**Section 14 – Animations And Transitions**

Chapter 188 – Animation Basics And CSS Transitions

1. Move to the left use this code in CSS.

.animate {

  transform: translateX(-150px);

}

If you want to move smoothly, you can use this CSS code.

.block {

  width: 8rem;

  height: 8rem;

  background-color: #290033;

  margin-bottom: 2rem;

  transition: transform 0.3s ease-out;

}

Chapter 189 – Understanding CSS Animations

1. We can add keyframes using CSS like below.

.block {

  width: 8rem;

  height: 8rem;

  background-color: #290033;

  margin-bottom: 2rem;

  /\* transition: transform 0.3s ease-out; \*/

}

Forwards means the box will move forward, 0.3s move slowly.

.animate {

  /\* transform: translateX(-150px); \*/

  animation: slide-fade 0.3s ease-out forwards;

}

Scale to make the box bigger, translate for position, 0% - 100% for the time.

@keyframes slide-fade {

  0% {

    transform: translate(0) scale(1);

  }

  70% {

    transform: translate(-120px) scale(1.1);

  }

  100% {

    transform: translate(-150px) scale(1);

  }

}

Chapter 190 – Why Is “Just CSS” Not Enough ?

1. Make a message dialog pop up slowly.

dialog {

  position: fixed;

  top: 30vh;

  width: 30rem;

  left: calc(50% - 15rem);

  margin: 0;

  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.26);

  border-radius: 12px;

  padding: 1rem;

  background-color: white;

  z-index: 100;

  border: none;

  animation: modal 0.3s ease-out forwards;

}

@keyframes modal {

  /\* If only 0% and 100% we can use from to \*/

  from {

    opacity: 0;

    transform: translateY(-50px) scale(0.9);

  }

  to {

    opacity: 1;

    transform: translateY(0) scale(1);

  }

}

Chapter 191 – Playing CSS Animations With Vue’s Help

1. Process Mounted
2. CSS adds a enter from class, enter from class, enter active class, and enter to class.
3. Vue finds out how long that process should take by analyzing these special CSS classes and looking for transitions and animations inside of them to then read the duration from them.
4. Vue will analyze special CSS code for these special CSS class names.
5. Process Unmounted
6. Vue adds leave from class, leave active class (at the same time), and leave to class and allows you to animate the removal of elements.
7. Vue will analyze the CSS code you write for these classes and read the duration of the transitions.
8. Remove the element from the DOM once that duration is over.
9. Execution

<div class="container">

    <transition>

      <p v-if="paraIsVisible">This is only sometimes visible ...</p>

    </transition>

    <button @click="toggleParagraph">Toggle Paragraph</button>

</div>

dialog {

  position: fixed;

  top: 30vh;

  width: 30rem;

  left: calc(50% - 15rem);

  margin: 0;

  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.26);

  border-radius: 12px;

  padding: 1rem;

  background-color: white;

  z-index: 100;

  border: none;

  animation: modal 0.3s ease-out forwards;

}

@keyframes modal {

  /\* If only 0% and 100% we can use from to \*/

  from {

    opacity: 0;

    transform: translateY(-50px) scale(0.9);

  }

  to {

    opacity: 1;

    transform: translateY(0) scale(1);

  }

}

Chapter 192 – Using The Transition Component

1. Based on Chapter 191, there are 3 times configuration, from, active, and to, so we can make the CSS code like below. This code can show the text like an animation fade in and fade out.

.v-enter-from {

  opacity: 0;

  transform: translateY(-30px);

}

.v-enter-active {

  transition: all 0.3s ease-out;

}

.v-enter-to {

  opacity: 1;

  transform: translateY(0);

}

.v-leave-from {

  opacity: 1;

  transform: translateY(0);

}

.v-leave-active {

  transition: all 0.3s ease-in;

}

.v-leave-to {

  opacity: 0;

  transform: translateY(30px);

}

Chapter 193 – CSS Animations With The Transition Component

1 .We can make and animation not using from to but just use active class.

@keyframes slide-scale {

  0% {

    transform: translate(0) scale(1);

  }

  70% {

    transform: translate(-120px) scale(1.1);

  }

  100% {

    transform: translate(-150px) scale(1);

  }

}

.v-enter-active {

  /\* transition: all 0.3s ease-out; \*/

  animation: slide-scale 0.3s ease-out;

}

.v-leave-active {

  /\* transition: all 0.3s ease-in; \*/

  animation: slide-scale 0.3s ease-out;

}

Chapter 194 – Using Custom CSS Class Names

1. We can use name=”” on transition tag or just input CSS inline style on transition tag like enter-to-class=”some-class” enter-acive-class=”…” etc. If we use name=”” we need to change all the CSS style using para-…, the result like below.

<transition name="para">

      <p v-if="paraIsVisible">This is only sometimes visible ...</p>

</transition>

.para-enter-active {

  animation: slide-scale 0.3s ease-out;

}

Chapter 195 – Example: Animating A Modal

1. Transition is one direct child element, just one base modal. We can use open in HTML tag like below.

<template>

  <div v-if="open" class="backdrop" @click="$emit('close')"></div>

  <transition name="modal">

    <dialog open v-if="open">

      <slot></slot>

    </dialog>

  </transition>

</template>

<script>

export default {

  props: ['open'],

  emits: ['close'],

};

</script>

.modal-enter-active {

  /\* transition: all 0.3s ease-out; \*/

  animation: modal 0.3s ease-out;

}

.modal-leave-active {

  /\* transition: all 0.3s ease-out; \*/

  animation: modal 0.3s ease-in reverse;

}

@keyframes modal {

  /\* If only 0% and 100% we can use from to \*/

  from {

    opacity: 0;

    transform: translateY(-50px) scale(0.9);

  }

  to {

    opacity: 1;

    transform: translateY(0) scale(1);

  }

}

Chapter 196 – Transitioning Between Multiple Element

1. This picture means we just emphasized that it would be fine to have multiple elements if we guarantee that only one is shown at the same time. Vue does not guarantee us that only one button is added. We can use v-else to make it clear in this code below.



<div class="container">

    <transition name="fade-button" mode="out-in">

      <button @click="showUsers" v-if="!usersAreVisible">Show Users</button>

      <button @click="hideUsers" v-else>Hide Users</button>

    </transition>

</div>

Chapter 197 – Using Transition Events

1. Using beforeEnter and beforeLeave. Animation: before enter, enter, and after enter.

<transition

      name="para"

      @before-enter="beforeEnter"

      @enter="enter"

      @after-enter="afterEnter"

      @before-leave="beforeLeave"

      @leave="leave"

      @after-leave="afterLeave">

      <p v-if="paraIsVisible">This is only sometimes visible ...</p>

</transition>

methods: {

    beforeEnter(el) {

      console.log('beforeEnter');

      // el to show the html tag that will be shown

      console.log(el);

    },

    enter(el) {

      console.log('enter');

      console.log(el);

    },

    afterEnter(el) {

      console.log('afterEnter');

      console.log(el);

    },

    beforeLeave(el) {

      console.log('beforeLeave');

      console.log(el);

    },

    leave(el) {

      console.log('leave');

      console.log(el);

    },

    afterLeave(el) {

      console.log('afterLeave');

      console.log(el);

    },

}

Chapter 198 – Building JavaScript Transitions (instead of CSS)

1. Make our own animation

enter(el, done) {

      // done is needed, it is used to make a sign that you have finish

      // calling a set of animation, you dont need to print again (1 time only)

      console.log('enter');

      console.log(el);

      let round = 1;

      const interval = setInterval(function() {

        el.syle.opacity = round \* 0.01;

        round++;

        if (round > 100) {

          clearInterval(interval);

          done();

        }

      }, 20);

    },

leave(el, done) {

      console.log('leave');

      console.log(el);

      let round = 1;

      const interval = setInterval(function() {

        el.style.opacity = 1 - round \* 0.01;

        round++;

        if (round > 100) {

          clearInterval(interval);

          done();

        }

      }, 20)

    },

1. Make it smoother and handling the error.

enter(el, done) {

      // done is needed, it is used to make a sign that you have finish

      // calling a set of animation, you dont need to print again (1 time only)

      console.log('enter');

      console.log(el);

      let round = 1;

      this.enterInterval = setInterval(() =>  {

        el.style.opacity = round \* 0.01;

        round++;

        if (round > 100) {

          clearInterval(this.enterInterval);

          done();

        }

      }, 20);

    },

leave(el, done) {

      console.log('leave');

      console.log(el);

      let round = 1;

      this.leaveInterval = setInterval(() =>  {

        el.style.opacity = 1 - round \* 0.01;

        round++;

        if (round > 100) {

          clearInterval(this.leaveInterval);

          done();

        }

      }, 20)

    },

Chapter 199 – Disabling CSS Transitions

1. We can not using CSS Transitions, the entire transition will be controlled through JavaScript, it skips searching for the respect of CSS classes. It will improve because it doesn’t need to do this unnecessary step. Using :css=”false” if you really don’t use CSS code for this transition (faster).

<transition

      :css="false"

      @before-enter="beforeEnter"

      @enter="enter"

      @after-enter="afterEnter"

      @before-leave="beforeLeave"

      @leave="leave"

      @after-leave="afterLeave"

      @enter-cancelled="enterCancelled"

      @leave-cancelled="leaveCancelled">

      <p v-if="paraIsVisible">This is only sometimes visible ...</p>

    </transition>

Chapter 200 – Getting Started With Animated Lists

1. We make a list of user name like below.

<template>

    <ul>

        <li v-for="user in users" :key="user" @click="removeUser(user)">

{{ user }}

</li>

    </ul>

    <div>

        <input type="text" v-model="userNameInput" ref="userNameInput" />

        <button @click="addUser">Add User</button>

    </div>

</template>

Chapter 201 – Animating Lists With “Transition – Group”

1. Transition is only for one element, how if we want to make a transition for element in function for ? We can use transition-group. Transition group will render an element to the DOM in the end, and you can control which element it is by giving it detect prop (add tag=”ul”).

<transition-group tag="ul">

    <li v-for="user in users" :key="user" @click="removeUser(user)"> {{ user }}</li>

</transition-group>

1. Add some CSS code to add a new username to the list.

.user-list-enter-from {

    opacity: 0;

    transform: translateX(-30px);

}

.user-list-enter-active {

    transition: all 1s ease-out;

}

.user-list-enter-to {

    opacity: 1;

    transform: translateX(0);

}

And the HTML code like below.

<ul>

<transition-group tag="ul" name="user-list">

<li v-for="user in users" :key="user" @click="removeUser(user)">{{ user }}</li>

</transition-group>

</ul>

Make an animation for enter and leave function will be like below.

.user-list-enter-from{

    opacity: 0;

    transform: translateX(-30px);

}

.user-list-enter-active {

    transition: all 1s ease-out;

}

.user-list-leave-active {

    transition: all 1s ease-in;

}

.user-list-enter-to,

.user-list-leave-from {

    opacity: 1;

    transform: translateX(0);

}

.user-list-leave-to {

    opacity: 0;

    transform: translateX(30px);

}

Chapter 202 – Animate List Item Movement

1. We can control another element which are not getting added or removed, but which also might need to move around to make place for the new element or to fill up the space of the leaving element. In this example, the name list below the new user name (which is added) will move under the new one smoothly.

.user-list-move {

    transition: transform 0.8s ease;

}

Below is also to make the transition smoothly during leaving.\

.user-list-leave-active {

    transition: all 1s ease-in;

    position: absolute;

}

Chapter 203 – Animating Route Changes

1. To make animation for route changing, do some configuration using code below.

npm install –save vue-router@next

Then type below to restart the server.

npm run serve

1. If we want to aware of the page that should be loaded and it will therefore not make this initial transition. We can then mount your app inside of this function to only mount and therefore render your app once the router is fully set up and aware of the page that should be loaded.

router.isReady().then(function() {

    app.mount('#app');

});

1. Make an animation for router.

import { createRouter, createWebHistory } from 'vue-router';

const router = createRouter({

    history: createWebHistory(),

    routes: [

        { path: '/', component: AllUsers },

        { path: '/goals', component: CourseGoals },

    ]

});

const app = createApp(App);

app.component('base-modal', BaseModal);

app.use(router);

Chapter 204 – An Important Note On Animated Route Changes

When animating route changes as shown in the previous lecture, there's **one important thing** you have to keep in mind:

Your route components **must NOT have multiple root elements!**

For example, if your route component looked like this:

<template>

<section>

<h2>Section 1</h2>

</section>

<section>

<h2>Section 2</h2>

</section>

</template>

The <transition> component would **not be able to animate route changes** and you would get a **warning** in the JS console in your browser dev tools.

**Why?**

Because you must not forget that <transition> **needs exactly one child element** (*with the special exceptions you learned about in this module*).

If your route component has **multiple root elements**, <transition> in the end **has multiple children**and that is the **problem**.

Hence, if you want to transition between routes, you need to ensure that your route components **have exactly one root element** - for example:

<template>

<div>

<section>

<h2>Section 1</h2>

</section>

<section>

<h2>Section 2</h2>

</section>

</div>

</template>