AULA 04 - Vídeo 4

Introdução

Componentes são usados para evitar copia e cola de código, são usados para reutilização de componentes tanto visuais mais comumente no front, mas também visto no back.

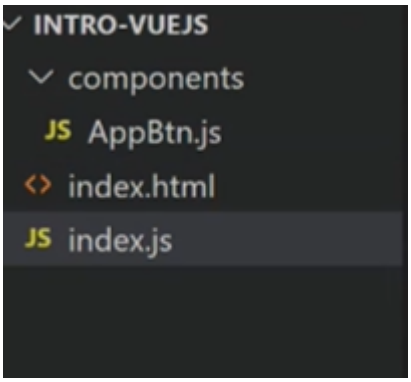
Há 2 maneiras de se fazer isso:
a **Primeira** é utilizando Single File Components(SFC), que usaremos depois com vite
e a **Segunda** com javascript puramente, ha qual usaremos nesta aula.

Segundo a propria documentação do Vue: **https://vuejs.org/guide/essentials/component-basics.html**

“Componentes nos permitem dividir a interface gráfica em pedaços independentes e reutilizá-los"

“Isso é muito semelhante a como aninhamos elementos HTML nativos, mas o Vue implementa seu próprio modelo de componente que nos permite encapsular conteúdo e lógica personalizados em cada componente.”

Usaremos a seguinte estrutura de Pastas:



Criaremos a constante Componente de Botão no ‘App Btn.js’

```
const AppBtn = {
    data: ()=> {
        return {
            titulo: "Ola"
        }
    },
    template: `
    <button>{{titulo}}</button>
    `
}
```

importamos ele no html

```
 <script src="components/AppBtn.js"></script>
```

e no ‘index.js’ registramos esse componente

```
Vue.createApp({
    components: {
        AppBtn: AppBtn
    },
```

ou renomeá lo:

```
 components: {
        "app-btnrenomeado": AppBtn
    },
```

ou usar o padrão:

```
components: {
        AppBtn
    },
```

e utilizamos ele no html

```
<app-btn></app-btn>
```

Código completo até o momento:

html:

```
<body>
    <div id="app">
        <h1>{{title}}</h1>
        <input v-model="nomeDigitado" type="text">
        <button @click="adicionar">Adicionar</button>
        <br>
        <!-- Nome Digitado: {{nomeDigitado}} -->
        <hr>
        <section>
            <div v-for="(nome, pos) in clientes" :key="nome">
                <span>{{nome}}</span>
                <button @click="remover(pos)">Remover</button>
            </div>
        </section>
        <app-btn></app-btn>
    </div>

                                                        <script
src="https://cdnjs.cloudflare.com/ajax/libs/vue/3.2.33/vue.global.min.j
s"></script>
    <script src="components/AppBtn.js"></script>
    <script src="index.js"></script>

</body>
```

index.js:

```
Vue.createApp({
    components: {
        AppBtn
    },
    data: () => {
        return {
            title: "Bem vindo ao VueJS",
            nomeDigitado: '',
            clientes: [
                'joao',
                'maria',
                'ze',
                'pafuncio'
```

```
            ]
        }
    },
   methods: {
    adicionar(){
        this.clientes.push(this.nomeDigitado);
        this.nomeDigitado = "";
    },
    remover(pos){
        //alert("rem " +pos)
        this.clientes.splice(pos, 1);
    }
   }
}).mount("#app")
```

AppBtn.js

```
const AppBtn = {
    data: ()=> {
        return {
            titulo: "Ola"
        }
    },
    template: `
    <button>{{titulo}}</button>
    `
}
```

"Não estamos usando da forma convencional, estamos com javascript normal, então possui algumas diferenças no '.vue' , com arquivos .vue e 'sfc', mas chegará ao ponto a qual geramos a aplicação via terminal com Vite (Built-in tool)."

Podemos utilizar nosso componente sem muito esforço:

```
        <app-btn></app-btn>
        <app-btn></app-btn>
        <app-btn></app-btn>
        <app-btn></app-btn>
```

Podemos também utilizar uma estilização padrao no nosso botão ao qual se repetirá.

```
 template: `
      <button  style="background-color:blue;  color:white;  border:none;
padding: 8px 10px;border-radius:5px;">
        {{titulo}}
    </button>
    `
```

Depois veremos como é feita a comunicacao pai e filho com props
e comunicação filho pai com event emitter.

Devemos Tomar cuidado com nomes de componente, não devemos utilizar nomes de tags Html em componentes
o comum é utilizar uma convenção de prefixo app, como "App Btn", ‘seria o botão customizado desse app’