-fade-enter // antes do elemento ser incluído ou removido, o estado atual

painel-fade-enter-active // quando o elemento esta sendo incluído

painel-fade-leave-active // quando o elemento esta sendo removido

Para que seja exibido /#/; na URL com o Router

Precisa independente do endereço que estiver acessado , precisa sempre devolver o endereço indexHTML

asd