**Introdução aos Componentes**

**Aula 105 – Revisão sobre componentes**

1. Vamos criar uma pasta chamada componentes e dentro dela um arquivo chamado revisão.html.
2. Vamos usar a forma antiga referenciando o vue diretamente no arquivo e depois iremos utilizar o vue-create ainda neste capítulo.
3. <meta *charset*="UTF-8">
4. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
5. <div *id*="app">
6. <span>{{contador}}</span>
7. <button *@click*="contador++">+</button>
8. <button *@click*="contador--">-</button>
9. </div>
10. <script>
11. **new** Vue({
12. el:'#app',
13. data: {
14. contador: 0
15. }
16. })
17. </script>
18. Desta forma não é possível duplicar as div’s, temos que criar compontentes.
19. Podemos criar utilizando a forma glogal. Ficando assim:
20. <meta *charset*="UTF-8">
21. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
22. <div *id*="app">
23. <contador></contador>
24. </div>
25. <script>
26. Vue.component('contador',{
27. template:`
28. <div>
29. <span>{{contador}}</span>
30. <button @click="contador++">+</button>
31. <button @click="contador--">-</button>
32. </div>
33. `,
34. data() {
35. return{
36. contador: 0
37. }
38. }
39. })
40. **new** Vue({
41. el:'#app',
43. })
44. </script>
45. Agora para utilizarmos mais de um componente basta inseri-los dentro da div app.
46. <div *id*="app">
47. <contador></contador>
48. <contador></contador>
49. <contador></contador>
50. </div>

**Aula 106 -Registrando componentes (Local e global)**

1. Vamos mostrar outra forma de registrar um componente, para isso vamos criar o arquivo compontente-registro.html.
2. Vamos duplicar a div app, criando uma div com app1 e outra app2, vamos também duplicar a instância vue apontando para app1 e outra para app2.
3. <meta *charset*="UTF-8">
4. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
5. <div *id*="app1">
6. <contador></contador>
7. <contador></contador>
8. <contador></contador>
9. </div>
10. <hr>
11. <div *id*="app2">
12. <contador></contador>
13. <contador></contador>
14. <contador></contador>
15. </div>
16. <script>
17. Vue.component('contador',{
18. template:`
19. <div>
20. <span>{{contador}}</span>
21. <button @click="contador++">+</button>
22. <button @click="contador--">-</button>
23. </div>
24. `,
25. data() {
26. return{
27. contador: 0
28. }
29. }
30. })
31. **new** Vue({
32. el:'#app1',
34. })
35. **new** Vue({
36. el:'#app2',
38. })
39. </script>
40. Isso só é possível devido termos criado o componente de forma global.

**Criando componente Local**

1. Vamos utilizar forma local de criação de componente, para isso vamos apagar o vue.componente e criar uma constante do tipo objeto chamada contador e vamos passar todo o conteúdo do componente.
2. Agora dentro da instância do vue app1 vamos criar um atributo chamado components, dentro de components vamos criar uma variável chamada contador1 que irá receber a variável contador. Agora a primeira instância app1 irá funcionar.
3. Vamos repetir o mesmo procedimento na instância app2 porém com o component chamado contador2.
4. <meta *charset*="UTF-8">
5. <script *src*="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
6. <div *id*="app1">
7. <contador1></contador1>
8. <contador1></contador1>
9. <contador1></contador1>
10. </div>
11. <hr>
12. <div *id*="app2">
13. <contador2></contador2>
14. <contador2></contador2>
15. <contador2></contador2>
16. </div>
17. <script>
18. const contador = {
19. template:`
20. <div>
21. <span>{{contador}}</span>
22. <button @click="contador++">+</button>
23. <button @click="contador--">-</button>
24. </div>
25. `,
26. data() {
27. return{
28. contador: 0
29. }
30. }
31. }
33. **new** Vue({
34. el:'#app1',
35. components:{
36. contador1: contador
37. }
38. })
39. **new** Vue({
40. el:'#app2',
41. components:{
42. contador2: contador
43. }
45. })
46. </script>

**Aula 107 – Criando projeto usando Vue Cli**

1. Vamos criar um projeto chamado componente-intro e armazená-lo dentro da pasta componentes. Vamos utilizar o modelo modelo padrão do vue 2.0.
2. Após criado o projeto iremos realizar algumas alterações.
3. Vamos apagar a pasta assets, a pasta componente e vamos limpar o conteúdo do componente app.
4. No componente App vamos criar uma extrutura de componente.
5. <template>
7. </template>
8. <script>
9. export default {
11. }
12. </script>
13. <style>
14. </style>
15. Vamos criar um H1 com uma interpolação de uma variável chamada ‘contador’.
16. Para isso vamos criar dentro de data: a variável contador.
17. <template>
18. <div *id*="app">
19. <h1>Contadores</h1>
20. <h1>{{contador}}</h1>
21. <button *onclick*="contador ++">+</button>
22. </div>
24. </template>
25. <script>
26. export default {
27. name: 'App',
28. data() {
29. return {
30. contador: 0
31. }
32. },
33. }
34. </script>
35. <style>
36. </style>
37. **Aula 108 – Criando um componente**
38. Vamos criar um novo componente e utilizar dentro do componente App.
39. O nome será contador.vue com a estrutura simples de um componente.
40. Dentro do campo de script vamos criar a variável contador em data e também dois métodos que será um de adicionar e outro de subtrair.
41. Estes métodos irão incrementar e decrementar a variável contador que está no campo data.
42. No template vamos adicionar uma div com a classe chamada ‘contador’para aplicarmos estilos.
43. Dentro da div vamos adicionar um span que terá a interpolação da variável ‘contador’.
44. Teremos também dois botões que irão chamar os métodos adicionar e subtrair, neste botões teremos a diretiva v-on que utilizará o evento do click do botão.
45. <template>
46. <div *class*="contador">
47. <span>{{contador}}</span>
48. <button *v-on*:*click*="adicionar">+</button>
49. <button @*click*="subtrair">-</button>
50. </div>
52. </template>
53. <script>
54. export default {
55. data() {
56. return {
57. contador: 0
58. }
59. },
60. methods: {
61. adicionar(){
62. *this*.contador++
63. },
64. subtrair(){
65. *this*.contador--
66. }
67. },
69. }
70. </script>
71. <style>
72. *.contador* span{
73. border-bottom: 1px solid #ccc;
74. height: 30px;
75. padding: 5px 25px;
76. }
77. *.contador* button{
78. height: 30px;
79. width: 30px;
80. border-radius: 15px;
81. background-color: coral;
82. color: #fff;
83. margin-left: 10px;
84. outline: none;
85. }
86. </style>
87. Após criar o componente iremos registrar o componente de maneira global.
88. Para isso vamos abrir o arquivo main.js e iremos importar o componente contador
89. import Contador from './Contador.vue'
90. Após isso vamos registrar o componente de forma global.
91. Vue.component('app-contador', Contador)
92. Criando o nome ‘app-contador’.
93. No arquivo App.vue iremos utilizar o componente no template.
94. <template>
95. <div *id*="app">
96. <app-contador/>
97. <app-contador/>
98. </div>
99. </template>
100. <script>
101. export default {
103. }
104. </script>
105. <style>
106. </style>

# CORREÇÃO DO ERRO error Component name “home“ should always be multi-word vue/multi-word-component-names

1. Localize o arquivo vue.config.js e inclua a expressão :
2. lintOnSave: false.

**Aula 109 e 110 – Usando componentes**

1. utilizando o exemplo anterior, vamos criar outro componente chamado Contadores.vue e dentro dele iremos criar a estrutura do componente e ele terá 5 componentes do tipo contador dentro de template, porém estes componentes serão renderizados utilizando a diretiva v-for. O componente Contador será importado dentro de script e dentro de export defauld nós iremos utilizar a propriedade componentes, registrando o componente Contador de forma local somente dentro do componente Contadores.
2. <template>
3. <div *class*="contadores">
4. <h1>Contadores</h1>
5. <app-contador *v-for*="c in 5" :*key*="c"/>
6. </div>
7. </template>
8. <script>
9. import Contador from './Contador.vue'
10. export default {
11. components: {'app-contador': Contador },
13. }
14. </script>
15. <style>
16. </style>
17. No arquivo Main.ja iremos registrar de forma global o componente Contadores, mudando app-contador para app-contadores e Contador para Contadores.
18. import Vue from 'vue'
19. import App from './App.vue'
20. import Contadores from './Contadores.vue'
21. Vue.config.productionTip = false
22. Vue.component('app-contadores', Contadores)
23. **new** Vue({
24. render: *h* => h(App),
25. }).$mount('#app')
26. No arquivo App.vue
27. Teremos uma estrutura simples de componente, dentro do template iremos criar uma div com id app, passando para a div o componente “app-contadores”
28. <template>
29. <div *id*="app">
30. <app-contadores/>
31. </div>
32. </template>
33. <script>
34. export default {
36. }
37. </script>
38. <style>
39. </style>

**Aula 111 – Usando CSS com escopo de componente**

1. Quando criamos um estilo dentro do componente este estilo pode impactar em toda a a aplicação, por isso utilizarmos a cláusula scoped dentro da tag style.
2. <style *scoped*>
3. span{
4. border-bottom: 1px solid #ccc;
5. height: 30px;
6. padding: 5px 25px;
7. }
9. button{
10. height: 30px;
11. width: 30px;
12. border-radius: 15px;
13. background-color: coral;
14. color: #fff;
15. margin-left: 10px;
16. outline: none;
17. }
18. </style>

Aula 112 – Hora de praticar: Organizar Projeto em Componentes

1. Após baixar o conteúdo da aula 112 teremos um único componente App, iremos organizar a aplicação quebrando o componente App em vários outros componentes.