1. Очищаем проект от лишнего
2. Импортируем hook в проект

import { useState } from 'react';

1. Для работы с тудушками нужна инфа о листе (что мы используем), изначально пустой массив:

todos – список дел,

setTodos – ф-ция для обновления списка

const [todos, setTodos] = useState([]);

1. Нужна форма для добавления новых элементов

text, setText - сначала пустая строка

const [text, setText] = useState('');

1. Разметку накидываем в return()

У input велью будет текст, а на событие onChange будет поэлементно вызываться SetText

Передавая туда из эвента target.value

<input value={text} onChange={(e)=>setText(e.target.value)}></input>

1. Добавляем кнопку и создаем метод addTodo, который будет выполнять эту операцию и вешаем на onClick кнопки

<button onClick={addTodo}>Add Todo</button>

А метод addTodo – вызывает метод setTodos и вносит новый элемент при этом оставляя все старые элементы

//------------------------------- добавление todo -----------------------------

  const addTodo = () => {

    if (text.trim().length) { //проверяем или не пустой текст

      setTodos([

        ...todos,

        {

          id: new Date().toISOString(), // идентификатор

          text,                         // сам текст

          complited: false, // статус завершенности (вначале дело еще не завершено)

        }

      ])

      setText('') // сбрасываем на пустой после добавления

    }

  }

//==============================================================================

Каждый новый эл-мент будет состоять из трех полей

1. Сам список будет выводиться в маркированном списке посредством обхода массива: для каждого дела будет создаваться свой listItem, где key будет брать с todos.id, в качестве содержимого будет: чекбокс, текст и кнопка для удаления

      <ul>

        {

          todos.map(todo => <li key={todo.id}>

            <input type="checkbox" />

            <span>{todo.text}</span>

            <span style={{ color: 'red' }}>&times;</span> {/\* спец символ \*/}

          </li>)

        }

      </ul>

1. Для удаления метод будет принимать айдишник и по сути фильтровать что есть, мы будет устанавливать в качестве значения нового списка дел отфильтрованный предыдущий оригинальный массив (мы будем брать все дела и смотреть на предмет того, что идентификатор не равен тому, что мы получим

  const removeTodo = (todoId) => {

    setTodos(todos.filter(todo => todo.id !== todoId))

  }

И повешаем событие онклик – каждый раз , когда я кликаю вызывать removeTodo и передавать туда айдишник дела

<span className='delete' onClick={() => removeTodo(todo.id)}>&times;</span>

todo.id берется отсюда - todos.map(todo => <li key={todo.id}>

1. И checkbox-у делаем управляемость checked и он будет работать в зависимости от todo.complited

<input type="checkbox" checked={todo.complited} />

Цепляем событие onChange c методо, который будет принимать айдишник дела.

<input type="checkbox" checked={todo.complited} onChange={() => { toggleTodoComplete(todo.id) }} />

В методе – берем оригинальный массив тудушек и замапим его. Внутри для каждого туду мы будем смотреть, если у этого туду его айди не равен нашему, который мы получили извне, то будем возвращать оригинальный туду. Если условие не выполнилось, мы нашли нужную тудушку – мы преобразовываем todo.complited на обратный и потом возвращаем видоизмененную тудушку

  const toggleTodoComplete = (todoId) => {

    setTodos(

      todos.map(

        todo => {

          if (todo.id !== todoId) return todo;

          return {

            ...todo,

            complited: !todo.complited,

          }

        }

      )

    );

1. Для визуализации – в файле scc – для инпута со значением чекет возьми следующий спан после него и сделай текст перечеркнутым

input:checked + span {

  text-decoration: line-through;

}

1. Ы
2. Ы
3. Ы
4. Ы
5. Ы
6. Ы
7. Ы
8. Ы