Mini Curso de Vue.js

vuejs.org @vuejs vuejs/vue

AZ Informática

Empresa de desenvolvimento de software

Há 30 anos no mercado e reconhecida em âmbito nacional, a AZ Tecnologia em Gestão desenvolve soluções Administrativas, tais como solicitações de compras, licitações, gestão de fornecedores e almoxarifado, patrimônio, contrato e etc.

azi.com.br azinformatica

Eli Gabilon

Desenvolvedor full-stack e Especialista Técnico da empresa AZ Tecnologia em Gestão.

Formado em Processos Gerenciais, Ciência da Computação, mestrando em Ciência da Computação aplicada à Robótica e Sistemas embarcados e Aplicações distribuídas.

Apreciador de IOT, e placas como Arduino, ESP para automação residencial.

eligabilon

Jeferson Urbieta

Desenvolvedor full-stack, Especialista Técnico na empresa AZ Tecnologia em Gestão.

Formado em Análise e Desenvolvimento de Sistemas e freelancer nas horas vagas. Apaixonado por tecnologia e inovação, sempre buscando coisas divertidas para fazer na área de programação.

jefersonurbieta

O que é Vue.js?

Vue.js

- Framework javascript para construção de **views** em aplicações web, que visa melhorar a manipulação do DOM
- Criado pelo chinês **Evan You**
- Lançado em fevereiro de 2014
- Mais de **150 mil** estrelas no github
- Competindo com frameworks de grandes empresa como Angular da google e React do facebook

Benefícios do Vue.js

- Framework progressivo
- Componentizado
- Flexível e Performático
- Curva de aprendizado baixa
- Robusto

Instância Vue

Instância Vue

Toda aplicação Vue é iniciada com a criação de uma nova instância Vue com a função Vue

```
new Vue({
   // opções
})
```

- Esta instância será a raiz de toda a aplicação
- Neste objeto de opções que passamos, vamos descrever o comportamento de nossa aplicação

Instância Vue (aplicação)

Uma opção importante para instância vue é a descrição de como vamos renderizar a aplicação

```
new Vue({
   // opções
})
```

- Isso diz para a instância vue que vamos renderizar nossa aplicação dentro da tag com o id '#app'

```
<div id="app"></div>
```

Dados

Uma propriedade super interessante da instância vue e de seus componentes é o **data**, onde declaramos nossas **variáveis**

```
new Vue({
 el: '#app',
 data() {
   return {
     mensagem: 'olá mundo!'
   }
 }
})
```

Interpolação

Agora que já temos nosso variável declarada, podemos **apresentá-la** na tela através da interpolação

```
{{ mensagem }}
```

Isto faz com que no momento que for renderizar essa chave, ela seja substituída pelo valor atual da variável que declaramos anteriormente

A interpolação também aceita expressões e chamada de métodos

```
{{ 1 + 1 }}
```

Métodos

Assim como o **data** podemos declarar nossos métodos através da propriedade **methods** que é um objeto da instância

```
new Vue({
 el: '#app',
 methods: {
   somar(a, b) {
     return a + b
   }
 }
})
```

Dados Computados

Quando precisamos de um **dado processado** para apresentar podemos utilizar os dados computados para processar esses

```
new Vue({
 el: '#app',
 data: {
   valor: 1
 },
 computed: {
   dobroDoValor: function () {
    return this.valor * 2
   }
 }
})
```

Observadores

Quando precisamos realizar uma operação ao alterar algum dado ou precisamos reagir rápido a uma alteração, podemos usar os observadores

```
new Vue({
 el: '#app',
 data: {
   valor: 1
 },
 watch: {
   valor: function (valorNovo, valorAntigo) {
     alert(valorAntigo + ' => ' + valorNovo)
   }
 },
})
```

Diretivas

Diretivas

Diretivas são atributos especiais com o prefixo v-

Espera-se que os valores atribuídos às diretivas sejam uma simples expressão Javascript

O trabalho de uma diretiva é aplicar reativamente efeitos colaterais ao DOM, ou seja, realizar algum efeito quando o valor da expressão é modificado.

```
<p v-if="vocePodeVer">Agora você me viu</p>
```

v-bind

Esta diretiva é usada para alterar algum parâmetro reativamente, isto é, quando você deseja passar uma variável para um elemento html

```
<a v-bind:href="url"> ... </a>
```

Isto fará com que quando a variável 'url' for alterada, o elemento 'a' também será alterado com o novo valor

A diretiva v-bind pode ser usada também com sua forma abreviada que é **:**

```
<a :href="url"> ... </a>
```

v-if

A diretiva v-if é usada para renderizar condicionalmente um bloco

O bloco só será renderizado se a expressão da diretiva retornar um valor verdadeiro. Caso a expressão retorne falso o bloco não existira no DOM

```
<h1 v-if="voceTemAcesso">Você pode ver isso</h1>
```

Aqui caso a variável 'voceTemAcesso' seja verdadeira você verá este elemento.

```
<h1 v-if="voceTemAcesso && ehSextaFeira">Você pode ver isso</h1>
```

v-else

É possível utilizar a diretiva v-else para indicar um 'bloco else' para o v-if

Um elemento v-else deve seguir imediatamente um elemento v-if ou v-else-if, caso contrário não será reconhecido.

```
<h1 v-if="voceTemAcesso">Você pode ver isso</h1>

<h1 v-else>Você não pode ver isso</h1>
```

v-show

Outra opção para mostrar condicionalmente um elemento é a diretiva v-show

A utilização é basicamente a mesma.

A diferença é que um elemento com v-show sempre será renderizado e permanecerá no DOM

v-show simplesmente alterna a propriedade CSS display do elemento

```
<h1 v-show="voceTemAcesso">Você pode ver isso</h1>
```

v-for

Esta diretiva é usada para renderizar uma lista de elementos com base nos dados de um Array

Usamos este formato **item in items** para usar a diretiva, onde **items é o array** e **item é o elemento que está sendo iterado**

```
<ul>
   <li v-for="item in items">
       {{ item.message }}
   </li>
</ul>
```

v-model

Você pode usar a diretiva v-model para criar interligações de mão dupla (two-way binding) entre os dados e elementos input, textarea e select de formulários

A diretiva automaticamente busca a maneira correta de atualizar o elemento com base no tipo de entrada

```
<input v-model="mensagem" placeholder="Me edite">

<p>A mensagem é: {{ mensagem }}</p>
```

v-on

Você pode usar a diretiva v-on para escutar eventos do DOM e rodar algum JavaScript quando tal evento for disparado

```
<button v-on:click="clicou()">Click aqui</button>
```

Quando o botão for clicado o vue irá chamar o método '`clicou`', assim você pode colocar sua lógica neste método

A diretiva v-on pode ser usada também com sua forma abreviada que é **@**

```
<button @click="clicou()">Click aqui</button>
```

Ciclo de Vida

Ciclo de Vida

beforeCreate : executa logo após a inicialização da instância

created: é executado quando a instância e os eventos , as propriedades calculadas , os dados e os métodos são criados. Geralmente é usado para inicializar propriedades do objeto de dados com consultas HTTP

beforeMount : executa imediatamente antes de ser adicionado ao DOM

mounted: é executado após adicioná-lo ao DOM. Pode ser usado para inicializar bibliotecas que dependem do DOM

Ciclo de Vida

beforeUpdate : é executado quando os dados são alterados, mas o DOM ainda não traduziu as alterações

updated: é executado após a alteração dos dados e o DOM mostra essas alterações

beforeDestroy: executa imediatamente antes de excluir a instância

destroyed: é executado quando a instância, eventos, diretivas e filhos do componente foram excluídos

Ciclo de Vida

```
new Vue({
    el: '#app',
    created() {
        console.log('created')
    },
    mounted() {
        console.log('mounted')
    },
    destroyed() {
        console.log('destroyed')
    }
})
```

Interligações de Classe e Estilo

Interligações de Classe e Estilo

Uma necessidade comum de interligação de dados é manipular as classes dos elementos e seus estilos inline através das propriedades **class** e **style**

Podemos manipular isso através do v-bind, só que para este caso temos um tratamento especial

```
<div v-bind:class="{ active: isActive }"></div>

<div class="static"
 :class="{ active: isActive, 'text-danger': hasError }">
</div>
```

Filtros

Filtros

Vue permite que você defina filtros que podem ser utilizados para aplicação de formatações de texto corriqueiras

Filtros são permitidos em interpolações mustache e expressões v-bind

```
<!-- em interpolações de texto -->
{{ mensagem | capitalize }}

<!-- em interligações de atributos -->
<div v-bind:id="id | formataId"></div>
```

Criando um filtro

A criação de um filtro é bem simples

Ele é uma função cujo primeiro parâmetro é sempre o valor a ser aplicado a manipulação

```
filters: {
 capitalize: function (value) {
   if (!value) return ''
   value = value.toString()
   return value.charAt(0).toUpperCase() + value.slice(1)
 }
}
```

Filtro global

Um filtro também pode ser criado diretamente globalmente para que ele esteja disponível em toda a aplicação vue

```
Vue.filter('capitalize', function (value) {
 if (!value) return ''
 value = value.toString()
 return value.charAt(0).toUpperCase() + value.slice(1)
})

new Vue({
 // ...
})
```

Mixins

Mixins

Mixins são uma forma flexível de distribuir funcionalidade reutilizável em diversos componentes Vue

Um objeto mixin pode conter quaisquer opções de componente

Quando um componente utiliza um mixin, todas as opções deste serão misturadas (em inglês, mixed in) com as opções do próprio componente

Seria uma forma de herança de dados e metodos para os componentes

Mixins

Este é nosso mixin

```
var mixin = {
 data: function () {
   return {
     mensagemImportante: 'olá',
   }
 },
 methods: {
   apresentar: function () {
     console.log(this.mensagemImportante)
   }
 }
}
```

Mixins

O componente que usa o mixin, tem todos os dados e métodos do mixin

```
new Vue({
 el: '#app',
 mixins: [mixin],
 methods: {
   conversar: function () {
     this.apresentar()
     console.log(this.mensagemImportante)
   }
 }
})
```

Mixin global

O mixin pode ser registrado globalmente também

```
Vue.mixin({
 methods: {
   conversar: function () {
     console.log('bla bla')
   }
 }
})

new Vue({
 // ...
})
```

Componentes

Componentes

Componentes são instâncias reutilizáveis do Vue

```
Vue.component('button-counter', {
    data: () => ({
        count: 0
    }),
    template: '<button v-on:click="count++">Você clicou em mim {{
count }} vezes.</button>'
})
```

```
<div id="app">
    <button-counter></button-counter>
</div>
```

Componentes

Para usar esses componentes em templates, eles devem ser registrados para que Vue saiba deles

Há dois tipos de registro de componentes, sendo eles: **global** e **local**

```
import Componente from './Componente'

export default {
   components: { Componente }
}
```

Props

Props são atributos personalizáveis que você pode registrar em um componente

Quando um valor é passado para um atributo prop, ele torna-se uma propriedade daquela instância de componente

```
Vue.component('blog-post', {
   props: ['title'],
   template: '<h3>{{ title }}</h3>'
})

<blog-post title="Minha jornada com Vue"></blog-post>
```

Eventos

À medida que desenvolvemos nossos componentes, você se depara com a necessidade de se comunicar de com componente pai

Para isso podemos emitir um evento quando algo ocorrer no componente filho

```
<button v-on:click="$emit('curtir')">
   curtir
</button>

<blog-post title="Minha jornada com Vue" @curtir="facaAlgo()"/>
```

Eventos

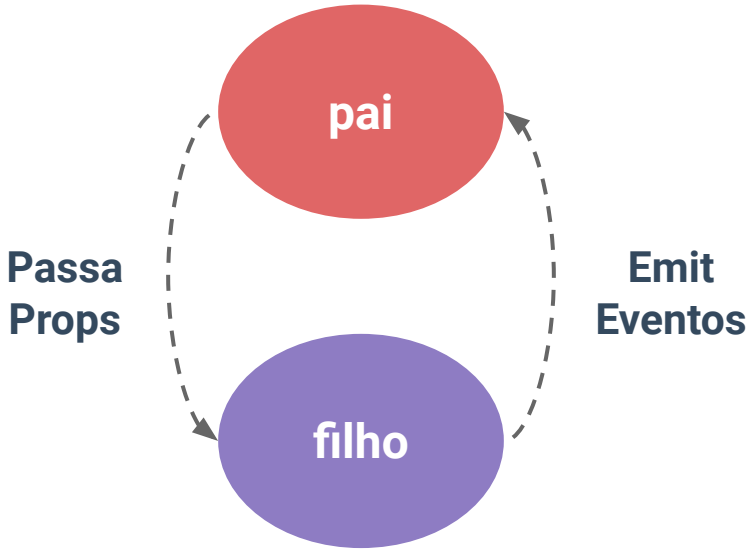
Podemos passar dados no evento, basta adicionar o segundo parâmetro da função $emit

```
<button v-on:click="$emit('curtir', postagem)">
   curtir
</button>

<blog-post title="Minha jornada com Vue" @curtir="facaAlgo"/>
```

Comunicação de componentes

Props entram, Eventos saem



Roteamento

Roteamento

Para fazer o roteamento no vue precisamos do Vue Router que é o roteador padrão do vue.js

Nele podemos configurar nossas rotas e os componentes que representam essas rotas

```
const router = new VueRouter({
    routes: [
        { path: '/home', component: Home },
        { path: '/login', component: Login }
    ]})
new Vue({
    router
}).$mount('#app')
```

Roteamento

No nosso html precisamos apenas dizer onde o conteúdo da rota irá entrar com a tag `<router-view>`

```
<div id="app">
   <h1>Titulo!</h1>
   <p>
       <router-link to="/home">Home</router-link>
       <router-link to="/login">Login</router-link>
   </p>
   <router-view></router-view>
</div>
```

Instalação do Vue.js

Instalação do Vue.js

Podemos utilizar o Vue de duas formas

- Via CDN importando o projeto diretamente no HTML

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

- Instalando via npm

```
npm install vue
```

Instalação via npm

É necessário ter o node.js instalado na máquina

Vue.js oferece um cli oficial para iniciar rapidamente um novo projeto configurado com vue.js

- Instalando o cli vue

```
npm install -g @vue/cli
```

- Utilizando cli

```
vue create <nome-projeto>
```

Vamos começar a ver Vue.js

Começando pelo básico

Vamos começar utilizado vue.js atreves do cdn para ver os conceitos mais basicos do vue

1. Crie um arquivo html

2. Adicione a tag <html> no arquivo e dentro dela as tags <head> e <body>

3. Adicione a importação do vue.js entro do head

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

Criando o projeto com Vue.js

Criando o projeto com Vue.js

Agora que já conhecemos o básico vamos criar um projeto através do Vue Cli, que gera um projeto pronto para começarmos o desenvolvimento no nosso software.

Instale o cli:

```
npm install -g @vue/cli
```

Para criar o projeto utilize o comando abaixo:

```
vue create <nome-projeto>
```

Rodando o Projeto

Para rodarmos o projeto primeiro entraremos na pasta

```
cd <nome-projeto>
```

Rodando o projeto

```
npm run serve
```

Links do GitHub

Exemplos Vue.js
https://github.com/eligabilon/vue_exemplos.git

Projeto Vue.js
https://github.com/JefersonUrbieta/projeto-vue.git

Vue.js com Node.js no backend
https://github.com/JefersonUrbieta/google-keep.git

Obrigado!

vuejs.org @vuejs vuejs/vue

Slides originais: The Progressive Framework & Modern Frontend Development with Vue.js