TransitionGroup

`<TransitionGroup>` is a built-in component designed for animating the insertion, removal, and order change of elements or components that are rendered in a list.

Differences from `<Transition>`

`<TransitionGroup>` supports the same props, CSS transition classes, and JavaScript hook listeners as `<Transition>`, with the following differences:

- By default, it doesn't render a wrapper element. But you can specify an element to be rendered with the `tag` prop.
- Transition modes are not available, because we are no longer alternating between mutually exclusive elements.
- Elements inside are **always required** to have a unique `key` attribute.
- CSS transition classes will be applied to individual elements in the list, **not** to the group / container itself.

TIP

When used in in-DOM templates, it should be referenced as `<transition-group>`.

Enter / Leave Transitions

Here is an example of applying enter / leave transitions to a `v-for` list using `<TransitionGroup>`:

```
    {{ item }}
  </li>
</TransitionGroup>
```

```css
.list-enter-active,
.list-leave-active {
  transition: all 0.5s ease;
}
.list-enter-from,
.list-leave-to {
  opacity: 0;
  transform: translateX(30px);
}
```

Add at random index Remove at random index

- 1
- 2
- 3
- 4
- 5

Move Transitions

The above demo has some obvious flaws: when an item is inserted or removed, its surrounding items instantly "jump" into place instead of moving smoothly. We can fix this by adding a few additional CSS rules:

```css
.list-move, /* apply transition to moving elements */
.list-enter-active,
.list-leave-active {
  transition: all 0.5s ease;
}

.list-enter-from,
.list-leave-to {
  opacity: 0;
  transform: translateX(30px);
}

/* ensure leaving items are taken out of layout flow so that moving
   animations can be calculated correctly. */
.list-leave-active {
```

Now it looks much better - even animating smoothly when the whole list is shuffled:

Add Remove Shuffle

- 1
- 2
- 3
- 4
- 5

Full Example

Custom TransitionGroup classes

You can also specify custom transition classes for the moving element by passing the `moveClass` prop to `<TransitionGroup>`, just like custom transition classes on `<Transition>`.

Staggering List Transitions

By communicating with JavaScript transitions through data attributes, it's also possible to stagger transitions in a list. First, we render the index of an item as a data attribute on the DOM element:

```template
<TransitionGroup
  tag="ul"
  :css="false"
  @before-enter="onBeforeEnter"
  @enter="onEnter"
  @leave="onLeave"
>
  <li
    v-for="(item, index) in computedList"
    :key="item.msg"
    :data-index="index"
  >
    {{ item.msg }}
  </li>
</TransitionGroup>
```

```js
function onEnter(el, done) {
  gsap.to(el, {
    opacity: 1,
    height: '1.6em',
    delay: el.dataset.index * 0.15,
    onComplete: done
  })
}
```

- Bruce Lee
- Jackie Chan
- Chuck Norris
- Jet Li
- Kung Fury

Full Example in the Playground

Related

- `<TransitionGroup>` API reference

Edit this page on GitHub