vue过渡动画的使用

一、vue过渡动画transtion

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>vue内置过滤动画组件</title>
    <style>
        .box {
            width: 200px;
            height: 200px;
            background-color: deeppink;
        }
    </style>
</head>

<body>
    <div id="app">
        <button type="button" @click="flag=!flag">切换</button>
        <div class="box" v-show="flag"></div>
    </div>
</body>
<script src="./js/vue.global.js"></script>
<script>
    Vue.createApp({
        data() {
            return {
                flag: true
            }
        }
    }).mount("#app")
</script>

</html>
```

> 在上面的代码不如，我们可以看到，当我们去改变 `isShow` 的时候， 盒子box在隐藏与显示，那么，我们能不能让这个盒子在隐藏与显示的时候执行一些特殊的动画效果呢?

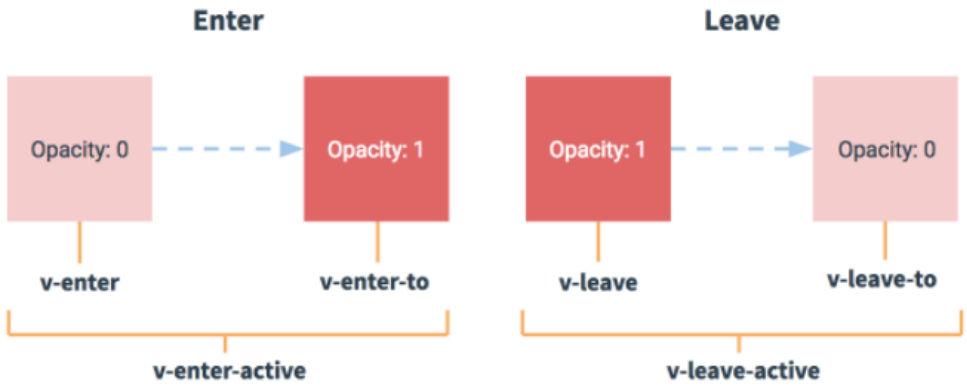
在vue的内部是可以直接进行的，vue的内部自带动画管理机制，在vue的内部是可以直接使用过渡的效果，它内部封装了一个 `<transition>` 的组件，可以让元素在 `进入/离开` 的时候执行特定的过渡效果

```
<transition>
    <div v-show="isShow" class="box"></div>
</transition>
```

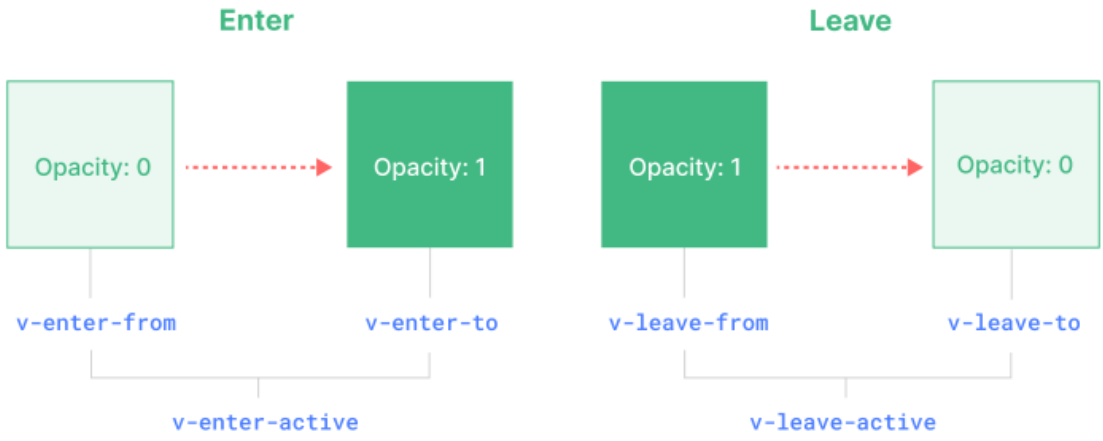
在上面的代码里面，我们将`box`使用`<transition>`进行包裹，box就可以执行动画了，但是我们并没有看到效果，原因是因为transiton的使用要进行相关的设置

box是需果隐藏与显示的，所以我们可以把这个盒子的隐藏与显示分为四个阶段，二个过程

1. 隐藏之前
2. 隐藏之后
3. 显示之前
4. 显示之后



上面的代码是vue2版本的示意图，在vue3的版本里面，已经做了更改，如下所示



上面就是vue中`<transition>`的过渡图，然后我们根据图片上面的提示得到了上面的代码

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>过渡动画</title>
    <style>
        .box {
            width: 200px;
            height: 200px;
            background-color: deeppink;
        }
```

```
15
16          /* 假设现在我们希望这个盒子在这里执行的是淡入淡出 */
17          /* 离开之前 */
18          .fade-leave-from {
19              opacity: 1;
20          }
21
22          /* 离开以后 */
23          .fade-leave-to {
24              opacity: 0;
25          }
26
27          /* 进入之前 */
28          .fade-enter-from {
29              opacity: 0;
30          }
31
32          /* 进入以后 */
33          .fade-enter-to {
34              opacity: 1;
35          }
36
37          /* 进入的过程 */
38          .fade-enter-active {
39              transition: all 1s linear;
40          }
41
42          /* 离开的过程 */
43          .fade-leave-active {
44              transition: all 1s linear;
45          }
46      </style>
47  </head>
48
49  <body>
50      <div id="app">
51          <button type="button" @click="isShow = !isShow">切换</button>
52          <transition name="fade">
53              <div v-show="isShow" class="box"></div>
54          </transition>
55      </div>
56  </body>
57  <script src="./js/vue.global.js"></script>
58  <script>
59      Vue.createApp({
60          data() {
61              return {
62                  isShow: true
63              }
64          }
65      }).mount("#app")
66  </script>
67
68  </html>
```

在上面的代码里面，我们可以将CSS进行简化

```css
/* 假设现在我们希望这个盒子在这里执行的是淡入淡出 */
/* 进入之前,离开以后 */
.fade-enter-from,
.fade-leave-to {
    opacity: 0;
}

/* 离开之前,进入以后 */
.fade-levae-from,.fade-enter-to{
    opacity: 1;
}
/* 进入的过程,离开的过程 */
.fade-enter-active,.fade-leave-active{
    transition: all 1s linear;
}
```

1. `v-enter-from` 代表进入之前
2. `v-enter-to` 代表进入以后
3. `v-leave-from` 代表离开以前
4. `v-leave-to` 代表离开以后

而上面的4个状态会形成2个过程，分别是进入的过程与离开的过程

1. `v-enter-active` 进入的过程
2. `v-leave-active` 离开的过程

前面的 `v` 指的是在 `<transition>` 组件上面的 `name` 属性值

tranition的组件下面只能有一个子元素，如果有多个子元素是会报的，如下所示

```vue
<transition name="box">
    <div class="box" v-show="flag">第一个盒子</div>
    <div class="box2" v-show="!flag">第一个盒子</div>
</transition>
```

```
▶ [Vue warn]: Template compilation error: <Transition>    vue.global.js:1622
expects exactly one child element or component.
2  |          <button type="button" @click="flag=!flag">切换</button>
3  |          <transition name="box">
4  |              <div class="box" v-show="flag">第一个盒子</div>
   |              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
5  |              <div class="box2" v-show="!flag">第一个盒子</div>
   |  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
6  |          </transition>
  at <App>
▶ [Vue warn]: <transition> can only be used on a single    vue.global.js:1622
element or component. Use <transition-group> for lists.
  at <BaseTransition appear=false persisted=true mode=undefined  ... >
  at <Transition name="box" persisted="" >
  at <App>
```

二、多元素同时执行过渡动画transition-group

如果需要在多个元素上面同时执行过渡效果，则需要使用`<transitoin-group>`

在这里还有一个小小的注意事项，如果有多个组件同时执行过的时候，则要使用`<transition-group>`去实现

```
<transition-group tag="div" class="img-box">
    <img v-show="currentIndex==0" src="./img/item1.jpg" :key="0" alt="">
    <img v-show="currentIndex==1" src="./img/item2.jpg" :key="1" alt="">
    <img v-show="currentIndex==2" src="./img/item3.jpg" :key="2" alt="">
    <img v-show="currentIndex==3" src="./img/item4.jpg" :key="3" alt="">
</transition-group>
```

`<transition-group>`默认情况下会帮我们生成一个`span`标签，但是我们仍然可以通过设置`tag`来生成指定的标签

transition案例

我们现在使用`transition`来完成一组左右切换的过程，这个过程后期会有项目当中的页面切换里面去使用

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>vue的过渡动画</title>
    <style>
        .img-box {
            width: 639px;
            height: 426px;
            border: 2px solid black;
            position: relative;
        }

        .img-box>img {
            position: absolute;
            left: 0px;
            top: 0px;
        }

        /* 编写vue的过滤动画  */
        /* 进入之前,离开以后 */
        .fade-enter-from,
        .fade-leave-to {
            opacity: 0;
        }
        /* 进入之后,离开之前 */
        .fade-enter-to,.fade-leave-from{
            opacity: 1;
        }
        .fade-enter-active,.fade-leave-active{
            transition: all 0.5s linear;
        }
```

```
    </style>
</head>

<body>
    <div id="app">
        <transition-group name="fade" tag="div" class="img-box">
            <img v-show="currentIndex==index" :src="item" v-for="(item,index) in imgList" :key="index" alt="">
        </transition-group>
        <button type="button" @click="currentIndex--">
            <--前</button>
                <button type="button" @click="currentIndex++">后-->
        </button>
    </div>
</body>
<script src="./js/vue.js"></script>
<script>
    Vue.createApp({
        el: "#app",
        data: {
            currentIndex: 0,
            imgList: [
                "./img/item1.jpg",
                "./img/item2.jpg",
                "./img/item3.jpg",
                "./img/item4.jpg"
            ]
        },
        watch: {
            currentIndex(newValue, oldValue) {
                if (newValue > 3) {
                    this.currentIndex = 0;
                } else if (newValue < 0) {
                    this.currentIndex = 3;
                }
            }
        }
    }).mount("#app")
</script>

</html>
```

> 在上面的代码里同，我们通过最基本的transition-group实现了淡入淡出的轮播图效果

案例

通过vue的过渡动画，实现一个滚动的轮播效果

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>vue-transition-group动画</title>
    <style>
        .img-box {
            width: 639px;
            height: 426px;
            border: 5px solid red;
            position: relative;
            overflow: hidden;
            margin: auto;
        }

        .img-box>img {
            position: absolute;
            left: 0;
            top: 0;
        }

        /* --------------------------从右向左走-----swiper-----------------------
---- */
        /* 进入之前 */
        .swiper-enter-from {
            transform: translateX(100%);
        }

        /* 进入之后 */
        .swiper-enter-to {
            transform: translateX(0%);
        }

        /* 进入的过程 */
        .swiper-enter-active {
            transition: all 0.5s linear;
        }

        /* 离开之前 */
        .swiper-leave-from {
            transform: translateX(0%);
        }

        /* 离开之后 */
        .swiper-leave-to {
            transform: translateX(-100%);
        }

        /* 离开的过程 */
        .swiper-leave-active {
            transition: all 0.5s linear;
        }

        /* --------------------------从左向右的变化--swiper2----------------------
------ */

        /* 进入之前 */
        .swiper2-enter-from {
            transform: translateX(-100%);
        }
```

```
62
63          /* 进入之后 */
64          .swiper2-enter-to {
65              transform: translateX(0%);
66          }
67
68          /* 进入的过程 */
69          .swiper2-enter-active {
70              transition: all 0.5s linear;
71          }
72
73          /* 离开之前 */
74          .swiper2-leave-from {
75              transform: translateX(0%);
76          }
77
78          /* 离开之后 */
79          .swiper2-leave-to {
80              transform: translateX(100%);
81          }
82
83          /* 离开的过程 */
84          .swiper2-leave-active {
85              transition: all 0.5s linear;
86          }
87      </style>
88  </head>
89
90  <body>
91      <div id="app">
92          <button type="button" @click="transitionGroupName='swiper2';currentIndex--">前一张</button>
93          <button type="button" @click="transitionGroupName='swiper';currentIndex++">后一张</button>
94          <div class="img-box">
95              <transition-group :name="transitionGroupName">
96                  <img v-for="(item,index) in list"
97                      :src="item" :key="item"
98                      v-show="currentIndex==index">
99              </transition-group>
100         </div>
101     </div>
102 </body>
103 <script src="./js/vue.global.js"></script>
104 <script>
105     Vue.createApp({
106         data() {
107             return {
108                 transitionGroupName: "swiper",
109                 currentIndex: 0,
110                 list: [
111                     "./img/item1.jpg",
112                     "./img/item2.jpg",
113                     "./img/item3.jpg",
114                     "./img/item4.jpg"
115                 ]
```

```
            }
        },
        mounted() {
            setInterval(() => {
                this.currentIndex++;
            }, 5000);
        },
        watch: {
            currentIndex(newValue, oldValue) {
                if (newValue < 0) {
                    this.currentIndex = 3;
                } else if (newValue > 3) {
                    this.currentIndex = 0;
                }
            }
        }
    }).mount("#app");
</script>

</html>
```

三、transition的钩子函数

```
<transition
  @before-enter="beforeEnter"
  @enter="enter"
  @after-enter="afterEnter"
  @enter-cancelled="enterCancelled"

  @before-leave="beforeLeave"
  @leave="leave"
  @after-leave="afterLeave"
  @leave-cancelled="leaveCancelled"
>
  <!-- ... -->
</transition>
```

transition的钩子函数其实可以理解为当过渡动画执行完以后要做的某些阶段的函数

四、总结

在vue的内部，有一些自带的组件

1. `Component` 动态组件渲染
2. `transitioin` 执行单个元素的过渡动画
3. `transition-group` 执行多个元素以的过渡动画

关于过渡的总结

1. 2个过程，进入的过程 `enter-active` ，离开的过程 `leave-actkve`
2. 4个状态，进入之前 `enter-from` ，进入之后 `enter-to` ，离开之前 `leave-from` ,离开之后 `leave-to`
3. 多个元素一定要使用 `transition-group` ，要在子元素上面指定 `key`
4. `transform-group` 默认生成的是 `span` 标签，如果希望生成其它的标签，可以使用 `tag`

五、补充Animate.css的使用

中文官方网站

http://www.animate.net.cn/

在现在框架里面，有很多第三方的动画库，我们常用的一个动画库是`animate.css`，这个动画库可以与vue进行完美的结合

1. 安装

```
$ yarn add animate.css
```

2. 导入到系统当中

```
<link rel="stylesheet" href="./css/animate.css">
```

3. 使用

```
<transition-group
                    enter-active-class="animate__animated animate__bounceInRight"
                    leave-active-class="animate__animated animate__rotateOutUpLeft"
                    tag="div" class="img-box">
    <img v-show="currentIndex==index" :src="item" v-for="(item,index) in imgList" :key="index" alt="">
</transition-group>
```