front-end

prof. Lucas Ferreira

engenharia de software

engenharia de computação

Frameworks de Interação:

Vue.js

Vue.js

Avançando nossa jornada, iremos agora conhecer um pouco de outro framework bem popular do mercado. Após o Vue.js iremos finalizar nossa quest trabalhando com React.js.

- O **Vue** (pronuncia-se /vjuː/, como *view*, em inglês) é um **framework** progressivo para a construção de interfaces de usuário
- Criado (2014) por **Evan You** (ex-funcionário do Google) inspirado no finado AngularJS
- O processo de criação do Vue.js deu-se neste momento: "*Eu imaginei seguinte: e se eu pudesse simplesmente extrair a parte que eu realmente gostei sobre o AngularJS e construir algo realmente leve e funcional?!*"
- Ao contrário de outros frameworks monolíticos, Vue foi projetado desde sua concepção para ser adotável incrementalmente
- A **biblioteca principal** é focada exclusivamente na camada visual (view layer), sendo fácil adotar e integrar com outras bibliotecas ou projetos existentes
- Têm muita "inspiração" (mix) de: **AngularJS**, **React** e o **Angular (atual)**.

Iniciando com Vue.js

Das tecnologias apresentadas até aqui, Vue.js é a de uso inicial ***"mais simples"***. Pode-se trabalhar tanto de maneira avançada como no recomendado para React *(ainda veremos)* ou Angular, mas também pode-se trabalhar de forma simples apenas carregando um arquivo .js externo em seu HTML padrão:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

Depois disso, você poderá trabalhar com ***"micro-inserições"*** de Vue.js em seu projeto já existente ou que tenha menor escala. Para outras opções de instalação confira o link abaixo:

https://vuejs.org/guide/quick-start.html

Criando um projeto com Vue.js

Porém como todo framework front-end de porte industrial temos sempre a opção de criar um projeto 100% baseado nessa tecnologia, um verdadeiro SPA (comentei sobre isso na aula anterior).

No caso do ***Vue.js*** ao rodar seu auxiliar de comando para criar um novo projeto, teremos também o ***Vite.js*** instalado no mesmo para "disponibilizar" um servidor local de trabalho:

```
npm init vue@latest
```

Este comando acima irá lhe fazer algumas perguntas sobre opcionais a serem ativados no projeto, responda o que precisa e uma nova pasta será criada com seus arquivos de trabalho.

Criando um projeto com Vue.js

A nova pasta terá o nome do projeto e num primeiro momento precisamos entrar dentro da mesma e instalar as dependências:

```
cd nome-do-projeto
npm install
```

Depois disso quando precisarmos rodar o projeto em modo desenvolvimento, é exatamente como vimos na nossa primeira experiência com Vite.js:

```
npm run dev
```

Componentes em Vue.js

Nosso projeto criado completamente focado em Vue com os comandos anteriores, já terá uma base instalada e rodando de Vue.js pronta para ser trabalhada. Vale observar que o conceito aqui é mantermos a ideia de "SPA".

O único arquivo .html principal *(index.html)* serve apenas para carregar o script principal e abrigar toda a nossa aplicação dentro de uma div primária, normalmente assim:

```
<div id="app"></div>
<script type="module" src="/src/main.js"></script>
```

Daqui em diante já podemos editar nossos arquivos JavaScript *(ou TypeScript)* e ir criando nossos componentes.

Componentes em Vue.js

Um componente em Vue.js possui uma estrutura mais simplificada do que um de Angular. Normalmente teremos em um único arquivo cujo a extensão exclusiva é ***.vue***, a possibilidade de trabalharmos ao mesmo tempo com JavaScript, HTML e CSS.

```
<template>
  <h1>This is an about page</h1>
</template>

<style>
.about {
  display: flex;
  flex-direction: column;
  align-items: center;
}
</style>

<script>
// lógica aqui...
</script>
```

Componentes em Vue.js

A parte de manuseio de controle de variáveis em Vue.js está mais parecida com o que vamos ver em React, do que o que vemos em Angular. No caso de Angular temos no controller lógico toda a declaração de variáveis e na parte de view apenas interações menores.

Já no caso do Vue.js como temos tudo em um único arquivo .vue podemos trabalhar mais próximo das variáveis que iremos usar em tela. No caso os elementos lógicos ficam dentro da tag ***<script>*** e a parte que é exibida em tela fica dentro do elemento ***<template>***.

Por fim o Vue.js possui dois padrões de interação com variáveis, começaremos vendo o padrão clássico que existe ativo desde o *Vue 1.0* e na sequência nos atualizaremos para o padrão *reativo* atual.

Componentes em Vue.js

No padrão clássico de interação de com variáveis de componentes, iremos na tag ***<script>*** declarar um escopo simples de objeto, sendo que objeto **`data`** nos retorna as variáveis e o objeto **`methods`** nos retorna funções úteis que podem modificar essas variáveis.

```
<template>
  <p>Contagem: {{ count }}</p>
  <button type="button" @click="incrementar">Inc++</button>
</template>

<script>
export default {
  data: () => ({
    count: 0,
  }),
  methods: {
    incrementar() {
      this.count += 1;
    },
  },
}
</script>
```

Componentes em Vue.js

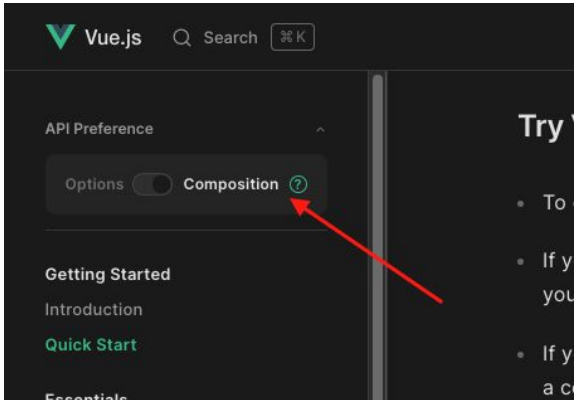
Já no padrão moderno, a busca do Vue.js em sua nova versão é criar padrões de códigos reaproveitáveis e interativos a fim de distribuir melhor o uso de variáveis em diversas partes da aplicação.

Precisaremos inserir o atributo `**setup**` na tag *<script>* e usar a função `**ref**` vinda do "core" do Vue.

```
<template>
  <p>Contagem: {{ count }}</p>
  <button type="button" @click="incrementar">Inc++</button>
</template>

<script setup>
import { ref } from "vue";

const count = ref(0);

function incrementar() {
  count.value++;
}
</script>
```

Componentes em Vue.js

A minha recomendação por quem irá começar agora em Vue.js é que busque dominar o padrão mais moderno e reativo de variáveis e outros elementos. Quando estiverem na documentação do Vue.js *(estudando)* busquem ativar essa chave aqui:



*A chave que alterna entre Options e **Composition** é o que define qual padrão a documentação irá lhe ensinar. As técnicas mais modernas estão no padrão **Composition**.*

Componentes em Vue.js

Sobre Templates em Vue.js

Assim como no Angular, o Vue.js por ser pesadamente inspirado no finado AngularJS também possui uma sintaxe de templates com HTML customizado, diversas coisas "próprias" e que nos são úteis.

Por exemplo, ao invés de `*ngIf`, temos em vue **`v-if`**.

Ao invés de `*ngFor`, temos em vue **`v-for`**.

Para o caso de eventos ao invés de `(click)="..."` temos **`@click="...."`**.

Muita coisa "inspirada" e que funciona bem parecido.

Exemplo de template HTML + Vue.js

```
<template>
  <p>Contagem: {{ count }}</p>
  <button :disabled="count > 5" type="button" @click="incrementar">Inc++</button>
</template>
```

Na template acima ***count*** será uma variável baseada na parte lógica de nosso componente. E o evento de onclick definido através de ***@click="incrementar"*** irá acionar uma função chamada ***incrementar()*** também na camada lógica de nosso componente. Por fim temos um atributo opcional no HTML, o componente button irá se "desativar" quando a contagem for superior a 5.

Vale a pena dar uma olhada na documentação completa de templates:
https://vuejs.org/guide/essentials/template-syntax.html

Templates em Vue.js: *Diretivas Estruturais*

Alguns elementos chave sobre as templates de Vue.js *vem de fábrica* com o projeto, são as **structural directives**.

Estas diretivas estruturais moldam ou reformulam a estrutura do DOM, geralmente adicionando, removendo e manipulando os elementos as quais estão conectadas.

As **structural directives** mais comuns são:

👉 ***v-if*** => condicionalmente adiciona ou remove um elemento do DOM (ou da tela)

```
<div v-if="count > 1">Meu Contador: false</div>
```

👉 ***v-for*** => repete a template para cada item de uma lista

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

v-model

Two-way data binding para campos de formulário

Ao desenvolver formulários, você geralmente exibe em um campo uma variável de dados e atualiza essa propriedade quando o usuário faz alterações.

A ligação de dados bidirecional ***(Two-way data binding)*** com a diretiva ***v-model*** deixa o "caminho" menos difícil:

```
<input v-model"name" />
```

Isso irá atualizar uma variável chamada `**name**` com o mesmo valor digitado no campo em nossa camada lógica.

SO...
LET'S CODE!

Links que valem a pena

Se você quer aprender mais sobre Vue.js apostem nos links abaixo:

- https://vuejs.org/guide/essentials/template-syntax.html
- https://vuejs.org/guide/essentials/reactivity-fundamentals.html
- https://vuejs.org/guide/essentials/class-and-style.html
- https://vuejs.org/guide/essentials/list.html
- https://vuejs.org/guide/components/slots.html
- https://vuejs.org/guide/scaling-up/state-management.html
- https://vuejs.org/guide/scaling-up/routing.html

NOSSO 3º EXERCÍCIO

Adivinhem o que vai ser?

Lá vem o professor chato do **CHAT** novamente...

A missão agora é recriar o nosso **CHAT DE ATENDIMENTO** online só que em **Angular** OU **Vue.js**.

Não vou estar aceitando React nesse exercício *(ainda não)*, quero estimular vocês a mexer com todas as tecnologias possíveis, então o esquema agora OU é **Angular** ou é **Vue.js**.

A entrega é até a próxima aula útil de conteúdo, naquele mesmo esquema darei 1h do começo da aula para vocês terminarem em sala de aula e solicitar minha ajuda se for o caso, ai das 20h em diante retomaremos a aula normalmente.

*P.S: Podem continuar fazendo em **DUPLA** ou **INDIVIDUALMENTE**, entregas com anexo OU link do GITHUB.*

obrigado 🚀