# Lab: Vue Animations

## Fading Text

Make an animation which fades in and fades out a paragraph when clicking a button.

### Code

|  |  |  |
| --- | --- | --- |
| **HTML** | **JS** | **CSS** |
| <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 {  opacity: 0;  } |

## Slide-Fade

Make an animation as in the first problem, but this time the text also slides (together with fading).

### Code

|  |  |  |
| --- | --- | --- |
| **HTML** | **JS** | **CSS** |
| <div id="example-1">  <button @click="show = !show">  Toggle render  </button>  <transition name="slide-fade">  <p v-if="show">hello</p>  </transition>  </div> | new Vue({  el: '#example-1',  data: {  show: true  }  }) | .slide-fade-enter-active {  transition: all .3s ease;  }  .slide-fade-leave-active {  transition: all .8s cubic-bezier(1.0, 0.5, 0.8, 1.0);  }  .slide-fade-enter, .slide-fade-leave-to {  transform: translateX(10px);  opacity: 0;  } |

## Text Bounce

Make an animation which bounces in and bounces out the text.

### Code

|  |  |  |
| --- | --- | --- |
| **HTML** | **JS** | **CSS** |
| <div id="example-2">  <button @click="show = !show">Toggle show</button>  <transition name="bounce">  <p v-if="show">hello</p>  </transition>  </div> | new Vue({  el: '#example-2',  data: {  show: true  }  }) | .bounce-enter-active {  animation: bounce-in .5s;  }  .bounce-leave-active {  animation: bounce-in .5s reverse;  }  @keyframes bounce-in {  0% {  transform: scale(0);  }  50% {  transform: scale(1.5);  }  100% {  transform: scale(1);  }  } |

## Button Transition

Make a button transit to another one when clicked.

### Example buttons

|  |
| --- |
| **HTML** |
| <div id="on-off">  <transition name = "fade">  <button v-if="isOn" key="off">  Off  </button>  <button v-else key="on">  On  </button>  </transition>  </div> |

## Button Transition with Transition Modes

Using the button from the last problem, make a transition with the **on-off** or **off-on** transition modes.

## Transition Between Components

You will be given two components. You have to make an element (button, checkbox, etc.) which on click changes from one component to the other with a transition.

### Code

|  |  |
| --- | --- |
| **HTML** | **JS** |
| <transition name="component-fade" mode="out-in">  <component v-bind:is="view"> </component>  </transition> | new Vue({  el: '#transition-components-demo',  data: {  view: 'v-a'  },  components: {  'v-a': {  template: '<div>Component A</div>'  },  'v-b': {  template: '<div>Component B</div>'  }  }  }) |

## List Entering/Leaving Transitions

You will be given a list with numbers and **Add** and **Remove** buttons. The **Add** button inserts a new number at a random position and the **Remove** button removes a random number from the list. Your job is to animate the entering and leaving of the numbers in the list.

### Code

|  |  |
| --- | --- |
| **HTML** | **JS** |
| <div id="list-demo">  <button v-on:click="add">Add</button>  <button v-on:click="remove">Remove</button>  <transition-group name="list" tag="p">  <span v-for="item in items" v-bind:key="item" class="list-item">  {{ item }}  </span>  </transition-group>  </div> | new Vue({  el: '#list-demo',  data: {  items: [1,2,3,4,5,6,7,8,9],  nextNum: 10  },  methods: {  randomIndex: function () {  return Math.floor(Math.random() \* this.items.length)  },  add: function () {  this.items.splice(this.randomIndex(), 0, this.nextNum++)  },  remove: function () {  this.items.splice(this.randomIndex(), 1)  },  }  })}) |

## Shuffle Numbers in List

You are given a list of numbers and a **Shuffle** button which shuffles them in random places. Your task is to animate the movement of the numbers when clicking the **Shuffle** button.

### Code

|  |  |
| --- | --- |
| **HTML** | **JS** |
| <div id="flip-list-demo" class="demo">  <button v-on:click="shuffle">Shuffle</button>  <transition-group name="flip-list" tag="ul">  <li v-for="item in items" v-bind:key="item">  {{ item }}  </li>  </transition-group>  </div> | new Vue({  el: '#flip-list-demo',  data: {  items: [1,2,3,4,5,6,7,8,9]  },  methods: {  shuffle: function () {  this.items = \_.shuffle(this.items)  }  }  }) |

## Find Elements in List

You will be given a list and a search bar. The logic for the search bar is already written. You have to animate the elements of the list going in or out when a part of the element's text is written.

|  |  |
| --- | --- |
| **HTML** | **JS** |
| <script src="https://cdnjs.cloudflare.com/ajax/ libs/velocity/1.2.3/velocity.min.js"> </script>  <div id="staggered-list-demo">  <input v-model="query">  <transition-group  name="staggered-fade"  tag="ul"  v-bind:css="false"  v-on:before-enter="beforeEnter"  v-on:enter="enter"  v-on:leave="leave"  >  <li  v-for="(item, index) in computedList"  v-bind:key="item.msg"  v-bind:data-index="index"  >{{ item.msg }}</li>  </transition-group>  </div> | new Vue({  el: '#staggered-list-demo',  data: {  query: '',  list: [  { msg: 'Bruce Lee' },  { msg: 'Jackie Chan' },  { msg: 'Chuck Norris' },  { msg: 'Jet Li' },  { msg: 'Kung Fury' }  ]  },  computed: {  computedList: function () {  var vm = this  return this.list.filter(function (item) {  return item.msg.toLowerCase() .indexOf(vm.query.toLowerCase()) !== -1  })  }  },  methods: {  beforeEnter: function (el) {  el.style.opacity = 0  el.style.height = 0  },  enter: function (el, done) {  var delay = el.dataset.index \* 150  setTimeout(function () {  Velocity(  el,  { opacity: 1, height: '1.6em' },  { complete: done }  )  }, delay)  },  leave: function (el, done) {  var delay = el.dataset.index \* 150  setTimeout(function () {  Velocity(  el,  { opacity: 0, height: 0 },  { complete: done }  )  }, delay)  }  }  }) |