**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 26 – Eventos – Passando seus próprios argumentos 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>

**Aula 27 – Modificadores de eventos**

1. Iremos utilizar o exemplo anterior onde iremos acrecentar um span ao final do parágrafo que possui a diretiva v-on com o evento mousemove recebendo o método atualizarXY. Ao movermos o mouse sobre o parágrafo, é passado a posição X e Y para as variáveis X e Y, então iremos limitar o posicionamento somente até o span.
2. Iremos utilizar então dentro do span a diretiva v-on novamente com o evento mousemove passando para ele o método parar.
3. Nesse método iremos receber o evento e iremos utilizar a função stopPropagation().
4. Ou podemos simplesmente utilizar o modificador de evento .stop no evento mousemove, ficando mousemove.stop e desta forma o evento do mouse será parado dentro do span.
5. Outro tipo de modificador de evento é o .stop.prevent que irá fazer com que o comportamento padrão não seja executado. Para exemplificar criamos uma âncora que terá a diretiva v-on:click e terá o link para o Google. Ao utilizarmos o .stop.prevent o link não irá funcionar.
6. Temos também o modificador de evento preventDefault() que será utilizado dentro de método chamado naoNavegar() esse método irá receber o evento do click porém será passado para esse evento a função .preventDefault() que irá interromper a execução padrão.
7. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
8. <div *id*="app">
9. <p>{{contador}}</p>
10. <button *v-on:click*="somar(5, $event)">Somar 1</button>
11. <p *v-on:mousemove*="atualizarXY">
12. Mouse: {{x}} e {{y}}
13. <span *v-on:mousemove*="parar">Parar Aqui!!!</span>
14. <span *v-on:mousemove.stop.prevent*>Parar Aqui!!!</span>
15. </p>
16. <a *v-on:click.prevent* *href*="http://google.com">Google</a>
17. <br>
18. <a *v-on:click*="naonavegar" *href*="http://google.com">Google</a>
19. </div>
20. <script>
21. **new** Vue({
22. el: '#app',
23. data: {
24. contador: 0,
25. x: 0,
26. y: 0
27. },
28. methods: {
29. somar(*passo*, *evento*) {
30. console.log(*this*.contador, *evento*);
31. return *this*.contador+=*passo*
32. },
33. atualizarXY(*event*){
34. //console.log(event);
35. *this*.x = *event*.clientX,
36. *this*.y = *event*.clientY
37. },
38. parar(*e*){
39. *e*.stopPropagation();
40. },
41. naonavegar(*e*){
42. *e*.preventDefault();
43. }
44. }
45. })
46. </script>

**Aula 28 – Modificadores de evento no teclado**

1. Vamos copiar o exemplo anterior e iremos criar um input do tipo texto, nesse input iremos utilizar a diretiva v-on:keyup e passaremos para ele um método chamado exibiralerta.
2. Nesse método iremos simplesmente exibir um alerta com a mensagem exibir alerta. Fazendo desta forma toda vez que digitarmos uma letra dentro do input e soltarmos a tecla será exibido um popup com um alerta e a mensagem exibir alerta.
3. Podemos modificar esse evento por exemplo acrescentando o .enter na frente de keyup e faremos com que o método exibirAlerta seja disparado somente quando apertarmos a tecla enter.
4. Podemos também fazer um encadeamento acrescentando em frente ao enter por exemplo .alt, desta forma o método somente será chamado quando for apertado as teclas alt+enter. Isso facilita a execução de métodos que precisam serer executados somente quando uma tecla ou várias teclas forem pressionadas.
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(5, $event)">Somar 1</button>
9. <p *v-on:mousemove*="atualizarXY">
10. Mouse: {{x}} e {{y}}
11. <span *v-on:mousemove*="parar">Parar Aqui!!!</span>
12. <span *v-on:mousemove.stop.prevent*>Parar Aqui!!!</span>
13. </p>
14. <a *v-on:click.prevent* *href*="http://google.com">Google</a>
15. <br>
16. <a *v-on:click*="naonavegar" *href*="http://google.com">Google</a>
17. <hr>
18. <input *type*="text" *v-on:keyup.enter.alt*="exibirAlerta">
19. </div>
20. <script>
21. **new** Vue({
22. el: '#app',
23. data: {
24. contador: 0,
25. x: 0,
26. y: 0
27. },
28. methods: {
29. somar(*passo*, *evento*) {
30. console.log(*this*.contador, *evento*);
31. return *this*.contador+=*passo*
32. },
33. atualizarXY(*event*){
34. //console.log(event);
35. *this*.x = *event*.clientX,
36. *this*.y = *event*.clientY
37. },
38. parar(*e*){
39. *e*.stopPropagation();
40. },
41. naonavegar(*e*){
42. *e*.preventDefault();
43. },
44. exibirAlerta(*e*){
45. alert('Exibir alerta');
47. }
48. }
49. })
50. </script>
51. **Desafio 02**

**Aula 32 – Código javaScript no template**

1. Vamos ver mais alguns detalhes sobre a interpolação, para isso iremos copiar o exemplo 09 e renomear para 10\_interpolação.
2. No parágrafo iremos multiplicar a variável contador por 2 e veremos esse resultado ser aplicado na tela.
3. Vamos copiar o parágrafo anterior e vamos incluir dentro das chaves duplas uma expressão ternária para que seja exibido uma mensagem levando em conta o valor do contador.
4. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
5. <div *id*="app">
6. <p>{{contador \* 2}}</p>
7. <p>{{contador > 10 ? 'Maior que 10' : 'Menor que 10'}}</p>
8. <button *v-on:click*="somar">Somar 1</button>
9. </div>
10. <script>
11. **new** Vue({
12. el: '#app',
13. data: {
14. contador: 0
15. },
16. methods: {
17. somar() {
18. *this*.contador++;
19. }
20. }
21. })
22. </script>

**Aula 32 – Two-Way-Binding**

1. Iremos utilizar o exemplo anterior porém iremos formata-lo para que ele fique somente com um parágrafo dentro do template e neste parágrafo terá um doble mustache com a variável titulo.
2. Teremos dentro da propriedade data do Vue, a variável titulo como o texto ‘Usando o Vue Js’.
3. Iremos criar um input do tipo texto que terá a diretiva v-bind:value onde passaremos para ela a variável ‘titulo’ dessa forma o valor da variável ‘titulo’ será populada dentro do input quando a tela for renderizada. Isso é um bind de uma única via ou seja o dado somente vai para o template.
4. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
5. <div *id*="app">
6. <p>{{titulo}}</p>
7. <input *type*="text" *v-bind:value*="titulo">
8. </div>
9. <script>
10. **new** Vue({
11. el: '#app',
12. data: {
13. titulo: 'Usando o Vue JS!'
14. }
15. })
16. </script>
17. Se quisermos alterar a variável input no backend temos que utilizar a diretiva v-model passando para ela a variável ‘titulo’, dessa forma ao alterarmos o valor no input a alteração será realizada também no backend, ou seja será realizada a alteração nas duas direções.
18. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
19. <div *id*="app">
20. <p>{{titulo}}</p>
21. <input *type*="text" *v-model*="titulo">
22. </div>
23. <script>
24. **new** Vue({
25. el: '#app',
26. data: {
27. titulo: 'Usando o Vue JS!'
28. }
29. })
30. </script>