React Keys

A key is a unique identifier. In React, it is used to identify which items have changed, updated, or deleted from the Lists. It is useful when we dynamically created components or when the users alter the lists. It also helps to determine which components in a collection needs to be re-rendered instead of re-rendering the entire set of components every time.

Keys should be given inside the array to give the elements a stable identity. The best way to pick a key as a string that uniquely identifies the items in the list.

Ex1:

import React,{ Component} from 'react';

class App extends Component {

render() {

return (

<div>

<h1> Keys example</h1>

<MenuBlog data={data}/>

</div>

);

}

}

function MenuBlog(props) {

const titlebar = (

<ol>

{props.data.map((show) =>

<li key={show.id}> {show.title}

</li>

)}

</ol>

);

const content = props.data.map((show) =>

<div key={show.id}> <h3>{show.title}: {show.content}</h3>

</div>

);

return (

<div>

{titlebar}

<hr />

{content}

</div>

);

}

const data = [

{id: 1, title: 'ES5', content: 'EcmaScript2015!!'},

{id: 2, title: 'ES6', content: 'Ecmascript2016!!'},

{id: 3, title: 'Typescript', content: 'Typescript is on top of ES5 and ES6!!'}

];

export default App;

Ex2:

import React, { Component } from 'react';

import logo from './logo.svg';

import './App.css';

import Todo from './Todo';

class App extends Component {

constructor(props) {

super(props);

this.state = {

show: false

};

}

render() {

return (

<div className="App">

<Todo />

</div>

);

}

}

export default App;

import React, { Component } from 'react';

class Todo extends Component {

constructor(props) {

super(props);

this.state = {

edit: false,

id: null,

mockData: [{

id: '1',

title: 'Buy Milk.',

done: false,

date: new Date()

}, {

id: '2',

title: 'Meeting with Ali.',

done: false,

date: new Date()

}, {

id: '3',

title: 'Tea break.',

done: false,

date: new Date()

}, {

id: '4',

title: 'Go for a run.',

done: false,

date: new Date()

}]

}

}

onSubmitHandle(event) {

event.preventDefault();

this.setState({

mockData: [...this.state.mockData, {

id: Date.now(),

title: event.target.item.value,

done: false,

date: new Date()

}]

});

event.target.item.value = '';

}

onDeleteHandle() {

let id = arguments[0];

this.setState({

mockData: this.state.mockData.filter(item => {

if (item.id !== id) {

return item;

}

})

});

}

onEditHandle(event) {

this.setState({

edit: true,

id: arguments[0],

title: arguments[1]

});

}

onUpdateHandle(event) {

event.preventDefault();

this.setState({

mockData: this.state.mockData.map(item => {

if (item.id === this.state.id) {

item['title'] = event.target.updatedItem.value;

return item;

}

return item;

})

});

this.setState({

edit: false

});

}

renderEditForm() {

if (this.state.edit) {

return <form onSubmit={this.onUpdateHandle.bind(this)}>

<input type="text" name="updatedItem" className="item" defaultValue={this.state.title} />

<button className="update-add-item">Update</button>

</form>

}

}

render() {

return (

<div>

{this.renderEditForm()}

<form onSubmit={this.onSubmitHandle.bind(this)}>

<input type="text" name="item" className="item" />

<button className="btn-add-item">Add</button>

</form>

<ul>

{this.state.mockData.map(item => (

<li key={item.id} className={ item.done ? 'done' : 'hidden' }>

{item.title}

<button onClick={this.onDeleteHandle.bind(this, item.id)}>Delete</button>

<button onClick={this.onEditHandle.bind(this, item.id, item.title)}>Edit</button>

</li>

))}

</ul>

</div>

);

}

}

export default Todo;