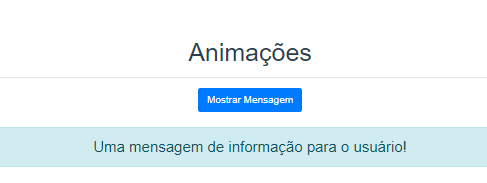
**MELHORANDO ESTÉTICA-TRANSIÇÕES-E-ANIMAÇÕES NO** **VUE.JS**

**Aula 210 – Preparando o código para as transições**

1. Vamos inserir algumas informações no template para podermos utilizar as transições.
2. Primeiramente incluía do campo data() as variáveis msg e exibir sendo que msg irá conter a mensagem que deverá ser mostrada para o usuário e exibir ser do tipo boolean que a principio será false.
3. No template vamos incluir um botão utilizando a estilização do bootstrap, o projeto foi configurado com o bootstrap, para isso bastou-se adiciona-lo ao projeto com npm i bootstrap-vue .
4. Vamos incluir também um b-alert.
5. Dentro do botão teremos uma chamada para o evento @click que ficará alterando o valor de exibir entre true e false toda vez que o botão for clicado.
6. Dentro de b-alert teremos um v-if que receberá a variável exibir, desta forma o alert somente será exibido quando a variável exibir for true.
7. <template>
8. <div *id*="app" *class*="container-fluid">
9. <h1>Animações</h1>
10. <hr>
11. <b-button *variant*="primary" *class*="mb-4"
12. @*click*="exibir = !exibir">Mostrar Mensagem</b-button>
13. <b-alert *variant*="info" *show* v-if="exibir">{{msg}}</b-alert>
14. </div>
15. </template>
16. <script>
17. export default {
18. data() {
19. return {
20. msg: 'Uma mensagem de informação para o usuário!',
21. exibir: false
22. }
23. },
24. }
25. </script>
26. <style>
27. *#app* {
28. font-family: 'Avenir', Helvetica, Arial, sans-serif;
29. *-webkit-font-smoothing*: antialiased;
30. *-moz-osx-font-smoothing*: grayscale;
31. text-align: center;
32. color: #2c3e50;
33. margin-top: 60px;
34. font-size: 1.5rem;
35. }
36. </style>
37. Para utilizarmos o efeito de transições temos que utilizar a tag <transition></transition> que executará a transição do elemento que está dentro da tag.
38. 

**Aula 211 – Configurando transição**

**Aula 212 – Definindo as classes CSS para transição**

1. Para utilizarmos as transições devemos definir as classes que serão utilizadas.
2. Na tag transition vamos incluir o name=’fade’
3. E na área de style iremos então definir as classes.
4. .fade-enter{} .fade-enter-active{}, .fade-enter-to{}, .fade-leave{}, .fade-leave-active{}, .fade-leave-to{}.
5. <style>
6. *#app* {
7. font-family: 'Avenir', Helvetica, Arial, sans-serif;
8. *-webkit-font-smoothing*: antialiased;
9. *-moz-osx-font-smoothing*: grayscale;
10. text-align: center;
11. color: #2c3e50;
12. margin-top: 60px;
13. font-size: 1.5rem;
14. }
15. *.fade-enter*{
16. }
17. *.fade-enter-active*{
18. }
19. *.fade-enter-to*{
20. }
21. *.fade-leave*{
22. }
23. *.fade-leave-enter*{
24. }
25. *.fade-leave-to*{
27. }
28. </style>

**Aula 213 – Criando transição Fade com propriedade CSS transition**

1. Vamos incluir na classe .fade-enter{} o atributo opacity com valor zero, ou seja no momento que o objeto for exibido ele estará invisível, na próxima classe .fade-enter-active{} utilizaremos agora o atributo transition passando para ele opacity 2s, ou seja o elemento irá demorar 2 segundos para sair da opacidade zero até um que é o máximo.
2. A classe .fade-enter-to{} por padrão o atributo opacity será 1, por isso podemos retirá-lo, mas caso a opacidade final não seja 1, devemos informar a opacidade de destino.
3. Temos então a transição de entrada, para fazermos a transição de saída basta configuramos as classes .fade-leave{} para 1 ou seja quando o elemento for ocultado ele começará com 1 e irá finalizar com 0 e o tempo de transição será 2s.
4. *.fade-enter*, *.fade-leave-to*{
5. opacity: 0;
6. }
7. *.fade-enter-active*, *.fade-leave-active*{
8. transition: opacity 2s;
9. }

**Aula 214 – Criando transição Slide com propriedade CSS.**

1. Vamos duplicar a tag transition anterior porém vamos nomeá-la com ‘slide’.
2. Teremos que definir as mesmas classes que foram definidas na transição anterior, ou seja, .slide-enter{} .slide-enter-active{} .slide-enter-to{} .slide-leave{} .slide-leave-active{} .slide-leave-to{}.
3. Porém vamos utilizar um recurso css chamado keyframes que são os atributos css que se quer aplicar em uma transformação.
4. Desta forma iremos agir nos eixos x, y e z.
5. Vamos então definir que o elemento irá sair de um ponto 0 e irá se mover 40px para cima.
6. Fazemos isso definindo da seguinte forma:
7. Entrada do slide -
8. @keyframes *slide-in*{
9. from { transform: translateY(40px);}
10. to {transform: translateY(0);}
11. }
12. Saída do slide –
13. @keyframes *slide-out*{
14. from { transform: translateY(0);}
15. to {transform: translateY(40px);}
16. }
17. Mas para a transição funcionar, agora devemos configurar as classes do slide.
18. Na classe .slide-enter-active{} nós iremos determinar uma animação do slide-in de 2s com o algoritmo ease, esse algoritmo é que dára a vida a animação e exite vários tipos.
19. Temos que configurar agora a animação de saída na classe .slice-leave-active{} que terá a animação slide-out com 2s e o mesmo algoritmo ease.
20. *.slide-enter-active*{
21. animation: slide-in 2s ease;
22. }
23. *.slide-leave-active*{
24. animation: slide-out 2s ease;
25. }

**Aula 215 – Misturando as propriedades Animation e transition**

1. Caso o tempo de transição das duas animações sejam diferentes, basta utilizarmos o atributo type na tag <transition> passando para o type qual a transição que irá mandar.
2. Para realizarmos a mistura dos dois tipos de animação basta utilizarmos as duas animações nas classes .slide-enter-active{} e .slide-leave-active{}
3. *.slide-enter-active*{
4. animation: slide-in 2s ease;
5. transition: opacity 2s;
6. }
7. *.slide-leave-active*{
8. animation: slide-out 2s ease;
9. transition: opacity 2s;
10. }

**Aula 216 – V-show**

1. Podemos utilizar o v-show da mesma forma que utilizamos o v-if.
2. <transition *name*="slide" *appear* >
3. <b-alert *variant*="info" *show* *v-show*="exibir">{{msg}}</b-alert>
4. </transition>

**Aula 217 – Configurando animação no carregamento do** **componente.**

1. Para podermos carregar a transição de um componente ao iniciar a aplicação devemos utilizar o atributo appear na tag <transition>, para exemplificar vamos utilizar na transition de slide.
2. <transition *name*="slide" *appear* >
3. <b-alert *variant*="info" *show* *v-show*="exibir">{{msg}}</b-alert>
4. </transition>
5. E passar a variável exibir para true.
6. data() {
7. return {
8. msg: 'Uma mensagem de informação para o usuário!',
9. exibir: true
10. }

**Aula 218 – Usando nomes diferentes de classes CSS**

1. É possível utilizar outro padrão de nomenclatura das classes do vue, sem problema nenhum, para isso.
2. Vamos encontrar a biblioteca na página <https://animate.style/>
3. Que possui várias animações. Vamos copiar o link do cdn e colar na nossa página index.html.
4. <link
5. rel="stylesheet"
6. href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"
7. />
8. Vamos duplicar a última transition do template e retirar o nome e appear.
9. dentro da tag transition vamos incluir as classes enter-active-class e leave-active-class e iremos utilizar uma classe da biblioteca animate.
10. <transition
11. *enter-active-class*="animated bounce"
12. *leave-active-class*="animated shake"  >
13. <b-alert *variant*="info" *show* *v-show*="exibir">{{msg}}</b-alert>
14. </transition>
15. Passe o valor da variável exibir para false.

**Aula 219 – Usando nomes e atributos dinâmicos**

1. Os nomes dos atributos da tag <transition> podem ser passados dinamicamente utilizando bind. Para exemplificar vamos duplicar a tag transition do slide.
2. <transition :*name*="tipoAnimacao" >
3. <b-alert *variant*="info" *show* *v-show*="exibir">{{msg}}</b-alert>
4. </transition>
5. Dentro de data() vamos criar uma variável chamada tipoAnimacao que terá um valor inicialmente de ‘fade’.
6. data() {
7. return {
8. msg: 'Uma mensagem de informação para o usuário!',
9. exibir: false,
10. tipoAnimacao: 'fade'
11. }
12. },
13. Agora no template vamos inserir uma tag <b-select> e dentro dela duas tags <option> com as opções de value ‘fade’e ‘slide’.
14. Na tag b-select iremos utilizar um v-model apontando para tipoAnimacao.
15. Na tag transition que foi duplicada vamos incluir um bind na propriedade :name passando tipoAnimacao para ela.
16. <b-select *v-model*="tipoAnimacao" *class*="mb-4">
17. <option *value*="fade">Fade</option>
18. <option *value*="slide">Slide</option>
19. </b-select>
20. <transition :*name*="tipoAnimacao" >
21. <b-alert *variant*="info" *show* *v-show*="exibir">{{msg}}</b-alert>
22. </transition>
23. C

**Aula 220 – Transicionar entre múltiplos elementos**

1. Vamos duplicar o último <transition> e acrescentar mais um alert com a variant warning, devemos alterar também o v-show para v-if e v-else.
2. Porém para funcionar a transição entre eles devemos inserir mais um atributo nas tags alert, que será o key passando como valor info e warning.
3. <transition :*name*="tipoAnimacao" >
4. <b-alert *variant*="info" *show* v-if="exibir" *key*="info">{{msg}}</b-alert>
5. <b-alert *variant*="warning" *show* v-else *key*="warning">{{msg}}</b-alert>
6. </transition>
7. Mas dessa forma ainda aparece um erro pois os elementos não estão com a ordem de entrada e saída bem definidos, para resolver isso vamos incluir um atributo em transition chamado mode passando para ele ‘out-in’ou seja primeiramente será mostrado o elemento que está saindo para depois mostrar o elemento que está entrando.