第 5 章　Vue 过渡和动画

在 Vue 项目中使用过渡和动画能优化用户体验和页面的交互性、影响用户的行为、引导用户的注意力以及帮助用户看到自己动作的反馈。例如，Vue 导航切换使用了过渡动画，用户体验更友好。本章将结合案例讲解 Vue 项目中过渡和动画的实现，及其在多个元素或组件中的使用方法。

本章要点

- 了解过渡和动画的含义
- 掌握内置过渡类名及自定义类名的使用方法
- 掌握单元素、多元素、多组件的过渡
- 掌握列表过渡的实现方法

励志小贴士

你无法决定明天是晴是雨，也无法决定此刻的坚持能得到什么。但你能决定今天有没有准备好雨伞，以及是否足够努力。胡思乱想少一点，脚踏实地多一些，就会距离成功更近一步。

5.1 过渡与动画

5.1.1 了解过渡与动画

在 CSS 3 中，过渡属性通过 transition 实现，动画属性通过 animation 实现。Vue 也实现了过渡与动画，在插入、更新或者移除 DOM 时，Vue 提供了多种过渡效果。Vue.js 会在适当的时机触发 CSS 过渡或动画，也可以提供相应的 JavaScript 钩子函数以在过渡过程中执行自定义的 DOM 操作。

Vue 提供了内置的过渡封装组件，即 transition 组件，其语法格式如下：

```
<transition name="fade">
<!--需要添加过渡的 div 标签-->
<div></div>
</transition>
```

上述代码中，＜transition＞标签用来放置需要添加过渡的 div 元素，使用 name 属性可以设置前缀，如果将 name 属性设置为 fade，那么“fade-”就是在过渡中切换的类名前缀，如 fade-enter、fade-leave 等。如果＜transition＞标签上没有设置 name 属性名，那么“v-”就是这些类名的默认前缀，如 v-enter、v-leave 等。Vue 提供了 6 个 CSS 类名，分别为 v-enter、v-enter-active、v-enter-to、v-leave、v-leave-active、v-leaveto。

通过＜transition＞标签搭配 CSS 动画（如@keyframes）可以实现动画效果，另外，＜transition＞标签还提供了一些钩子函数，可以结合 JavaScript 代码完成动画效果，具体内容会在后文讲解。

5.1.2 transition 组件

Vue 提供了 transition 的封装组件，在下列情形中，可以给任何元素和组件添加进入/离开过渡：

- 条件渲染（使用 v-if）
- 条件展示（使用 v-show）
- 动态组件
- 组件根节点

Vue 为 transition 提供了 3 个进入过渡的类和 3 个离开过渡的类，具体如表 5-1 所示。

表 5-1 过渡类型

过渡状态	过渡类型	说明
进入（enter）	v-enter	进入过渡的开始状态，作用于开始的一帧
	v-enter-active	进入过渡生效时的状态，作用于整个过程
	v-enter-to	进入过渡的结束状态，作用于结束的一帧

续表

过渡状态	过渡类型	说明
离开(leave)	v-leave	离开过渡的开始状态,作用于开始的一帧
	v-leave-active	离开过渡生效时的状态,作用于整个过程
	v-leave-to	离开过渡的结束状态,作用于结束的一帧

表 5-1 中,6 个类的生效时间如下。

- v-enter:在元素被插入之前生效,在元素被插入之后的下一帧移除。
- v-enter-active:在整个进入过渡的阶段中应用,在元素被插入之前生效,在过渡动画完成之后移除。
- v-enter-to:在元素被插入之后的下一帧生效(与此同时,v-enter 被移除),在过渡动画完成之后移除。
- v-leave:在离开过渡被触发时立刻生效,下一帧被移除。
- v-leave-active:在整个离开过渡的阶段中应用,在离开过渡被触发时立刻生效,在过渡动画完成之后移除。
- v-leave-to:在离开过渡被触发之后的下一帧生效(与此同时,v-leave 被移除),在过渡动画完成之后移除。

Vue 过渡具体如图 5-1 所示。

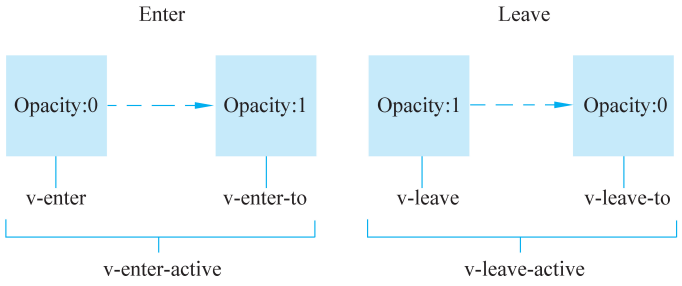

图 5-1 Vue 过渡

下面通过一个案例演示如何使用内置的 class 类名实现过渡。

【例 5-1】 使用内置的 class 类名实现过渡。

(1) 创建文件夹 chapter05,在该目录下创建 demo01.html 文件,具体代码如下:

```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>Transition 标签</title>
    <style>
        .box {
            width: 200px;
```

```
9                height: 50px;
10               background-color: orange;
11           }
12           /* 进入和离开的过程 */
13           .fade-enter-active, .fade-leave-active {
14               transition: width 3s; /* width 的变化,动画时间是 3 秒 */
15           }
16           /* 进入的初始状态和离开的结束状态 */
17           .fade-enter, .fade-leave-to {
18               width: 0px;
19           }
20           /* 进入的结束状态和离开的初始状态 */
21           .fade-enter-to, .fade-leave {
22               width: 200px;
23           }
24       </style>
25   </head>
26   <body>
27   <div id="app">
28       <button type="button" @click="show=!show">Toggle</button>
29       <transition name="fade">
30           <div class="box" v-if="show">
31               <h1>Hello,Vue!</h1>
32           </div>
33       </transition>
34   </div>
35   <script src="vue.js"></script>
36   <script>
37       var vm = new Vue({
38           el: '#app',
39           data: {
40               show: true
41           }
42       })
43   </script>
44   </body>
45   </html>
```

在上述代码中,第 29 行将＜transition＞标签的 name 属性值设置为 fade,这里的 fade 是自定义类名前缀,因此在写 CSS 样式时,相对应的类名前缀以“fade-”开头;如果 transition 不设置 name 属性,则第 12～23 行设置的 CSS 类均以“v-”为前缀;第 30 行的 div 元素为一个长方形,通过使用 v-if 指令切换组件的可见性,通过 show 设置显示的状态,这样在单击按钮时,可以通过切换布尔值实现元素的显示和隐藏。在代码的第 12～23 行编写以“fade-”开头的 CSS 样式,以实现动画效果。

（2）在浏览器中打开 demo01.html 文件，运行结果如图 5-2 所示。

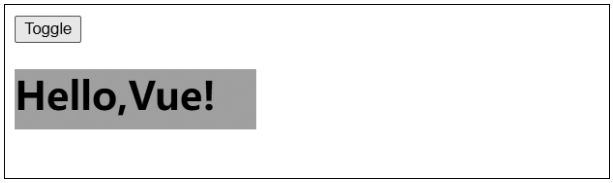

图 5-2　运行初始效果

在图 5-2 所示的页面中，单击 Toggle 按钮，会看到图形宽度变化的动画效果，其变化过程中的某个效果如图 5-3 所示。

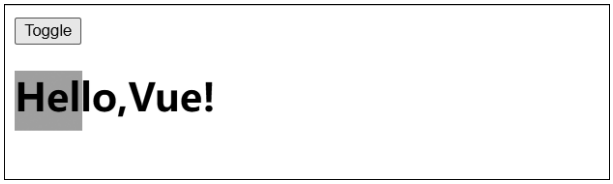

图 5-3　动画效果

5.2　单元素/组件的过渡

实现过渡动画通常有三种方式，一是使用 Vue 的＜transition＞标签结合 CSS 样式实现动画；二是利用 animate.css 结合 transition 实现动画；三是利用 Vue 中的钩子函数实现动画。下面具体讲解实现过渡动画的三种方式。

5.2.1　使用@keyframes 创建 CSS 动画

使用@keyframes 创建 CSS 动画时，v-enter 类名在节点插入 DOM 后不会立即删除，而是在 animationend（动画结束）事件触发时删除。

@keyframes 规则创建动画是指将一套 CSS 样式逐步演变成另一套样式，在创建动画的过程中可以多次改变 CSS 样式，通过百分比或关键词 from 和 to（等价于 0 和 100%）规定动画的状态。@keyframes 的语法格式如下：

```
@keyframes animation-name {
    keyframes-selector { css-styles; }
}
```

在上述语法中，keyframes-selector 表示动画时长的百分比，css-styles 表示一个或者多个合法的 CSS 样式属性。

下面通过例 5-2 演示如何使用@keyframes 创建 CSS 动画。

【例 5-2】　使用@keyframes 创建 CSS 动画。

(1) 创建 chapter05/demo02.html 文件,具体代码如下:

```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>Transition 标签</title>
    <style>
        .box {
            width: 200px;
            height: 50px;
            background-color: orange;
        }
        .v-enter-active {
            animation: animate 1s;
        }
        .v-leave-active {
            animation: animate 1s reverse;
        }
        @keyframes animate {
            0%{
                opacity: 0;
                transform: translateX(400px) scale(1);
            }
            50%{
                opacity: .5;
                transform: translateX(200px) scale(1.5);
            }
            100%{
                opacity: 1;
                transform: translateX(0) scale(1);
            }
        }
    </style>
</head>
<body>
<div id="app">
    <button @click="show = !show">click</button>
    <transition>
        <div class="box" v-if="show">hello world</div>
    </transition>
</div>
<script src="vue.js"></script>
<script>
```

```
43        var vm = new Vue({
44            el: '#app',
45            data: {
46                show: true
47            }
48        })
49    </script>
50    </body>
51    </html>
```

在上述代码中，第 36 行给 button 按钮添加了单击事件，通过单击按钮可以改变变量 show 的值，＜transition＞标签没有设置 name 属性，所以第 12～17 行的 CSS 样式使用了“v-”作前缀。第 18～31 行用于通过@keyframes 规则创建名称为 animate 的动画样式，其中，0 表示动画的开始状态，100％表示动画的结束状态。

（2）在浏览器中打开 demo02.html 文件，如图 5-4 所示。

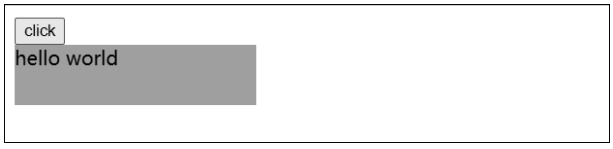

图 5-4　页面初始效果

在图 5-4 所示的页面中，单击 click 按钮，会看到图形变化的动画效果，其变化过程中的某个效果如图 5-5 所示。

图 5-5　CSS 动画

5.2.2　animate.css 结合 transition 实现动画

animate.css 是一个跨浏览器的 CSS 3 动画库，它内置了很多经典的 CSS 3 动画，使用起来很方便。

animate.css 的官网地址是 https://daneden.github.io/animate.css/。

下面通过例 5-3 讲解如何使用自定义类名和 animate.css 库实现动画效果。

【例 5-3】　使用自定义类名和 animate.css 库实现动画效果。

（1）创建 chapter05/demo03.html 文件，并且引入 animate.css，具体代码如下：

```
1    <!DOCTYPE html>
2    <html lang="zh">
3    <head>
```

```
4        <meta charset="UTF-8">
5        <title>animation.css</title>
6        <link rel="stylesheet"
      href =" https://cdnjs. cloudflare. com/ajax/libs/animate. css/4. 1. 1/
      animate.min.css"/>
7    <style>
8            .box {
9                width: 200px;
10               height: 50px;
11               background-color: orange;
12           }
13       </style>
14   </head>
15   <body>
16   <div id="app">
17       <button @click="show = !show">click</button>
18       <transition
19               enter-active-class="animated bounceInLeft"
20               leave-active-class="animated bounceOutLeft" >
21           <div class="box" v-if="show">hello world</div>
22       </transition>
23   </div>
24   <script src="vue.js"></script>
25   <script>
26       var vm = new Vue({
27           el: '#app',
28           data: {
29               show: true
30           }
31       })
32   </script>
33   </body>
34   </html>
```

上述代码中，第 6 行引入了 animate.css 文件，第 19、20 行给＜transition＞标签设置了 enter-active-class 与 leave-active-class 两个属性，用来自定义类名，属性值为 animate.css 动画库中定义好的类名。例如，第 19 行的 animated bounceInLeft 包含两个类名，animated 是基本的类名，任何想实现动画的元素都要添加它；bounceInLeft 是动画的类名，bounceInLeft 表示入场动画，bounceOutLeft 表示出场动画。

（2）在浏览器中运行 demo03.html 文件，单击 click 按钮，即可看到文字显示或隐藏的动画效果，如图 5-6 所示。

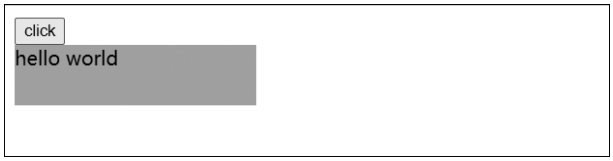

图 5-6　初始效果

5.2.3　钩子函数实现动画

除了使用 CSS 动画外，还可以借助 JavaScript 完成动画。<transition>标签中定义了一些动画钩子函数，可以用来实现动画，包括如下：

```
1    <transition>
2        @before-enter="beforeEnter"
3        @enter="enter"
4        @after-enter="afterEnter"
5        @enter-cancelled="enterCancelled"
6        @before-leave="beforeLeave"
7        @leave="leave"
8        @after-leave="afterLeave"
9        @leave-cancelled="leaveCancelled"
10       v-bind:css="false">
11   </transition>
```

在上述代码中，入场钩子函数分别是 beforeEnter(入场前)、enter(入场)、afterEnter(入场后)和 enterCancelled(取消入场)，出场钩子函数分别是 beforeLeave(出场前)、leave(出场)、afterLeave(场后)和 leaveCancelled(取消出场)；第 10 行为仅使用 JavaScript 过渡的元素添加 v-bind:css="false"，表示 Vue 会跳过 CSS 的检测，以免在过渡过程中受到 CSS 的影响。

下面通过例 5-4 讲解如何使用钩子函数实现动画效果。

【例 5-4】　使用钩子函数实现动画效果。

(1) 创建 chapter05/demo04.html 文件，具体代码如下：

```
1    <!DOCTYPE html>
2    <html lang="zh">
3    <head>
4        <meta charset="UTF-8">
5        <title>动画中的 JavaScript 钩子函数的实现</title>
6        <script src="vue.js"></script>
         <style>
                .ball {
7               width: 15px;
8               height: 15px;
```

```
            border-radius: 50%;
            background-color: orangered;
        }
    </style>
</head>
<body>
<div id="app" >
    <input type="button" value="快到碗里来" @click="flag=!flag">
    <transition
            @before-enter="beforeEnter"
            @enter="enter"
            @after-enter="afterEnter">
        <div class="ball" v-show="flag"></div>
    </transition>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            flag: false
        },
        methods: {
            //设置小球开始动画之前的起始位置
            beforeEnter(el) {
                el.style.transform = "translate(0, 0)"
            },
            enter(el, done) {
                //这句话没有实际的作用,但如果不写,出不来动画效果
                //可以认为 el.offsetWidth 会强制动画刷新
                el.offsetWidth
                el.style.transform = "translate(150px, 450px)"
                el.style.transition = 'all 1s ease'
                //这里的 done,其实就是 afterEnter 这个函数,也就是说,done 是
                //afterEnter 函数的引用
                done()
            },
            afterEnter(el){
                //动画完成之后
                this.flag = !this.flag
            }
        }
    })
</script>
</body>
</html>
```