Diretivas

v-bind - Liga dinamicamente atributos ou propriedades de um elemento a uma expressão do vue, ou seja, transforma o valor do atributo em vue. Pode ser representado de forma abreviada como dois pontos:
`v-bind:href=”variavel”` ou `:href=”funcao”`

v-cloak - Utilizando em conjunto com o CSS ele esconde o conteúdo até o vue terminar de compilar o template, ou seja, tudo que esta dentro do elemento que possui o v-cloak só será exibido depois que o vue terminar de compilar:
No Vue:

```
<div v-cloak>
    Só apareço depois que o vue compilar!
</div>
```

no CSS:

```
[v-cloak] {
   display: none;
}
```

v-for - É usada para renderizar elementos com base em uma lista ou objeto, ou seja, a partir de uma lista ou objeto vai repetir o elemento em que ele esta. É obrigatório usar um key no elemento que esta o for:

```
<li v-for=”(item, index) in items” :key=”index”>
   {{ index }} - {{ item }}
</li>
```

O <li> repetirá a partir da quantidade de itens que tem na lista items.

v-if - Serve para renderizar ou não um elemento no DOM com base em uma condição, se for verdadeira aparece e se for falsa o elemento não é nem adicionado no DOM:

`<p v-if="isLoggedIn">Bem-vindo!</p>`

v-else - É usada em conjunto com o v-if ou v-else-if para renderizar um bloco alternativo, caso todas as condições anteriores sejam falsas, o elemento com o v-else precisa estar imediatamente após um v-if ou v-else-if, sem nenhum outro elemento entre eles, alem disso, não aceita expressão:
`<p v-else>Faça login!<p>`

v-else-if – Permite criar condições intermediarias entre um v-if e um v-else, funciona como em estruturas tradicionais do javascript. Vai renderizar o primeiro bloco que for verdadeiro os seguintes são ignorados assim que uma condição é satisfeita:
`<p v-else-if="condição"></p>`

v-model – Cria uma ligação entre uma variavel e o valor inserido em um input (text, checkbox, radio, etc.), fazendo com que a variavel receba o valor inserido no input em tempo real, ou seja, tudo que é inserido no input é passado pra variavel:
`<input type="text" v-model="variavel" />`
`<p>{{ variavel }}<p>`

Modificadores:
`<input v-model.lazy="" />` - Atualiza apenas quando o input perde o foco.
`<input v-model.number="" />` - Converte a entrada para numero.
`<input v-model.trim="" />` - Remove espaços em branco no inicio e fim.

v-on - Serve para ouvir eventos do DOM (como cliques, teclas, envio de formulário, etc.). Pode ser representado de forma abreviada como um @. É basicamente a forma de atrelar eventos no vue:

@click: Clique do mouse

@submit: Envio de formulario

@change: Mudança de valor

@input: Entrada de texto

`<button @click=”funcao”><button>`

Modificadores:
`<form @submit.prevent=”funcao”></form>` - Chama event.preventDefault( ).
`<button @click.stop=”funcao”></button>` - Chama event.stopPropagation( ).
`<button @click.once=”funcao”></button>` - Executa apenas uma vez.
`<button @click.capture=”funcao”></button>` - Usa captura em vez de propagação normal.
`<button @click.self=”funcao”></button>` - Executa apenas se o evento for no próprio elemento
`<button @click.passive=”funcao”></button>` - Marca o listener como passive, útil para melhorar performance em eventos de rolagem/touch.

v-once – Faz com que um elemento ou componente seja renderizado apenas uma vez, durante a primeira renderização, serve para otimizar performance. Ele não impede que o conteudo apareça, apenas impede que mude dinamicamente, afetando apenas o vue e não o HTML:
`<p v-once>{{ message }}</p>`

Mesmo que a variavel message seja alterada depois, o conteúdo não será re-renderizado.

v-pre – Diz ao vue para ignorar a compilação daquele trecho do template, ou seja, o vue não avalia diretivas, interpolações ({{}}) ou bindings dentro do elemento com v-pre, serve para melhorar a performance pulando a compilação de partes estáticas ou mostrar o código “cru”:
`<p v-pre>{{ isso_não_será_renderizado }}</p>`

Combina com o elemento HTML <pre> que funciona da mesma forma, permite exibir o texto exatamente como foi escrito:
`<pre v-pre></pre>`

v-show – Serve para mostrar ou esconder um elemento no DOM com base em uma condição, sem removê-lo do DOM, ou seja, sempre esta no DOM, só esconde via CSS, trocando o display:
`<p v-show=”isVisible”></p>`

v-slot – É usada para definir slots em componentes, permitindo passar conteúdo personalizado para dentro de componentes filhos. Permite que o componente pai controle o conteúdo de certas áreas do filho, mantendo a logica e estrutura centralizadas no componente filho:

Componente Filho

```
<template>
  <div class="card">
    <header>
      <slot name="header"></slot>
    </header>

    <main>
      <slot></slot> <!-- slot padrão -->
    </main>

    <footer>
      <slot name="footer"></slot>
    </footer>
  </div>
</template>
```

Componente Pai

```
<Card>
  <template v-slot:header>
    <h1>Título do Card</h1>
  </template>
```

```
  <p>Esse é o conteúdo principal.</p> <!-- Slot padrão -->

  <template v-slot:footer>
    <small>Rodapé aqui</small>
  </template>
</Card>
```

Basicamente, o <h1> vai dentro do <header>, o <small> vai dentro do <footer> e o <p> vai dentro do <main>.

v-text – Serve para inserir texto diretamente no conteudo de um elemento HTML, substituindo qualquer conteúdo anterior. Pega o valor de uma variavel e insere como texto puro no elemento, não interpreta HTML:
`<p v-text=”variavel”></p>`

v-html – Serve para inserir conteúdo HTML bruto (não como texto, mas como elementos reais) dentro de um elemento, é como usar innerHTML:
`<div v-html=”<p>texto<p>”></div>`

Diretiva Personalizada – Diretivas criadas pelo usuario, para aplicar comportamentos específicos a elementos do DOM. Serve para manipular o DOM diretamente com lógica personalizada:

Exemplo de Diretiva v-focus:
em um arquivo ou direto no main.js

```
app.directive(‘focus’, {
    mounted(el) {
        el.focus()
    }
})
```

`<input type=”text” v-focus />`

É possível usar os seguintes hooks no objeto da diretiva:

created: Quando a diretiva é registrada

beforeMount: Antes do elemento ser inserido no DOM

mounted: Quando o elemento foi inserido no DOM

beforeUpdate: Antes da atualização do DOM

updated: Depois da atualização

beforeUnmount: Antes de remover do DOM

unMounted: Após a remoção do DOM