Link to job listing: https://www.glassdoor.com/job-listing/web-developer-multiply-JV_IC1131270_KO0,13_KE14,22.htm?jl=2523938938&ctt=1512446392399

Link to Vue.js Documentation Source: https://vuejs.org/v2/guide/transitions.html

Link to GitHub Repository: https://github.com/weixianlow/CS4830_Exploration4_Fall2017

Link to Instance: http://cs4830.weixianlow.me/exploration4

Journal:

Installation:

To install the Vue.js framework, it's as simple with adding a script tag containing Vue.js' CDN into the html page you would like Vue.js to be enabled on.

Setting up:

To use Vue.js, you would be required to create an instance where it's tied to a div element on the html page. You can do so by adding an id tag to the element you want the vue.js instance tied to:

```
<div id="cssAnimation">
```

After tying the div element to the vue.js instance, you would then need to create a new instance using the following format:

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue',
    show: true
  }
})
```

By doing that, the element would have given Vue.js access on the DOM. Since Vue.js is a reactive framework, it will work as similar as AngularJS.

Areas Explored (Animations and Transitions):

There are a couple of animation/transition I have explored on this exploration, as you could see with the link to my instance, all three would have different type of animation/transition tied to a button event.

To begin setting up in this demo the following CSS code is used.

```
<style>
  .fade-enter-active, .fade-leave-active {
    transition: opacity .5s
  }
  .fade-enter, .fade-leave-to /* .fade-leave-active below version 2.1.8 */ {
    opacity: 0
  }

  .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);
    }
  }
  .component-fade-enter-active, .component-fade-leave-active {
    transition: opacity .3s ease;
  }
  .component-fade-enter, .component-fade-leave-to
  /* .component-fade-leave-active below version 2.1.8 */ {
    opacity: 0;
  }
  .list-item {
    display: inline-block;
    margin-right: 10px;
  }
  .list-enter-active, .list-leave-active {
    transition: all 1s;
  }
  .list-enter, .list-leave-to /* .list-leave-active below version 2.1.8 */ {
    opacity: 0;
    transform: translateY(30px);
  }
</style>
```

After creating the necessary CSS codes, we will then dive into the html and js code for the examples:

1) Example 1
   a. In this example, this is the code needed to handle a fade in and out button toggle.
   b.
```
var app1 = new Vue({
  el: '#cssTransition',
  data: {
    show: true
  }
})
```
   c.
```
<div id="cssTransition">
  <h3>CSS Transitions</h3>
  <button v-on:click="show = !show">Toggle Transition</button>
  <br>
  <transition name="fade">
    <p v-if="show">This will fade away when the button is clicked.</p>
  </transition>
</div>
```
   d. In this code, the button will basically trigger a change of value in the Vue instance code, by changing show from true to its opposite value.
2) Example 2
   a. In this example this is the code needed to handle a specific fade in and out animation.
   b.
```
var app2 = new Vue({
  el: '#cssAnimation',
  data: {
    show: true
  }
})
```
   c.
```
<div id="cssAnimation">
  <h3>CSS Animations</h3>
  <button @click="show = !show">Toggle animation</button>
  <transition name="bounce">
    <p v-if="show">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris facilisis enim libero, at lacinia diam fermentum id. Pellentesque habitant morbi tristique senectus et netus.</p>
  </transition>
</div>
```
   d. In this code, the button will perform exactly the same as the previous example, but the only difference is that the transition has a specific class type, which is bounce.
3) Example 3
   a. In this example, this is the code needed to handle an edit to a list on the DOM by adding In some animation.

b.
```
var app3 = 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)
    },
  }
})
```

c.
```
<div id="list-demo">
  <h3>List entering/leaving demo</h3>
  <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>
```

d. In this code, the list is added and removed using the CSS code listed above and also the methods in the Javascript code so that random numbers can be added in.