Manipulando entrada de usuário com formulário

1. **Aula 169 – Ligação simples utilizando v-model**
2. Para utilizar a diretiva v-model podemos simplesmente criar uma variável dentro da propriedade data()
3. data() {
4. return {
5. //método uitlizando somente a interpolação simples
6. email: ''
7. }
8. }
9. e interpolando essa variável dentro do input.
10. <Rotulo *nome*="E-mail">
11. <input *v-model*="mail" *type*="text">
12. </Rotulo>
13. Resultados
14. <div *class*="painel">
15. <div *class*="cabecalho">Resultado</div>
16. <Rotulo *nome*="E-mail">
17. <span>{{mail}}</span>
18. </Rotulo>
19. Podemos criar também um objeto e passar o objeto com o atributo no input.
20. data() {
21. return {
22. //utilizando um objeto passando todos os atributos necessáros para o registro dos dados
23. user:{
24. mail:'',
25. password:'',
26. age:''
27. }
28. }
29. Interpolando o objeto com seu atribulo no input
30. <form *class*="painel">
31. <div *class*="cabecalho">Formulário</div>
32. <Rotulo *nome*="E-mail">
33. <input *v-model*="user.mail" *type*="text">
34. </Rotulo>
35. Resultado
36. <div *class*="painel">
37. <div *class*="cabecalho">Resultado</div>
38. <Rotulo *nome*="E-mail">
39. <span>{{user.mail}}</span>
40. </Rotulo>

Aula 171 – Modificar entrada de usuário com modificadores de input

1. Se utilizarmos o modificar .lazy o valor do input somente será passado para o backend quando o usuário sair do campo.
2. <Rotulo *nome*="E-mail">
3. <input *v-model*.*lazy*="user.mail" *type*="text">
4. </Rotulo>
5. Podemos utilizar um outro modificador junto com lazy para retirar os espaços em branco do inicio e final dentro do input.
6. <Rotulo *nome*="E-mail">
7. <input *v-model*.lazy.*trim*="user.mail" *type*="text">
8. </Rotulo>
9. Temos também outro modificador que é para garantir que um número seja passado no formato numérico para o backend.
10. <Rotulo *nome*="Idade">
11. <input *type*="number" *v-model*.*number*="user.age">
12. </Rotulo>
13. Lembrando que esse modificador somente pode ser utilizado onde o campo recebe somente números.

Aula 172 – Utilizando textArea e salvando quebras de linha

1. Vamos criar uma variável dentro de data(), chamada mensagem.
2. dentro do input utilizamos v-model e passamos ‘mensagem’.
3. <Rotulo *nome*="Mensagem">
4. <textarea *name*="" *cols*="30" *rows*="5" *v-model*="mensagem"></textarea>
5. </Rotulo>
6. No resultado interpolamos a variável mensagem.
7. <Rotulo *nome*="Mensagem">
8. <span>{{ mensagem }}</span>
9. </Rotulo>
10. Porem por padrão o resultado não terá quebra de linhas pois há uma propriedade CSS chamada White-space que desabilita isso. Para podermos então mostrar as quebras de linha e espaços setamos o White-space: pre; dentro de style no front-end.
11. <Rotulo *nome*="Mensagem">
12. <span *style*="white-space: pre;">{{ mensagem }}</span>
13. </Rotulo>

Aula 173 – Utilizando checkboxex e salvando os dados em um array.

1. Para isso vamos criar um atributo dentro de data(), chamado características.
2. data() {
3. return {
4. //método uitlizando somente a interpolação simples
5. // email: ''
6. //utilizando um objeto passando todos os atributos necessáros para o registro dos dados
7. user:{
8. mail:'',
9. password:'',
10. age:''
11. },
12. mensagem: '',
13. caracteristicas:[]
14. }
15. Agora no lado da exibição vamos criar uma tag <ul> com <li> utilizando um v-for para percorrer o array características e exibir os valores do array.
16. <Rotulo *nome*="Qual produto?">
17. <span>
18. <ul>
19. <li v-for="c in caracteristicas" :*key*="">{{ c }}</li>
20. </ul>
21. </span>
22. </Rotulo>

Aula 174 – Utilizando botões radio

1. Primeiramente temos que incluir nos botões radio o atributo value em cada botão.
2. <Rotulo *nome*="Qual produto?">
3. <span *class*="mr-4"><input *type*="radio" *value*="web"> Web</span>
4. <span *class*="mr-4"><input *type*="radio" *value*="mobile"> Mobile</span>
5. <span><input *type*="radio" *value*="outro"> Outro</span>
6. </Rotulo>
7. Após isso vamos no campo data() e criamos um atributo camado produto e passamos para ele o valor ‘web’ para ser selecionado como padrão.
8. data() {
9. return {
10. //método uitlizando somente a interpolação simples
11. // email: ''
12. //utilizando um objeto passando todos os atributos necessáros para o registro dos dados
13. user:{
14. mail:'',
15. password:'',
16. age:''
17. },
18. mensagem: '',
19. caracteristicas:[],
20. produdo: 'web'
21. }
22. Após isso vamos inserir em cada botão radio um v-model apontando para produto.
23. <Rotulo *nome*="Qual produto?">
24. <span *class*="mr-4"><input *type*="radio" *value*="web" *v-model*="produto"> Web</span>
25. <span *class*="mr-4"><input *type*="radio" *value*="mobile" *v-model*="produto"> Mobile</span>
26. <span><input *type*="radio" *value*="outro" *v-model*="produto"> Outro</span>
27. </Rotulo>
28. Basta agora interpolar a variável produto para visualizar a opção selecionada.
29. <Rotulo *nome*="Marque as Opções">
30. <span>{{ produto }}</span>
31. </Rotulo>

Aula 175 – Manipulando combobox com <select> e <option>

1. Vamos criar um array de objetos com as prioridades.
2. prioridades: [
3. {'codigo':1, nome:'Baixa'},
4. {'codigo':2, nome:'Moderada'},
5. {'codigo':3, nome:'Alta'}
6. ]
7. Agora dentro do option no select, vamos utilizar um v-for para percorrer as opções nesse select utilizaremos :key passando para ele ‘prioridade.codigo’ :value passando também ‘prioridade.codigo’.
8. Lembrando de selecionar qual o atributo do objeto será mostrado no campo do combobox.
9. Podemos utilizar também o recurso de :selected para passar um valor padrão que deverá ser selecionado quando a página for renderizada.

<Rotulo *nome*="Prioridade">

                    <select *name*="" *id*="">

                        <option v-for="prioridade in prioridades"

                            :*key*="prioridade.codigo"

                            :*value*="prioridade.codigo"

                            :*selected*="prioridade.codigo === "

                                >{{ prioridade.nome }}</option>

                    </select>

                </Rotulo>

1. Para mostrarmos o valor teremos que criar um atributo chamado prioridade no campo data()
2. prioridade: 1,
3. passando ele para um v-model no select
4. <Rotulo *nome*="Prioridade">
5. <select *name*="" *id*="" *v-model*="prioridade">
6. <option v-for="prioridade in prioridades"
7. :*key*="prioridade.codigo"
8. :*value*="prioridade.codigo"
9. :*selected*="prioridade.codigo === 3"
10. >{{ prioridade.nome }}</option>
11. </select>
12. </Rotulo>
13. e interpolando o mesmo onde o valor será renderizado.
14. <Rotulo *nome*="Prioridade">
15. <span>{{ prioridade }}</span>
16. </Rotulo>

Aula 176 – O que o V-model faz e como criar um input personalizado.

1. Veremos o componente personalizado chamado ‘escolha’, que possui no template uma div com a class escolha emitindo um evento chamado ‘input’ com o valor ‘!value’ temos também um bind :class que irá atribuir valores de true ou false para as classes css ligado e desligado.
2. Temos outra div com a classe botão
3. Dentro de export default temos uma props que terá uma variável chamada ‘value’ que será do tipo boolean e required true.
4. <template>
5. <div *class*="escolha"
6. @*click*="$emit('input',!value)"
7. :*class*="{ligado: value, desligado: !value}">
8. <div  *class*="botao"></div>
9. </div>
10. </template>
11. <script>
12. export default {
13. props:{
14. value: {
15. type: *Boolean*,
16. required: true
17. }
18. }
20. }
21. </script>
22. No aplicativo Pai APP teremos um rótulo que receberá um componente chamado Escolha e esse componente terá um v-model com a variável escolha atribuída.
23. <Rotulo *nome*="Primeira Reclamação?">
24. <Escolha *v-model*="escolha" />
25. </Rotulo>
26. Teremos dentro do atributo data() uma variável chamada escolha com valor true.
27. escolha: true

**Aula 177 – Submetendo o Formulário**

1. Agora que temos todos os componentes do formulário, lincados, vamos implementar o submit que irá interceptar o clique do botão ‘enviar’ e direcionar para um método. Para isso iremos criar uma variável que irá esconder todo o conteúdo e somente irá mostrar o resultado quando o submitir tiver sido realizado.
2. Pra isso vamos criar a variável chamada enviado que irá receber o valor false. Vamos atribuir essa variável em um v-if dentro da tag form com class=”painel” e um v-else dentro class=’painel’ após o fechamento do form.
3. <form *class*="painel" v-if="!enviado">
4. Ou seja se a variável enviado for false o formulário será mostrado caso contrário ele será escondido e será mostrado o resultado dos campos preenchidos
5. <div *class*="painel" v-else>
6. Para funcionar temos que chamar dentro do botão enviar, através do evento @click o método enviar, que irá basicamente alterar o valor da variável enviado de false para true. Sendo que iremos utilizar o atributo ‘.prevent’do evento click para que não seja executado o comportamento padrão do submit de um formulário.
7. <button @*click*.*prevent*="enviar">Enviar</button>
8. O método enviar ficará da seguinte forma.
9. methods: {
10. enviar(){
11. *this*.enviado = true;
12. }
13. },
14. Aula 178 – Praticando