# **进入/离开 & 列表过渡**

这里是一个典型的例子：

|  |
| --- |
| <div id="demo">  <button v-on:click="show = !show">  Toggle  </button>  <transition name="fade">  <p v-if="show">hello</p>  </transition> </div> |

|  |
| --- |
| new Vue({  el: '#demo',  data: {  show: true  } }) |

|  |
| --- |
| .fade-enter-active, .fade-leave-active {  transition: opacity .5s; } .fade-enter, .fade-leave-to /\* .fade-leave-active below version 2.1.8 \*/ {  opacity: 0; } |

### **[过渡的类名](https://cn.vuejs.org/v2/guide/transitions.html" \l "%E8%BF%87%E6%B8%A1%E7%9A%84%E7%B1%BB%E5%90%8D" \o "过渡的类名)**

在进入/离开的过渡中，会有 6 个 class 切换。

v-enter：定义进入过渡的开始状态。在元素被插入时生效，在下一个帧移除。

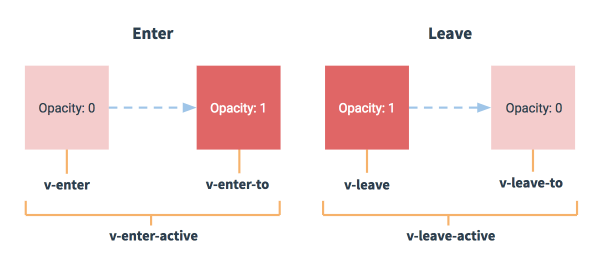
v-enter-active：定义过渡的状态。在元素整个过渡过程中作用，在元素被插入时生效，在 transition/animation 完成之后移除。这个类可以被用来定义过渡的过程时间，延迟和曲线函数。

v-enter-to: ****2.1.8版及以上**** 定义进入过渡的结束状态。在元素被插入一帧后生效 (与此同时 v-enter 被删除)，在 transition/animation 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生效，在下一个帧移除。

v-leave-active：定义过渡的状态。在元素整个过渡过程中作用，在离开过渡被触发后立即生效，在 transition/animation 完成之后移除。这个类可以被用来定义过渡的过程时间，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。在离开过渡被触发一帧后生效 (与此同时 v-leave 被删除)，在 transition/animation 完成之后移除。



Vuex

1. 对象state

// 仓库  
const state={  
 **singer**:{}  
}  
export default state;

1. actions
2. getters

export const singer= state => state.**singer**;

1. mutations

// 字符串常量  
export const SET\_SINGER='SET\_SINGER';

import \* as types from './mutations-type.js';  
// 修改state  
const mutations={  
 // 函数的一种写法,第一个参数是仓库state，第二个参数是我们传递进来的值  
 [types.SET\_SINGER](state,singer){  
 state.**singer**=singer;  
 }  
}  
export default mutations;

1. Index

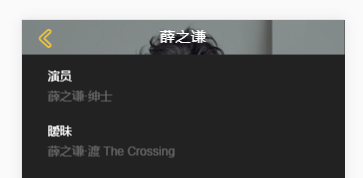
import Vue from 'vue';  
import Vuex from 'vuex';  
import \* as actions from './actions.js';  
import \* as getters from './getters.js';  
import state from './state.js';  
import mutations from './mutations.js';  
// 调试用  
import createLogger from 'vuex/dist/logger'  
Vue.use(Vuex)  
// 调试用  
const debug = process.**env**.**NODE\_ENV** !== 'production'  
export default new Vuex.Store({  
 actions,  
 getters,  
 state,  
 mutations,  
 **strict**: debug,  
 **plugins**: debug ? [createLogger()] : []  
});

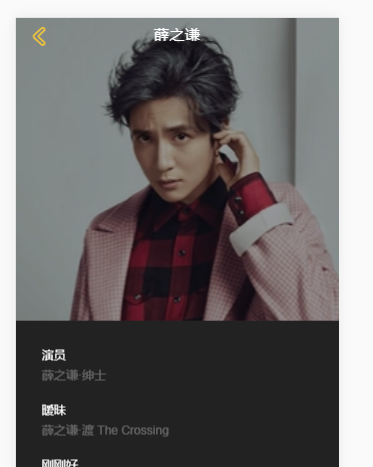
1. 组件内使用

// 导出修改store的行为方法，必须在methods中导出，即可在组件内使用，this.setSinger(xxx),不用该方法导出相当于this.$store.commit('SET\_SINGER');  
...mapMutations({  
 **setSinger**:'SET\_SINGER'  
})

// 导出获取仓库state的方法,名称（字符串）要与与getters中的函数名一致，必须在computed中导出，即可在组件内使用，this.singer,不用该方法导出相当于return this.$store.getters.singer或者this.$store.state.singer;  
...mapGetters([  
 'singer'  
])

Scroll组件上拉时的效果（上拉可以时图片也随着变小，scroll在顶端下拉时图片可以放大）





<!--用于跟随歌曲列表滚动时滚动-->  
<!--这里不能该scroll组件的高度是因为滚动时又同时改变高度会出现问题，所以我们用一个背景层‘.bg-layer’利用z-index的层级关系使他在scroll下面，背景图上面，scroll滚动时把背景层同时移动-->  
<div class="bg-layer" ref="**layer**"></div>  
<scroll :data="**songs**" class="list" ref="list" :probe-type="**probeType**" :listen-scroll="**listenScroll**" @scroll="scroll">  
 <div class="song-list-wrapper">  
 <song-list :songs="**songs**"></song-list>  
 </div>  
</scroll>

mounted(){  
 // 获取图片的高度  
 this.**imageHeight**=this.$refs.image.clientHeight;  
  
 // 元素this.$refs.layer往上移动的最小距离，往上移动是负数，下移是正数  
 this.**minTransalteY**=-this.**imageHeight**+RESERVED\_HEIGHT;  
  
 // this.$refs.list得到的是vue组件，this.$refs.list.$el得到的是vue组件下的子元素。  
 //1、clientHeight网页(内容)可见区域高：document.body.clientHeight 即页面浏览器中可以看到内容的这个区域的高度，一般是最后一个工具条以下到状态栏以上的这个区域，与页面内容无关。注意：IE、Opera 认为 scrollHeight 是网页内容实际高度，可以小于 clientHeight。2、offsetHeight网页可见区域高：document.body.offsetHeight (包括边线的宽)  
 this.$refs.list.**$el**.**style**.**top**=`${this.**imageHeight**}px`;  
},

**watch**:{  
 scrollY(newY){  
 // 层级的初始值  
 var zIndex=0;  
 // 放大或缩小的初始值  
 var scale=1;  
 // 放大或缩小的百分比  
 var percent=Math.abs(newY/this.**imageHeight**);  
 if(newY>0){  
 // 放大的倍数  
 scale=1+percent;  
 // 让放大后的图片可以遮住scroll组件  
 zIndex=10;  
 }  
  
 // 元素this.$refs.layer往上滚动的距离，Math.max是取出2个数中的最大数，this.minTransalteY,newY都是负数  
 var translateY=Math.max(this.**minTransalteY**,newY)  
 this.$refs.**layer**.**style**['transform']=`translate3d(0,${translateY}px,0)`;  
  
 // 当滚动的距离大于最小的移动距离时，newY和this.minTransalteY都是负数，即，背景层this.$refs.layer已经移动到最顶端时  
 if(newY<this.**minTransalteY**){  
 // 由于我们在最开始的样式是height为0，paddingTop是70%，所以才能达到上下3:7的布局，这事我们要把paddingTop设为0，height设为返回按钮元素的高度，把zIndex设为10，才能遮挡住文字  
 this.$refs.image.style.paddingTop='0';  
 this.$refs.image.style.height=`${RESERVED\_HEIGHT}px`;  
 zIndex=10;  
 }else{  
 // 背景层this.$refs.layer还没有移动到最顶端时，恢复原来的样式  
 this.$refs.image.style.paddingTop='70%';  
 this.$refs.image.style.height='0';  
 }  
 this.$refs.image.style['transform'] = `scale(${scale})`;  
 this.$refs.image.style.zIndex=zIndex;  
 }  
},