Vue.js 监听属性 Vue.js 混入

# Vue.js 过渡 & 动画

本章节我们主要讨论 Vue.js 的过渡效果与动画效果。

## 过渡

Vue 在插入、更新或者移除 DOM 时，提供多种不同方式的应用过渡效果。

Vue 提供了内置的过渡封装组件，该组件用于包裹要实现过渡效果的组件。

### 语法格式

```
<transition name = "nameoftransition">
   <div></div>
</transition>
```

我们可以通过以下实例来理解 Vue 的过渡是如何实现的：

**实例**

```
<div id = "databinding">
<button v-on:click = "show = !show">点我</button>
<transition name = "fade">
<p v-show = "show" v-bind:style = "styleobj">动画实例</p>
</transition>
</div>
<script type = "text/javascript">
var vm = new Vue({
el: '#databinding',
data: {
show:true,
styleobj :{
fontSize:'30px',
color:'red'
}
},
methods : {
}
});
</script>
```

尝试一下 »

实例中通过点击 "点我" 按钮将变量 show 的值从 true 变为 false。如果为 true 显示子元素 p 标签的内容。

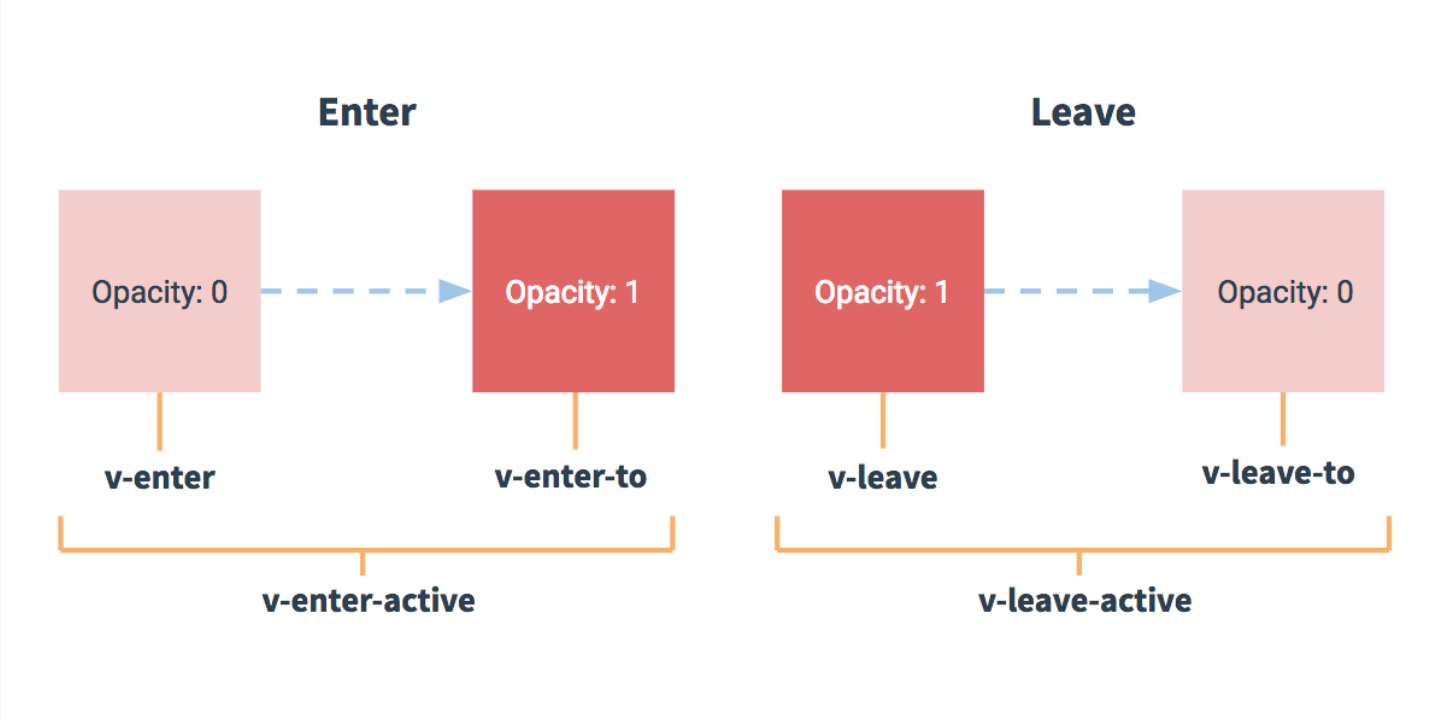
下面这段代码展示了 transition 标签包裹了 p 标签：

```
<transition name = "fade">
    <p v-show = "show" v-bind:style = "styleobj">动画实例</p>
```

```
</transition>
```

过渡其实就是一个淡入淡出的效果。Vue在元素显示与隐藏的过渡中，提供了 6 个 class 来切换：

- `v-enter`：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`: **2.1.8版及以上** 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` 被移除)，在过渡/动画完成之后移除。
- `v-leave`: 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`: **2.1.8版及以上** 定义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` 被删除)，在过渡/动画完成之后移除。




对于这些在过渡中切换的类名来说，如果你使用一个没有名字的 `<transition>`，则 `v-` 是这些类名的默认前缀。如果你使用了 `<transition name="my-transition">`，那么 `v-enter` 会替换为 `my-transition-enter`。

`v-enter-active` 和 `v-leave-active` 可以控制进入/离开过渡的不同的缓和曲线，在下面章节会有个示例说明。

## CSS 过渡

通常我们都使用 CSS 过渡来实现效果。

如下实例：

**实例**

```
<div id = "databinding">
<button v-on:click = "show = !show">点我</button>
<transition name="slide-fade">
<p v-if="show">hello</p>
</transition>
</div>
<script type = "text/javascript">
new Vue({
el: '#databinding',
data: {
show: true
}
})
</script>
```

尝试一下 »

## CSS 动画

CSS 动画用法类似 CSS 过渡，但是在动画中 v-enter 类名在节点插入 DOM 后不会立即删除，而是在 animationend 事件触发时删除。

**实例**

```
<div id = "databinding">
<button v-on:click = "show = !show">点我</button>
<transition name="bounce">
<p v-if="show">菜鸟教程 -- 学的不仅是技术，更是梦想！！！ </p>
</transition>
</div>
<script type = "text/javascript">
new Vue({
el: '#databinding',
data: {
show: true
}
})
</script>
```

尝试一下 »

## 自定义过渡的类名

我们可以通过以下特性来自定义过渡类名：

- enter-class
- enter-active-class
- enter-to-class (2.1.8+)
- leave-class

- leave-active-class
- leave-to-class (2.1.8+)

自定义过渡的类名优先级高于普通的类名，这样就能很好的与第三方（如：animate.css）的动画库结合使用。

**实例**

```
<div id = "databinding">
<button v-on:click = "show = !show">点我</button>
<transition
name="custom-classes-transition"
enter-active-class="animated tada"
leave-active-class="animated bounceOutRight"
>
<p v-if="show">菜鸟教程 -- 学的不仅是技术，更是梦想！！！</p>
</transition>
</div>
<script type = "text/javascript">
new Vue({
el: '#databinding',
data: {
show: true
}
})
</script>
```

尝试一下 »

## 同时使用过渡和动画

Vue 为了知道过渡的完成，必须设置相应的事件监听器。它可以是 transitionend 或 animationend ，这取决于给元素应用的 CSS 规则。如果你使用其中任何一种，Vue 能自动识别类型并设置监听。

但是，在一些场景中，你需要给同一个元素同时设置两种过渡动效，比如 animation 很快的被触发并完成了，而 transition 效果还没结束。在这种情况中，你就需要使用 type 特性并设置 animation 或 transition 来明确声明你需要 Vue 监听的类型。

## 显性的过渡持续时间

在很多情况下，Vue 可以自动得出过渡效果的完成时机。默认情况下，Vue 会等待其在过渡效果的根元素的第一个 transitionend 或 animationend 事件。然而也可以不这样设定——比如，我们可以拥有一个精心编排的一系列过渡效果，其中一些嵌套的内部元素相比于过渡效果的根元素有延迟的或更长的过渡效果。

在这种情况下你可以用 <transition> 组件上的 duration 属性定制一个显性的过渡持续时间 (以毫秒计)：

```
<transition :duration="1000">...</transition>
```

你也可以定制进入和移出的持续时间：

```
<transition :duration="{ enter: 500, leave: 800 }">...</transition>
```

## JavaScript 钩子

可以在属性中声明 JavaScript 钩子:

**HTML 代码：**

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

**JavaScript 代码：**

```
// ...
methods: {
// --------
// 进入中
// --------
beforeEnter: function (el) {
// ...
},
// 此回调函数是可选项的设置
// 与 CSS 结合时使用
enter: function (el, done) {
// ...
done()
},
afterEnter: function (el) {
// ...
},
enterCancelled: function (el) {
// ...
},
// --------
// 离开时
// --------
beforeLeave: function (el) {
// ...
},
// 此回调函数是可选项的设置
// 与 CSS 结合时使用
leave: function (el, done) {
// ...
done()
},
```

```
afterLeave: function (el) {
// ...
},
// leaveCancelled 只用于 v-show 中
leaveCancelled: function (el) {
// ...
}
}
```

这些钩子函数可以结合 CSS transitions/animations 使用，也可以单独使用。

当只用 JavaScript 过渡的时候，**在 `enter` 和 `leave` 中必须使用 `done` 进行回调。**否则，它们将被同步调用，过渡会立即完成。

推荐对于仅使用 JavaScript 过渡的元素添加 `v-bind:css="false"`，Vue 会跳过 CSS 的检测。这也可以避免过渡过程中 CSS 的影响。

一个使用 Velocity.js 的简单例子：

**实例**

```
<div id = "databinding">
<button v-on:click = "show = !show">点我</button>
<transition
v-on:before-enter="beforeEnter"
v-on:enter="enter"
v-on:leave="leave"
v-bind:css="false"
>
<p v-if="show">菜鸟教程 -- 学的不仅是技术，更是梦想！！！ </p>
</transition>
</div>
<script type = "text/javascript">
new Vue({
el: '#databinding',
data: {
show: false
},
methods: {
beforeEnter: function (el) {
el.style.opacity = 0
el.style.transformOrigin = 'left'
},
enter: function (el, done) {
Velocity(el, { opacity: 1, fontSize: '1.4em' }, { duration: 300 })
Velocity(el, { fontSize: '1em' }, { complete: done })
},
leave: function (el, done) {
Velocity(el, { translateX: '15px', rotateZ: '50deg' }, { duration: 600 })
Velocity(el, { rotateZ: '100deg' }, { loop: 2 })
Velocity(el, {
rotateZ: '45deg',
translateY: '30px',
translateX: '30px',
opacity: 0
```

```
}, { complete: done })
}
}
})
</script>
```

尝试一下 »

## 初始渲染的过渡

可以通过 `appear` 特性设置节点在初始渲染的过渡

```
<transition appear>
  <!-- ... -->
</transition>
```

这里默认和进入/离开过渡一样，同样也可以自定义 CSS 类名。

```
<transition
  appear
  appear-class="custom-appear-class"
  appear-to-class="custom-appear-to-class" (2.1.8+)
  appear-active-class="custom-appear-active-class"
>
  <!-- ... -->
</transition>
```

自定义 JavaScript 钩子：

```
<transition
  appear
  v-on:before-appear="customBeforeAppearHook"
  v-on:appear="customAppearHook"
  v-on:after-appear="customAfterAppearHook"
  v-on:appear-cancelled="customAppearCancelledHook"
>
  <!-- ... -->
</transition>
```

### 多个元素的过渡

我们可以设置多个元素的过渡，一般列表与描述：

需要注意的是当有相同标签名的元素切换时，需要通过 `key` 特性设置唯一的值来标记以让 Vue 区分它们，否则 Vue 为了效率只会替换相同标签内部的内容。

```
<transition>
  <table v-if="items.length > 0">
    <!-- ... -->
  </table>
  <p v-else>抱歉，没有找到您查找的内容。</p>
</transition>
```

如下实例：

```
<transition>
  <button v-if="isEditing" key="save">
    Save
  </button>
  <button v-else key="edit">
    Edit
  </button>
</transition>
```

在一些场景中，也可以通过给同一个元素的 `key` 特性设置不同的状态来代替 `v-if` 和 `v-else`，上面的例子可以重写为：

```
<transition>
  <button v-bind:key="isEditing">
    {{ isEditing ? 'Save' : 'Edit' }}
  </button>
</transition>
```

使用多个 `v-if` 的多个元素的过渡可以重写为绑定了动态属性的单个元素过渡。例如：

```
<transition>
  <button v-if="docState === 'saved'" key="saved">
    Edit
  </button>
  <button v-if="docState === 'edited'" key="edited">
    Save
  </button>
  <button v-if="docState === 'editing'" key="editing">
    Cancel
  </button>
</transition>
```

可以重写为：

```
<transition>
  <button v-bind:key="docState">
```

```
    {{ buttonMessage }}
  </button>
</transition>

// ...
computed: {
  buttonMessage: function () {
    switch (this.docState) {
      case 'saved': return 'Edit'
      case 'edited': return 'Save'
      case 'editing': return 'Cancel'
    }
  }
}
```

Vue.js 监听属性 Vue.js 混入

点我分享笔记