

Vue3过渡&动画实现

一手微信study322

王红元 coderwhy

- 在开发中，我们想要给一个组件的**显示和消失添加某种过渡动画**，可以很好的**增加用户体验**：
 - React框架本身并**没有提供任何动画相关的API**，所以在React中使用过渡动画我们需要使用一个**第三方库 react-transition-group**；
 - Vue中为我们提供**一些内置组件和对应的API**来完成动画，利用它们我们可以**方便的实现过渡动画效果**；
- 我们来看一个案例：

一手微信study322

 - Hello World的显示和隐藏；
 - 通过下面的代码实现，是不会有动画效果的；

Toogle

Hello World

- 没有动画的情况下，**整个内容的显示和隐藏会非常的生硬**：
 - 如果我们希望给**单元素或者组件实现过渡动画**，可以使用 **transition 内置组件**来完成动画；

Vue的transition动画

■ 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

```
<transition name="fade">
  <h2 v-if="show">Hello World</h2>
</transition>
```

一手微信study322

```
<style scoped>
  .fade-enter-from,
  .fade-leave-to {
    opacity: 0;
  }

  .fade-enter-to,
  .fade-leave-from {
    opacity: 1;
  }

  .fade-enter-active,
  .fade-leave-active {
    transition: opacity 1s ease;
  }
</style>
```



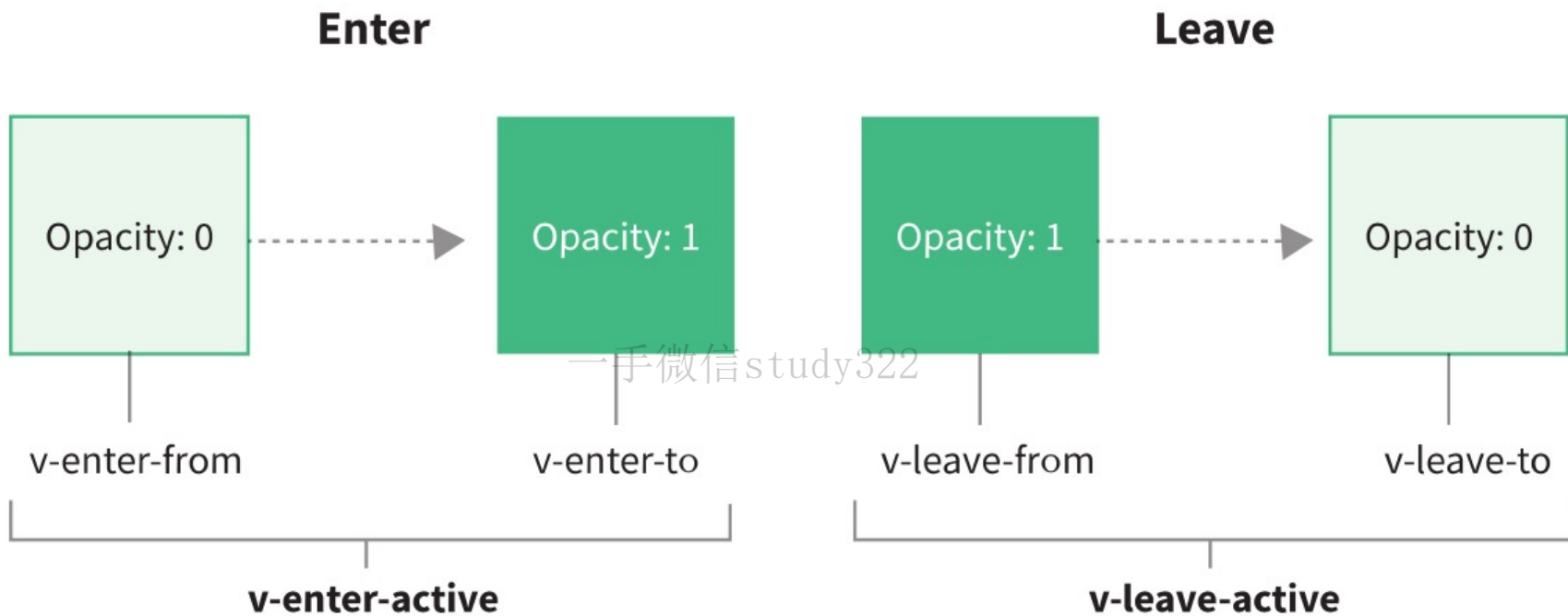
Transition组件的原理

- 我们会发现，Vue自动给h2元素添加了动画，这是为什么呢？
- 当插入或删除包含在 transition 组件中的元素时，Vue 将会做以下处理：
 - 1.自动嗅探目标元素是否应用了CSS过渡或者动画，如果有，那么在恰当的时机添加/删除 CSS类名；
 - 2.如果 transition 组件提供了JavaScript钩子函数，这些钩子函数将在恰当的时机被调用；
 - 3.如果没有找到JavaScript钩子并且也没有检测到CSS过渡/动画，DOM插入、删除操作将会立即执行；
- 那么都会添加或者删除哪些class呢？

过渡动画class

- 我们会发现上面提到了很多个class，事实上Vue就是帮助我们在这些class之间来回切换完成的动画：
- **v-enter-from**：定义进入过渡的开始状态。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**：定义进入过渡生效时的状态。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。这个类可以被用来定义进入过渡的过程时间，延迟和曲线函数。
- **v-enter-to**：定义进入过渡的结束状态。在元素被插入之后下一帧生效（与此同时 v-enter-from 被移除），在过渡/动画完成之后移除。
- **v-leave-from**：定义离开过渡的开始状态。在离开过渡被触发时立刻生效，下一帧被移除。
- **v-leave-active**：定义离开过渡生效时的状态。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。这个类可以被用来定义离开过渡的过程时间，延迟和曲线函数。
- **v-leave-to**：离开过渡的结束状态。在离开过渡被触发之后下一帧生效（与此同时 v-leave-from 被删除），在过渡/动画完成之后移除。

class添加的时机和命名规则



■ class的name命名规则如下：

- 如果我们使用的是一个没有name的transition，那么所有的class是以 v- 作为默认前缀；
- 如果我们添加了一个name属性，比如 `<transition name="why">`，那么所有的class会以 why- 开头；

- 前面我们是通过transition来实现的动画效果，另外我们也可以通过animation来实现。

```
<button @click="show = !show">
  Toogle
</button>
```

```
<transition name="bounce">
```

```
  <h2 v-if="show">
```

但是太阳，他每时每刻都是夕阳也都是旭日。当他熄灭着走下山去收尽苍凉残照之际，正是他在另一面燃烧着爬上山巅布散烈烈朝辉之时。那一天，我也将沉静着走下山去，扶着我的拐杖。有一天，在某一处山洼里，势必会跑上来一个欢蹦的孩子，抱着他的玩具。

```
  </h2>
```

```
</transition>
```

```
.app {
  width: 500px;
  margin: 0 auto;
}
.bounce-enter-active {
  animation: bounce-in 0.5s;
}
.bounce-leave-active {
  animation: bounce-in 0.5s reverse;
}
@keyframes bounce-in {
  0% {
    transform: scale(0)
  }
  50% {
    transform: scale(1.25)
  }
  100% {
    transform: scale(1);
  }
}
```

同时设置过渡和动画

- Vue为了知道过渡的完成，内部是在监听 `transitionend` 或 `animationend`，到底使用哪一个取决于元素应用的CSS规则：
 - 如果我们只是使用了其中的一个，那么Vue能自动识别类型并设置监听；
- 但是如果我们同时使用了过渡和动画呢？
 - 并且在这个情况下可能某一个动画执行结束时，另外一个动画还没有结束；
 - 在这种情况下，我们可以设置 `type` 属性为 `animation` 或者 `transition` 来明确的告知Vue监听的类型；

```
<transition name="why"
  type="transition"
  @after-enter="afterEnter">
  <h2 v-if="show">Hello World</h2>
</transition>
```


显示的指定动画时间

■ 我们也可以显示的来指定过渡的时间，通过 **duration** 属性。

■ **duration**可以设置两种类型的值：

□ **number类型**：同时设置进入和离开的过渡时间；

□ **object类型**：分别设置进入和离开的过渡时间；

一手微信study322

```
<transition name="why"
  type="transition"
  @after-enter="afterEnter"
  :duration="1000">
  <h2 v-if="show">Hello World</h2>
</transition>
```

```
<transition name="why"
  type="transition"
  @after-enter="afterEnter"
  :duration="{enter: 800, leave: 1000}">
  <h2 v-if="show">Hello World</h2>
</transition>
```

过渡的模式mode

- 我们来看当前的动画在两个元素之间切换的时候存在的问题：

Toogle

Hello World你好啊,李银河

一手微信study322

- 我们会发现 Hello World 和 你好啊，李银河是**同时存在**的：
 - 这是因为默认情况下**进入和离开动画**是同时发生的；
 - 如果确实我们希望达到这个的效果，那么是没有问题；
- 但是如果我们**不希望同时执行进入和离开动画**，那么我们需要设置transition的**过渡模式**：
 - **in-out**: 新元素先进行过渡，完成之后当前元素过渡离开；
 - **out-in**: 当前元素先进行过渡，完成之后新元素过渡进入；

动态组件的切换

- 上面的示例同样适用于我们的**动态组件**：

```
<div>
  <div>
    <button @click="show = !show">
      Toogle
    </button>
  </div>

  <transition name="why">
    <component :is="show ? 'home' : 'about'"></component>
  </transition>
</div>
```

一手微信study322

appear初次渲染

- 默认情况下，首次渲染的时候是没有动画的，如果我们希望给他添加上去动画，那么就可以增加另外一个属性 `appear`：

```
<div>
  <button @click="show = !show">
    Toogle
  </button>
</div>

<transition name="why" appear>
  <component :is="show ? 'home' : 'about'"></component>
</transition>
```

一手微信study322



认识animate.css

■ 如果我们手动一个个来编写这些动画，那么效率是比较低的，所以在开发中我们可能会引用一些**第三方库的动画库**，比如**animate.css**。

■ 什么是animate.css呢？

□ **Animate.css** is a library of ready-to-use, cross-browser animations for use in your web projects. Great for emphasis, home pages, sliders, and attention-guiding hints.

□ **Animate.css**是一个已经**准备好的、跨平台的动画**库为我们的web项目，对于强调、主页、滑动、注意力引导非常有用；

■ 如何使用Animate库呢？

□ 第一步：需要**安装animate.css**库；

□ 第二步：导入**animate.css**库的样式；

□ 第三步：使用**animation动画或者animate提供的类**；



自定义过渡class



■ 我们可以通过以下 attribute 来自定义过渡类名：

□ enter-from-class

□ enter-active-class

□ enter-to-class

□ leave-from-class

□ leave-active-class

□ leave-to-class

一手微信study322

■ 他们的优先级高于普通的类名，这对于 **Vue** 的过渡系统和其他第三方 **CSS 动画库**，如 [Animate.css](#). 结合使用十分有用。

animate.css库的使用

■ 安装animate.css :

```
npm install animate.css
```

■ 在main.js中导入animate.css :

```
import "animate.css";
```

一手微信study322

■ 接下来在使用的时候我们有两种用法 :

□ 用法一：直接使用animate库中定义的 keyframes 动画；

□ 用法二：直接使用animate库提供给我们的类；

```
.why-enter-active {  
  animation: flip 1s;  
}  
  
.why-leave-active {  
  animation: flip 1s reverse;  
}
```

```
<transition name="why"  
  enter-active-class="animate__animated animate__lightSpeedInRight"  
  leave-active-class="animate__animated animate__lightSpeedOutRight">  
  <h2 v-if="show">Hello World</h2>  
</transition>
```

■ 某些情况下我们希望通过JavaScript来实现一些动画的效果，这个时候我们可以选择使用gsap库来完成。

■ 什么是gsap呢？

□ GSAP是The GreenSock Animation Platform（GreenSock动画平台）的缩写；

□ 它可以[通过JavaScript](#)为CSS属性、SVG、Canvas等设置动画，并且是浏览器兼容的；

一手微信study322

■ 这个库应该如何使用呢？

□ 第一步：需要[安装gsap](#)库；

□ 第二步：导入[gsap](#)库；

□ 第三步：使用[对应的api](#)即可；

■ 我们可以先安装一下gsap库：

```
npm install gsap
```


- 在使用动画之前，我们先来看一下transition组件给我们提供的JavaScript钩子，这些钩子可以帮助我们监听动画执行到什么阶段了。

```
<transition
  @before-enter="beforeEnter"
  @enter="enter"
  @after-enter="afterEnter"
  @enter-cancelled="enterCancelled"
  @before-leave="beforeLeave"
  @leave="leave"
  @after-leave="afterLeave"
  @leave-cancelled="leaveCancelled"
  :css="false"
>
<!-- ... -->
</transition>
```

```
beforeEnter() {
  console.log("beforeEnter");
},
enter(el, done) {
  console.log("enter");
  done();
},
afterEnter() {
  console.log("afterEnter")
},
enterCancelled() {
  console.log("enterCancelled")
},
```

```
beforeLeave() {
  console.log("beforeLeave")
},
leave(el, done) {
  console.log("leave");
  done();
},
afterLeave() {
  console.log("afterLeave")
},
leaveCancelled() {
  console.log("leaveCancelled")
}
```

- 当我们使用JavaScript来执行过渡动画时，需要**进行 done 回调**，否则它们将会被同步调用，过渡会立即完成。
- 添加 **:css="false"**，也会让 Vue 会**跳过 CSS 的检测**，除了性能略高之外，这可以避免过渡过程中 CSS 规则的影响。

- 那么接下来我们就可以结合gsap库来完成动画效果：

```
<transition
  name="why"
  @enter="enter"
  @leave="leave">
  <h2 v-if="show">Hello World</h2>
</transition>
```

```
enter(el, done) {
  gsap.from(el, {
    scale: 0,
    x: 200,
    onComplete: done
  })
},

leave(el, done) {
  gsap.to(el, {
    scale: 0,
    x: 200,
    onComplete: done
  })
}
```

gsap实现数字变化

- 在一些项目中，我们会见到数字快速变化的动画效果，这个动画可以很容易通过gsap来实现：

```
<div>
  <input type="number" v-model.number="counter" step="100">
  <h2>{{animatedNumber}}</h2>
  <!-- 下面的写法也是可以的 -->
  <h2>{{showNumber.toFixed(0)}}</h2>
</div>
```

```
data() {
  return {
    counter: 0,
    showNumber: 0
  }
},
computed: {
  animatedNumber() {
    return this.showNumber.toFixed(0);
  }
},
watch: {
  counter(newValue) {
    gsap.to(this, {duration: 1, showNumber: newValue})
  }
}
```

认识列表的过渡

■ 目前为止，过渡动画我们只要是**针对单个元素或者组件**的：

- 要么是**单个节点**；

- 要么是**同一时间渲染多个节点中的一个**；

■ 那么如果希望渲染的是一个**列表**，并且**该列表中**添加删除数据也希望有动画执行呢？

- 这个时候我们要使用 `<transition-group>` **组件**来完成；

■ 使用 `<transition-group>` 有如下的特点：

- 默认情况下，它**不会渲染一个元素的包裹器**，但是你可以**指定一个元素并以 tag attribute 进行渲染**；

- **过渡模式不可用**，因为我们不再相互切换特有的元素；

- 内部元素总是**需要提供唯一的 key attribute 值**；

- **CSS 过渡的类将会应用在内部的元素中，而不是这个组/容器本身**；

列表过渡的基本使用

■ 我们来做一个案例：

- 案例是一列数字，可以继续添加或者删除数字；
- 在添加和删除数字的过程中，对添加的或者移除的数字添加动画；



一手微信study322

■ 具体代码查看课堂演练

列表过渡的移动动画

- 在上面的案例中虽然新增的或者删除的节点是有动画的，但是对于哪些其他需要移动的节点是没有动画的：
 - 我们可以通过使用一个新增的 `v-move` 的class来完成动画；
 - 它会 **在元素改变位置的过程** 中应用；
 - 像之前的名字一样，我们可以 **通过name来自定义前缀**；

一手微信study322

添加数字

删除数字

打乱数字

13 1 8 2 3 5 14 11 12 15

10

列表的交错过渡案例

■ 我们来通过gsap的延迟delay属性，做一个交替消失的动画：

- abc
- cba
- dna
- why
- kobe
- james
- curry

```
beforeEnter(el) {  
  el.style.opacity = 0;  
  el.style.height = 0;  
},  
enter(el, done) {  
  gsap.to(el, {  
    opacity: 1,  
    height: "1.6em",  
    delay: el.dataset.index * 0.5,  
    onComplete: done  
  })  
},  
leave(el, done) {  
  gsap.to(el, {  
    opacity: 0,  
    height: 0,  
    delay: el.dataset.index * 0.5,  
    onComplete: done  
  })  
}
```

- 目前我们是使用组件化的方式在开发整个Vue的应用程序，但是**组件和组件之间有时候会存在相同的代码逻辑**，我们希望对**相同的代码逻辑进行抽取**。
- 在Vue2和Vue3中都支持的一种方式就是**使用Mixin来完成**：
 - Mixin提供了一种非常灵活的方式，来**分发Vue组件中的可复用功能**；
 - 一个Mixin对象可以包含**任何组件选项**；一手微信study322
 - 当组件使用Mixin对象时，所有**Mixin对象的选项将被 混合 进入该组件本身的选项中**；

Mixin的基本使用

V Home.vue U X

src > 14_Mixin混入 > pages > V Home.vue > {} "Home.vue"

```
1 <template>
2   <div>
3     <button @click="foo">foo点击</button>
4   </div>
5 </template>
6
7 <script>
8   import sayHelloMixin from '../mixins/sayHello';
9
10  export default {
11    mixins: [sayHelloMixin]
12  }
13 </script>
14
15 <style scoped>
16
17 </style>
```

JS sayHello.js U X

src > 14_Mixin混入 > mixins > JS sayHello.js > [E] default

```
1 const sayHelloMixin = {
2   created() {
3     this.sayHello();
4   },
5   methods: {
6     sayHello() {
7       console.log("Hello Page Component");
8     }
9   }
10 }
11
12 export default sayHelloMixin;
```

Mixin的合并规则

■ 如果Mixin对象中的选项和组件对象中的选项发生了冲突，那么Vue会如何操作呢？

□ 这里分成不同的情况来进行处理；

■ 情况一：如果是data函数的返回值对象

□ 返回值对象默认情况下会进行合并；

□ 如果data返回值对象的属性发生了冲突，那么会保留组件自身的数据；

■ 情况二：如何生命周期钩子函数

□ 生命周期的钩子函数会被合并到数组中，都会被调用；

■ 情况三：值为对象的选项，例如 methods、components 和 directives，将被合并为同一个对象。

□ 比如都有methods选项，并且都定义了方法，那么它们都会生效；

□ 但是如果对象的key相同，那么会取组件对象的键值对；

■ 如果组件中的某些选项，是所有的组件都需要拥有的，那么这个时候我们可以使用**全局的mixin**：

□ 全局的Mixin可以使用 **应用app的方法 mixin** 来完成注册；

□ 一旦注册，那么**全局混入的选项将会影响每一个组件**；

```
const app = createApp(App);  
app.mixin({  
  created() {  
    console.log("global mixin created");  
  }  
})  
app.mount("#app");
```

一手微信study322