Vue 3 Transitions classes

<transition> is an element that wraps around the element you want to apply a transition to.

<transition>

<h1>Hello</h1>

</transition>

You can apply a transition on entering the page. It has three stages

1. enter-from (the start of entering into the browser)
2. enter-to (the end point of entering into the browser)
3. enter-active (the process of entering into the browser which would actually be between “enter-from” and “enter-to”

The enter-form is the start state of the element. The enter-to is the end state of the element. The transition classes are set on the enter-active state. This is when the transition is taking place. The duration is set in this state

.enter-from{

Opacity: 0; // start = not visible

}

.enter-to{

Opacity: 1; // end = visible

}

.enter-from{

Transition: opacity 2s ease; // it will take 2 seconds to go from not visible to visible

}

Similar three classes can be set for leaving the page:

1. leave-from (the start of leaving the page)
2. leave-to (the end point of leaving the page)
3. leave-active (the process of leaving the page which would actually be between “leave-from” and “leave-to”

You can also name your transition like this:

<transition name=”title”>

<h1>Hello</h1>

</transition>

You would get the following classes (Vue regulates it like this)

title-enter-from

title-enter-to

title-enter-active

title-leave-from

title-leave-to

title-leave-active

This allows you to have a different transition an any element in the same page.

Vue creates the classes and they are ready to be configured

Transition group

This transition can be applied to a number of elements that will have the same transition. A good example would be an unsorted list:

<ul>

<li v-for=””></li>

</ul>

The list items would be the multiple items to apply the transition to. This can be achieved with a Transition Group.

The ul-tag will be replaced with a new TransitionGroup tag; but VUE will still have to know what the original tag was so it will know what to render. We have to add a tag attribute. It also will need a name.

<TransitionGroup tag=”ul” name=”list”>

<li v-for=””></li>

</TransitionGroup>

Conditional rendering.

If you have to divs that are rendered by a condition then we do not need to use a group transition. This is because only one element is visible at any time. So one element (depending on the condition) is leaving or entering. We can use single named transitions for this.

Example conditional rendering:

<div v-if="todos.length">

<TransitionGroup tag="ul" name="list" appear>

<li v-for="todo in todos" :key="todo.id" @click="deleteTodo(todo.id)">

{{ todo.text }}

</li>

</TransitionGroup>

</div>

<div v-else>Woohoo, nothing left todo!</div>

We wrap this entire piece of code with a named single transition:

<Transition name="todolist" mode="out-in">

<div v-if="todos.length">

<TransitionGroup tag="ul" name="list" appear>

<li v-for="todo in todos" :key="todo.id" @click="deleteTodo(todo.id)">

{{ todo.text }}

</li>

</TransitionGroup>

</div>

<div v-else>Woohoo, nothing left todo!</div>

</Transition>

We get the familiar classes for entering stage of elements and for the leave stage.

In this context transition mode can be “out-in” or “in-out”. We need to set this because otherwise the transition will be jerky. For instance “out-in” basically means that the element leaving will be transitioned first and the element entering will be transitioned afterwards consecutively (and thus not a the same time causing jerky transitions).