|  |  |
| --- | --- |
| **ch27.ReactJS Animation** | **Date: 22-02-2022** |

**Topics**

React Props Validation,

# React Props Validation

# 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.

## **Installation**

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

1. $ npm install react-transition-group --save

## **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.

Let us see the example below, which clearly help to understand the React Animation.

**Example**

**App.js**

In the App.js file, import react-transition-group component, and create the CSSTransition component that uses as a wrapper of the component you want to animate. We are going to use **transitionEnterTimeout** and **transitionLeaveTimeout** for CSS Transition. The Enter and Leave animations used when we want to insert or delete elements from the list.

1. **import** React, { Component } from 'react';
2. **import** { CSSTransitionGroup } from 'react-transition-group';
4. **class** App **extends** React.Component {
5. constructor(props) {
6. **super**(props);
7. **this**.state = {items: ['Blockchain', 'ReactJS', 'TypeScript', 'JavaTpoint']};
8. **this**.handleAdd = **this**.handleAdd.bind(**this**);
9. }
11. handleAdd() {
12. **const** newItems = **this**.state.items.concat([
13. prompt('Enter Item Name')
14. ]);
15. **this**.setState({items: newItems});
16. }
18. handleRemove(i) {
19. let newItems = **this**.state.items.slice();
20. newItems.splice(i, 1);
21. **this**.setState({items: newItems});
22. }
24. render() {
25. **const** items = **this**.state.items.map((item, i) => (
26. <div key={item} onClick={() => **this**.handleRemove(i)}>
27. {item}
28. </div>
29. ));
31. **return** (
32. <div>
33. <h1>Animation Example</h1>
34. <button onClick={**this**.handleAdd}>Insert Item</button>
35. <CSSTransitionGroup
36. transitionName="example"
37. transitionEnterTimeout={800}
38. transitionLeaveTimeout={600}>
39. {items}
40. </CSSTransitionGroup>
41. </div>
42. );
43. }
44. }
45. export **default** App;

**Main.js**

1. **import** React from 'react';
2. **import** ReactDOM from 'react-dom';
3. **import** App from './App.js';
5. ReactDOM.render(<App />, document.getElementById('app'));

**style.css**

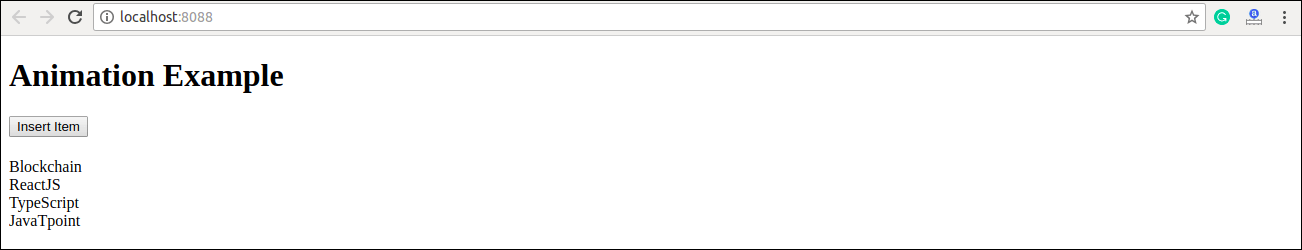
Add style.css file in your application, and add the following CSS styles. Now, to use this CSS file, you need to add the **link** of this file in your HTML file.

1. .example-enter {
2. opacity: 0.01;
3. }
5. .example-enter.example-enter-active {
6. opacity: 1;
7. transition: opacity 500ms ease-in;
8. }
10. .example-leave {
11. opacity: 1;
12. }
14. .example-leave.example-leave-active {
15. opacity: 0.01;
16. transition: opacity 300ms ease-in;
17. }

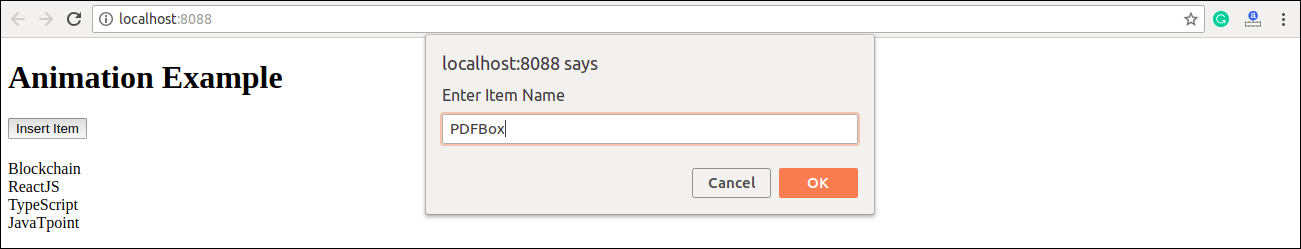
In the above example, the animation durations are specified in both the **CSS** and **render** method. It tells React component when to remove the animation classes from the list and if it is leaving when to remove the element from the DOM.

**Output**

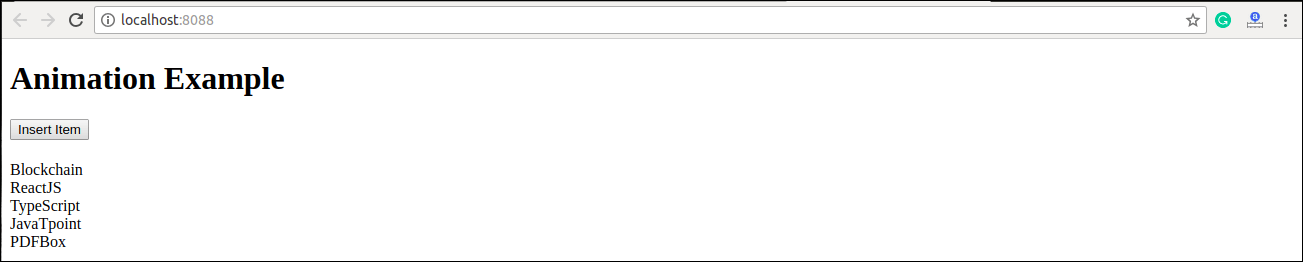
When we execute the above program, it gives the below output.



Click on '**Insert Item**' button, the following screen appears.



Once we insert the item and press **Ok**, the new item can be added in the list with fade in style. Here, we can also delete any item from the list by clicking on the particular link.



Next Topic[React Bootstrap](https://www.javatpoint.com/react-bootstrap)