**SESSÃO 2**

**AULA 18 – Binding de Atributos**

1. Para passarmos valores para as propriedades das tags html temos que utilizar diretivas, uma delas é a v-bind que irá resolver a variável link dento do href da tag <a>
2. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
3. <div *id*="app">
4. <p>{{titulo}}</p>
5. <p>{{saudacao()}}</p>
6. <a *v-bind:href*="link">Google</a>
7. </div>
8. <script>
9. **new** Vue({
10. el: '#app',
11. data: {
12. titulo: 'Usando VueJs 2!',
13. link: 'http://google.com.br'
14. },
15. methods: {
16. saudacao: function(){
17. return  *this*.titulo;
18. }
19. }
20. })
21. </script>
22. C

**Aula 20 – Evitando re-renderização**

1. Para evitar a re-redenrização de uma variável, temos que utilizar a diretiva v-once para que aquele valor somente seja carregado uma vez e não seja alterado.
2. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
3. <div *id*="app">
4. <p *v-once*>{{titulo}}</p>
5. <p>{{saudacao()}}</p>
6. <a *v-bind:href*="link">Google</a>
7. </div>
8. <script>
9. **new** Vue({
10. el: '#app',
11. data: {
12. titulo: 'Usando VueJs 2!',
13. link: 'http://google.com.br'
14. },
15. methods: {
16. saudacao: function(){
17. *this*.titulo = 'Bom dia!!!'
18. return  *this*.titulo;
19. }
20. }
21. })
22. </script>

**Aula 21 – Imprimir HTML puro**

1. Em algumas situações temos que imprimir html puro através do Javascript então temos que fazer uma interpolação. Para isso utilizamos a diretiva v-html=’variavel do data()’
2. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
3. <div *id*="app">
4. <p *v-once*>{{titulo}}</p>
5. <p>{{saudacao()}}</p>
6. <a *v-bind:href*="link">Google</a>
7. <hr>
8. <p *v-html*='linkHtml'></p>
9. </div>
10. <script>
11. **new** Vue({
12. el: '#app',
13. data: {
14. titulo: 'Usando VueJs 2!',
15. link: 'http://google.com.br',
16. linkHtml:'<a href="http://google.com.br">google</a>'
17. },
18. methods: {
19. saudacao: function(){
20. *this*.titulo = 'Bom dia!!!'
21. return  *this*.titulo;
22. }
23. }
24. })
25. </script>
26. DESAFIO

**Aula 24 – Eventos – diretiva v-on:click**

1. Podemos utilizar a diretiva v-on para chamar uma função, dentro do elemento html utilizamos v-on:’tipo de evento’ onde o tipo do evento seria por exemplo click.
2. No exemplo abaixo temos uma variável ‘contador’ que é mostrada dentro de um parágrafo. Temos também um botão que utiliza a diretiva v-on:click que irá executar o método somar.
3. Toda vez que clicarmos no botão o contador irá incrementar.
4. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
5. <div *id*="app">
6. <p>{{contador}}</p>
7. <button *v-on:click*="somar">Somar 1</button>
8. </div>
9. <script>
10. **new** Vue({
11. el: '#app',
12. data: {
13. contador: 0
14. },
15. methods: {
16. somar() {
17. return *this*.contador++
18. }
19. }
20. })
21. </script>

**Aula 25 – Eventos – diretiva v-on:mousemove**

1. Neste exemplo iremos mostrar como capturar a posição do mouse, faremos isso utilizando a diretiva v-on:mousemove. Utilizaremos o exemplo anterior onde iremos acrecescentar um parágrafo ao template e nesse parágrafo iremos incluir x e y que serão as variáveis da posição do mouse.
2. Dentro do data() iremos incluir então as variáveis x e y com valor iniciamente de zero.
3. Iremos acrescentar outro método chamado atualizarXY que irá receber a posição do mouse através do evento clientX e clientY e irá passa-los para as variáveis x e y.
4. Para vermos os eventos que estão dentro de mousemove, podemos utilizar o console.log(event) para mostrar os eventos que estão dentro de mousemove.
5. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
6. <div *id*="app">
7. <p>{{contador}}</p>
8. <button *v-on:click*="somar">Somar 1</button>
9. <p *v-on:mousemove*="atualizarXY">Mouse: {{x}} e {{y}}</p>
10. </div>
11. <script>
12. **new** Vue({
13. el: '#app',
14. data: {
15. contador: 0,
16. x: 0,
17. y: 0
18. },
19. methods: {
20. somar() {
21. return *this*.contador++
22. },
23. atualizarXY(*event*){
24. //console.log(event);
25. *this*.x = *event*.clientX,
26. *this*.y = *event*.clientY
27. }
28. }
29. })
30. </script>

**Aula 25 – Eventos – Passando seus próprios parâmetros e eventos**

1. Utilizando o exemplo anterior, iremos incluir no template na diretiva v-on:click=somar, um parâmetro que neste caso será o número 5. E iremos alterar o método soma passando para ele um parâmetro que iremos chamar de passo. Toda vez que clicarmos no botão o contador será acrescido do passo que é 5.
2. Podemos também passar eventos como parâmetros, mas para fazer isso temos que utilizar no template a expressão chave ‘$event’, essa expressão não pode ser diferente caso contrário ocorrerá um erro.
3. Dentro do método passamos então o outro parâmetro que neste caso será o evento que está sendo passado pelo template.
4. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
5. <div *id*="app">
6. <p>{{contador}}</p>
7. <button *v-on:click*="somar(5, $event)">Somar 1</button>
8. <p *v-on:mousemove*="atualizarXY">Mouse: {{x}} e {{y}}</p>
9. </div>
10. <script>
11. **new** Vue({
12. el: '#app',
13. data: {
14. contador: 0,
15. x: 0,
16. y: 0
17. },
18. methods: {
19. somar(*passo*, *evento*) {
20. console.log(*this*.contador, *evento*);
21. return *this*.contador+=*passo*
22. },
23. atualizarXY(*event*){
24. //console.log(event);
25. *this*.x = *event*.clientX,
26. *this*.y = *event*.clientY
27. }
28. }
29. })
30. </script>