**DIRETIVAS**

**v-show** ( destrói ou recria blocos)

**v-if | v-else** ( destrói ou recria blocos)

**v-bind** (adicionar imagens ou documentos)

**v-for** (mostrar listas ou itens da lista)

**v-model** ( mostra na tela ao mesmo tempo, usado em formulários )

**v-on** ( métodos ( eventos ))

**------------------------------------------------------------------------------------------------------------------------------**

**v-show**

Usamos a propriedade v-show, para destruir ou recriar um bloco, durante a alternância, precisamos criar uma variável para atribuir um resultado para o v-show, caso a variável tenha o valor (true) o elemento será mostrado na tela, se o valor for (false), o v-show vai **ocultar da tela e do console**.

EX:

<body>

    <div id="app"> <!--INICIO DO APP-->

        <h1 v-show="statustitulo"> {{titulo}} </h1>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/vue.js"></script>

    <script>

        var app = new Vue({

            el: "#app",

            data: {

                statustitulo: true, //valor para atribuir ao v-show

                titulo: 'Mostrando o v-show', // variável do titulo

            }

        })

    </script>

</body>

**v-if | v-else**

Usamos o v-if e o v-else, juntos, para mostrar ou ocultar um bloco determinado pela condição, v-if ( **se** a condição for verdadeira) vai mostrar o bloco do v-if, **se não**, vai mostrar o bloco do v-else.

O v-else oculta somente da tela, no console o bloco continua visível, também é possível colocar outros conteúdos para mostrar no bloco do v-else.

<body>

    <div id="app">

        <h1 v-if="statustitulo">{{ titulo }}</h1> <!—bloco v-if (true)-->

        <div v-else> Não existe titulo </div> <!—bloco v-else (false)-->

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/vue.js">

</script>

    <script>

        var app = new Vue({

            el: "#app",

            data: {

statustitulo: true, //true, executa o titulo na tela |false, ele oculta o titulo

            titulo: "Aula 01 - Vue 2",

            }})

    </script>

</body>

**v-bind**

Usamos para adicionar documentos no html, como imagens.

Criamos uma variável em (data), com a URL de uma imagem, ou do documento desejado, no HTML adicionamos uma tag imagem, com o v-bind recebendo o endereço (src) que está recebendo o nome da variável com o valor da URL.

<body>

    <div id="app">

        <!--atribuimos ao v-bind o elemento src, o src recebe a variavel que está recebendo a URL da imagem-->

        <img v-bind:src="imagem" alt="">

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/vue.js"></script>

    <script>

        var app = new Vue({

            el: "#app",

            data: {

                imagem: "https://blog.geekhunter.com.br/wp-content/ uploads/2019/12/vue-js.png"<!—URL da imagem-->

            }})

    </script>

</body>

**v-for**

Geralmente usado para listas, criamos um objeto no (data) e criamos vários elementos (itens), com o v-for é possível mostrar a lista ou algum item especifico no HTML.

<body>

    <div id="app">

        <ul>

<!—v-for recebe item do objeto linguagens -->

            <li v-for="item in linguagens">{{item.nome}}</li>

        </ul>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/ vue.js"> </script>

    <script>

        var app = new Vue({

            el: "#app",

            data: {

                titulo: "Lista de Usuario",

OBJETO -->linguagens: [

                    {nome: 'JavaScript'},

                    {nome:'PHP'},

          ITENS-->  {nome:'C#'},

                    {nome: 'Java'},

                    {nome: 'Python'},

                ]

            }})

    </script>

</body>

**v-model**

É muito usado para formulários, dentro do input, texarea, checkbox, radio button, entre outros. O v-model atualiza automaticamente o campo, mostra em tempo real as modificações na tela.

Sempre precisamos atribuir o value nas opções dos campos, abaixo á opções dos campos mais usados em formulários.

<body>

    <div id="app">

        <label for=""> Nome </label>

        <input type="text" name="nome" id="nome" v-model="nome">

        <br>

        <label for="">Telefone:</label>

        <input type="text" name="tel" id="tel" v-model="telefone">

        <br>

        <label for="">Deseja receber nossas novidades:</label><br>

        <input type="radio" value="Sim" v-model="novidades">Sim

        <input type="radio" value="Não" v-model="novidades">Não

        <br>

        <label for="">Interesses</label>

  <input type="checkbox" value="Futebol" v-model="interesses"> Futebol

        <input type="checkbox" value="Volei" v-model="interesses">Volei

      <input type="checkbox" value="Corrida" v-model="interesses"> Corrida

      <input type="checkbox"value="Formula 1"v-model="interesses"> Formula 1

        <br>

        <label for="">Como nos conheceu?</label>

        <select  v-model="conheceu">

            <option value="">Escolha</option>

            <option value="Google">Google</option>

            <option value="TV">TV</option>

            <option value="Revista">Revista</option>

        </select>

        <button type="submit">Enviar</button>

        <!--IMPRIMINDO AS VARIAVEIS NA TELA -->

        Nome: {{nome}} <br>

        Telefone: {{telefone}} <br>

        Novidades: {{novidades}} <br>

        Interesses: {{interesses}} <br>

        Como nos Conheceu? {{conheceu}} <br>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/vue.js"></script>

    <script>

        var app = new Vue({

            el: "#app",

            data: {

                nome:" ",

                telefone:" ",

                novidades: " ",

                interesses:[ ],

                conheceu:" "

            }})

    </script>

</body>

**v-on**

Para usar os métodos do vue.js, precisamos de uma área para criar todos os metodos, essa área deve se chamar **(methods)** dentro dela criamos os métodos, que são como “funções”, que tem um bloco com os comandos a serem executados, quando certo evento chamar o metodo.

//METODOS - PARTE LOGICA

                methods: {

          método -> enviar()

{ bloco de comandos a serem executados

                        alert("enviado")    }

                }})

Para chamar os métodos, precisamos que aconteça um evento, esses eventos podem ser clique de mouse, apertar uma tecla, entre outras formas.

Colocamos no HTML uma tag que seja possível atribuir um evento á ela, como um botão, nessa tag (botão), adicionamos o **v-on: (mais a ação desejada).** Ex: v-on:click ( evento de clique ), esse evento vai receber o método que foi criado.

**Exemplo:**

Evento clique recebe o método enviar

<input id="enviar" type="button" value="Enviar" v-on:click="enviar">

Ao clicar, tudo que estiver dentro do bloco do método enviar, será executado.

OBS: É possível encurtar a sintaxe, usando o **@** podemos excluir o v-on:.

Ex: **@click**

<input id="enviar" type="button" value="Enviar" @click="enviar">

**Exemplo Completo:**

<body>

    <div id="app"> <!--COMEÇO DO APP-->

quando usuário clicar, execute enviar

       <input id="enviar" type="button" value="Enviar" v-on:click="enviar">

    </div><!--FIM DO APP-->

        <script src="https://cdn.jsdelivr.net/npm/vue@2.7.13/dist/vue.js"></script>

        <script>

            var app = new Vue({

                el: "#app",

                data: {

                    //VARIAVEIS

                },

                //METODOS - PARTE LOGICA

                methods: {

                   enviar() {

                        alert("enviado")

                   }}})

        </script>

</body>