Recap

You have learned the basics on how to write your own React application! This lesson will provide you a recap to what you have learned so far and will also share how your App.js code looks by now.

Let's recap the last chapter:

- **React**
  - Use `this.state` and `setState()` to manage your local component state
  - Pass functions or class methods to your element handler
  - Use forms and events in React to add interactions
  - Unidirectional data flow is an important concept in React
  - Embrace controlled components
  - Compose components with children and reusable components
  - Usage and implementation of ES6 class components and functional stateless components
  - Approaches to style your components
- **ES6**
  - Functions that are bound to a class are class methods
  - Destructuring of objects and arrays
  - Default parameters
- **General**
  - Higher-order functions

Again, it makes sense to take a break, internalize the lessons, and apply them on your own. Experiment with the source code you have written so far. The source code for this project is found in the official repository.

Your `src/App.js` should look like the following by now:

```
import React, { Component } from 'react';
require('./App.css');
require('./App.css');

const list = [
  {
    title: 'React',
    url: 'https://reactjs.org/',
    author: 'Jordan Walke',
    num_comments: 3,
    points: 4,
    objectID: 0,
  },
  {
    title: 'Redux',
    url: 'https://redux.js.org/',
    author: 'Dan Abramov, Andrew Clark',
    num_comments: 2,
    points: 5,
    objectID: 1,
  },
];

const isSearched = (searchTerm) => (item) =>
  item.title.toLowerCase().includes(searchTerm.toLowerCase());

class App extends Component {

  constructor(props) {
    super(props);

    this.state = {
      list,
      searchTerm: '',
    };

    this.onSearchChange = this.onSearchChange.bind(this);
    this.onDismiss = this.onDismiss.bind(this);
  }

  onSearchChange(event) {
    this.setState({ searchTerm: event.target.value });
  }

  onDismiss(id) {
    const isNotId = item => item.objectID !== id;
    const updatedList = this.state.list.filter(isNotId);
    this.setState({ list: updatedList });
  }

  render() {
    const { searchTerm, list } = this.state;
    return (
      <div className="page">
        <div className="interactions">
          <Search
            value={searchTerm}
            onChange={this.onSearchChange}
          >
            Search
          </Search>
        </div>
        <Table
          list={list}
          pattern={searchTerm}
          onDismiss={this.onDismiss}
        />
      </div>
    );
  }
}


const Search = ({ value, onChange, children }) =>
  <form>
    {children} <input
      type="text"
      value={value}
      onChange={onChange}
    />
  </form>

const Table = ({ list, pattern, onDismiss }) =>
  <div className="table">
    {list.filter(isSearched(pattern)).map(item =>
      <div key={item.objectID} className="table-row">
        <span style={{ width: '40%' }}>
          <a href={item.url}>{item.title}</a>
        </span>
        <span style={{ width: '30%' }}>
          {item.author}
        </span>
        <span style={{ width: '10%' }}>
          {item.num_comments}
        </span>
        <span style={{ width: '10%' }}>
          {item.points}
        </span>
        <span style={{ width: '10%' }}>
          <Button
            onClick={() => onDismiss(item.objectID)}
            className="button-inline"
          >
            Dismiss
          </Button>
        </span>
      </div>
    )}
  </div>

const Button = ({ onClick, className = '', children }) =>
  <button
    onClick={onClick}
    className={className}
    type="button"
  >
    {children}
  </button>

export default App;
```