TodoItem Component

In this lesson, we create a TodoItem component that displays one item in the list.

WE'LL COVER THE FOLLOWING

TodoItem

We create a TodoItem component, based on several hooks we created: `useQuery`, `useDeleteTodo`, `useToggleTodo`, and `useFlasher`

```
import React from 'react';

import { useDispatch, useTrackedState } from './store';
import { useFlasher } from './utils';

const renderHighlight = (title, query) => {
  if (!query) return title;
  const index = title.indexOf(query);
  if (index === -1) return title;
  return (
    <React.Fragment>
      {title.slice(0, index)}
      <b>{query}</b>
      {title.slice(index + query.length)}
    </React.Fragment>
  );
};

const TodoItem = ({ id, title, completed }) => {
  const dispatch = useDispatch();
  const state = useTrackedState();
  const delTodo = () => {
    dispatch({ type: 'DELETE_TODO', id });
  };
  };
  return (
    <li ref={useFlasher()}>
      <input
        type="checkbox"
        checked={!!completed}
        onChange={() => dispatch({ type: 'TOGGLE_TODO', id })}
      />
      <span
        style={{
          textDecoration: completed ? 'line-through' : 'none',
        }}
      >
        {completed ? title : renderHighlight(title, state.query)}
      </span>
      <button onClick={delTodo}>Delete</button>
    </li>
  );
};

export default React.memo(TodoItem);
```

This component receives 3 props: `id`, `title`, and `completed`. They are all primitive values. An alternative way is to receive 1 prop, `todo` object.

This is specific to React Tracked, but there's a pitfall with object props if you use React.memo. With React Tracked, we recommend primitive props for components that are wrapped by `React.memo`. If one needs object props there's a workaround, but it's out of the scope of this course.

Next

In the next lesson, we create `NewTodo` component.