**[next →](https://www.javatpoint.com/react-bootstrap)**[**← prev**](https://www.javatpoint.com/react-css)

React Animation

The animation is a technique in which images are manipulated to appear as moving images. It is one of the most used technique to make an interactive web application. In React, we can add animation using an explicit group of components known as the **React Transition Group**.

React Transition Group is an add-on component for managing component states and useful for defining **entering** and **exiting** transitions. It is not able to animate styles by itself. Instead, it exposes transition states, manages classes and group elements, and manipulates the DOM in useful ways. It makes the implementation of visual transitions much easier.

React Transition group has mainly **two APIs** to create transitions. These are:

1. **ReactTransitionGroup:** It uses as a low-level API for animation.
2. **ReactCSSTransitionGroup:** It uses as a high-level API for implementing basic CSS transitions and animations.

## React Transition Group Components

React Transition Group API provides **three** main components. These are:

1. Transition
2. CSSTransition
3. Transition Group

### Transition

It has a simple component API to describe a transition from one component state to another over time. It is mainly used to animate the **mounting** and **unmounting** of a component. It can also be used for in-place transition states as well.

We can access the Transition component into four states:

* entering
* entered
* exiting
* exited

### CSSTransition

The CSSTransition component uses CSS stylesheet classes to write the transition and create animations. It is inspired by the **ng-animate** library. It can also inherit all the props of the transition component. We can divide the "CSSTransition" into **three** states. These are:

* Appear
* Enter
* Exit

CSSTransition component must be applied in a pair of class names to the child components. The first class is in the form of **name-stage** and the second class is in the **name-stage-active**. For example, you provide the name fade, and when it applies to the 'enter' stage, the two classes will be **fade-enter** and **fade-enter-active**. It may also take a prop as Timeout which defines the maximum time to animate.

### TransitionGroup

This component is used to manage a set of transition components (Transition and CSSTransition) in a list. It is a state machine that controls the **mounting** and **unmounting** of components over time. The Transition component does not define any animation directly. Here, how 'list' item animates is based on the individual transition component. It means, the "TransitionGroup" component can have different animation within a component.

Installation

We need to install **react-transition-group** for creating animation in React Web application. You can use the below command.

 npm install react-transition-group  --save

npm install react-addons-css-transition-group

create in src folder style.css

.example-appear {

opacity: 0.04;

}

.example-appear.example-appear-active {

opacity: 2;

transition: opacity 50s ease-in;

}

class App extends React.Component {

render() {

return (

<div>

<ReactCSSTransitionGroup transitionName = "example"

transitionAppear = {true} transitionAppearTimeout = {500}

transitionEnter = {false} transitionLeave = {false}>

<h1>My Element...</h1>

</ReactCSSTransitionGroup>

</div>

);

}

}

export default App;

Ex2:

**App.js**

**import** React, { Component } from 'react';

**import** { CSSTransitionGroup } from 'react-transition-group';

**class** App **extends** React.Component {

    constructor(props) {

**super**(props);

**this**.state = {items: ['Blockchain', 'ReactJS', 'TypeScript', 'JavaTpoint']};

**this**.handleAdd = **this**.handleAdd.bind(**this**);

  }

  handleAdd() {

**const** newItems = **this**.state.items.concat([

      prompt('Enter Item Name')

    ]);

**this**.setState({items: newItems});

  }

  handleRemove(i) {

    let newItems = **this**.state.items.slice();

    newItems.splice(i, 1);

**this**.setState({items: newItems});

  }

  render() {

**const** items = **this**.state.items.map((item, i) => (

      <div key={item} onClick={() => **this**.handleRemove(i)}>

        {item}

      </div>

    ));

**return** (

      <div>

    <h1>Animation Example</h1>

            <button onClick={**this**.handleAdd}>Insert Item</button>

            <CSSTransitionGroup

               transitionName="example"

           transitionEnterTimeout={800}

               transitionLeaveTimeout={600}>

               {items}

            </CSSTransitionGroup>

      </div>

    );

  }

}

export **default** App;

**style.css**

.example-enter {

  opacity: 0.01;

}

.example-enter.example-enter-active {

  opacity: 1;

  transition: opacity 500ms ease-in;

}

.example-leave {

  opacity: 1;

}

.example-leave.example-leave-active {

  opacity: 0.01;

  transition: opacity 300ms ease-in;

}