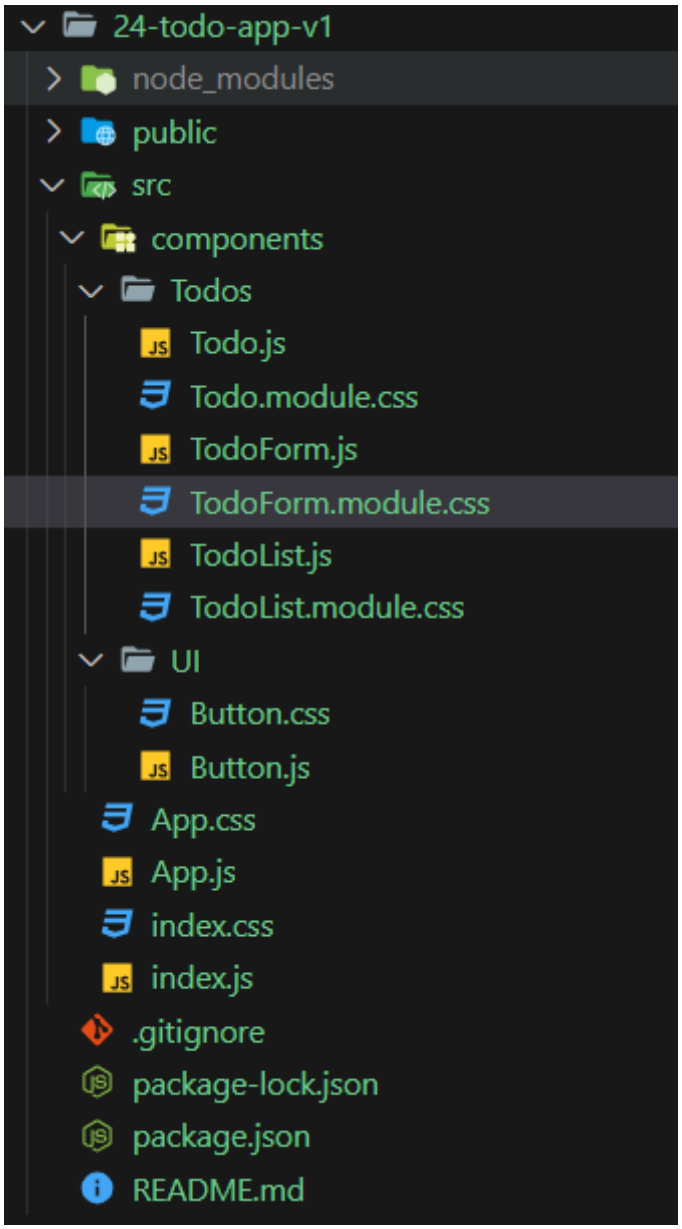
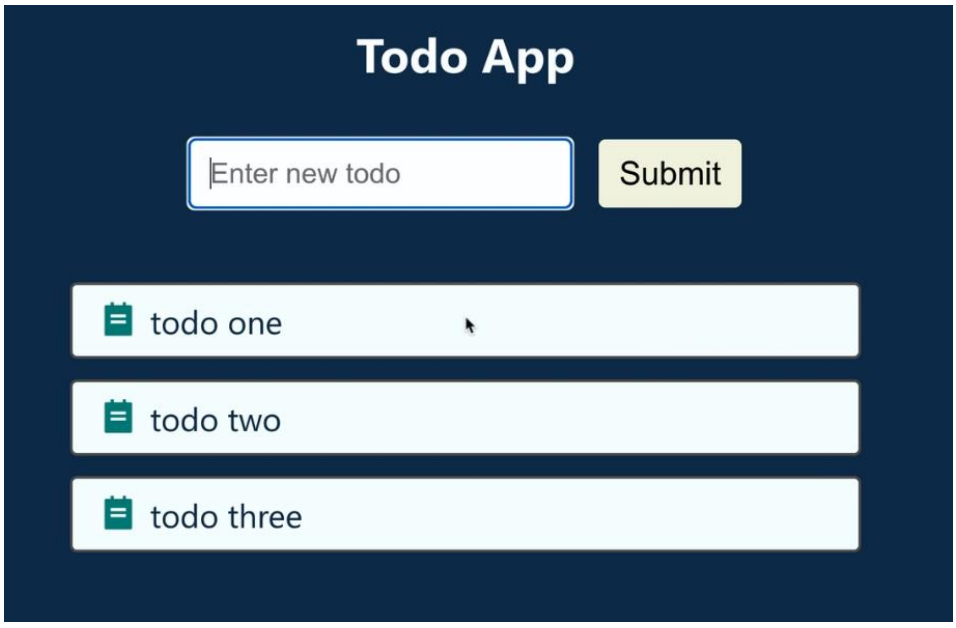
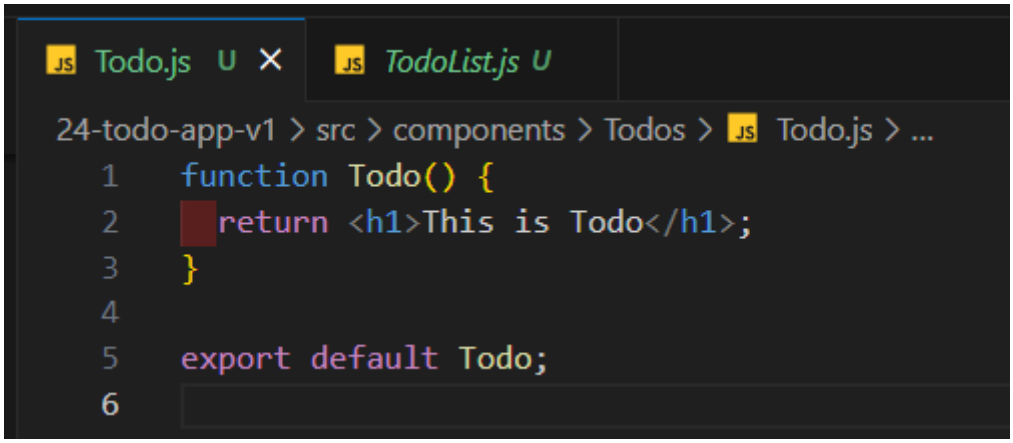
Первая версия проекта Todo App

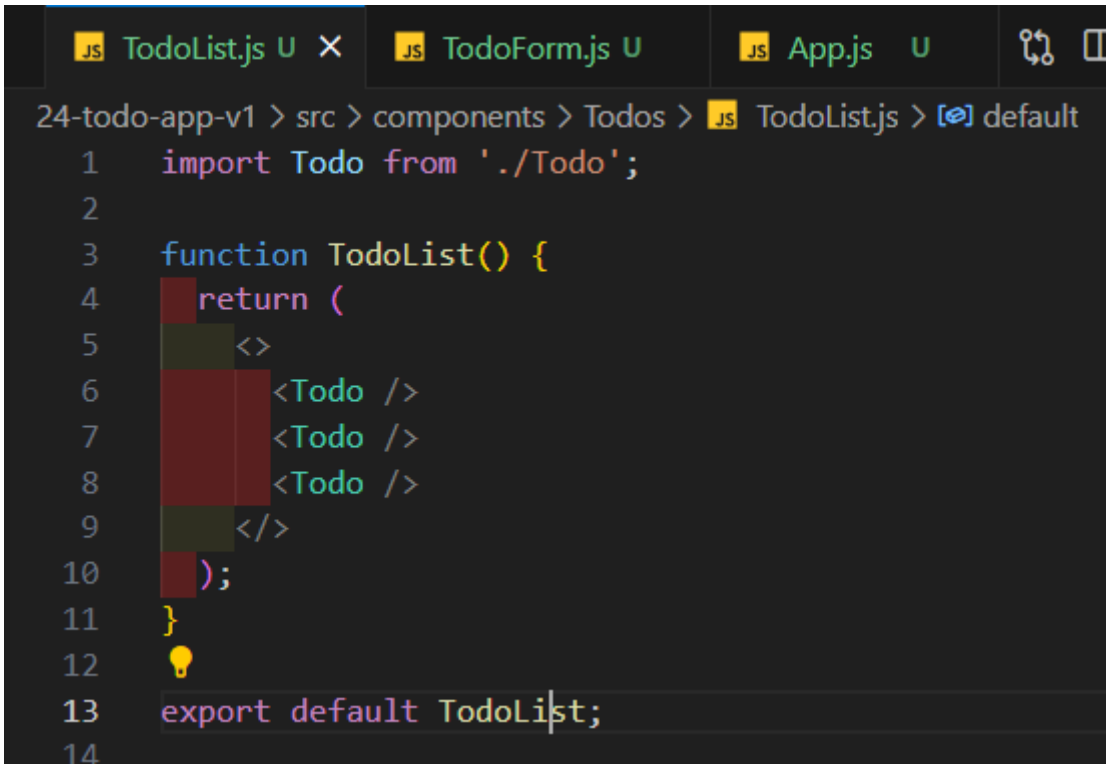




Начнем проект с создания заготовок компонент и постепенно будем добавлять функционал.

Создадим форму и список задач. Тексты для задач пока поместим в массив строк. Пока без стилевого оформления.





TodoList.js U | TodoForm.js U | App.js U

24-todo-app-v1 > src > components > Todos > TodoForm.js > default

```
1  function TodoForm() {
2    return (
3      <form>
4        <input placeholder="Enter new todo" />
5        <button type="submit">Submit</button>
6      </form>
7    );
8  }
9
10 export default TodoForm
```

TodoList.js U | TodoForm.js U | App.js U

24-todo-app-v1 > src > App.js > default

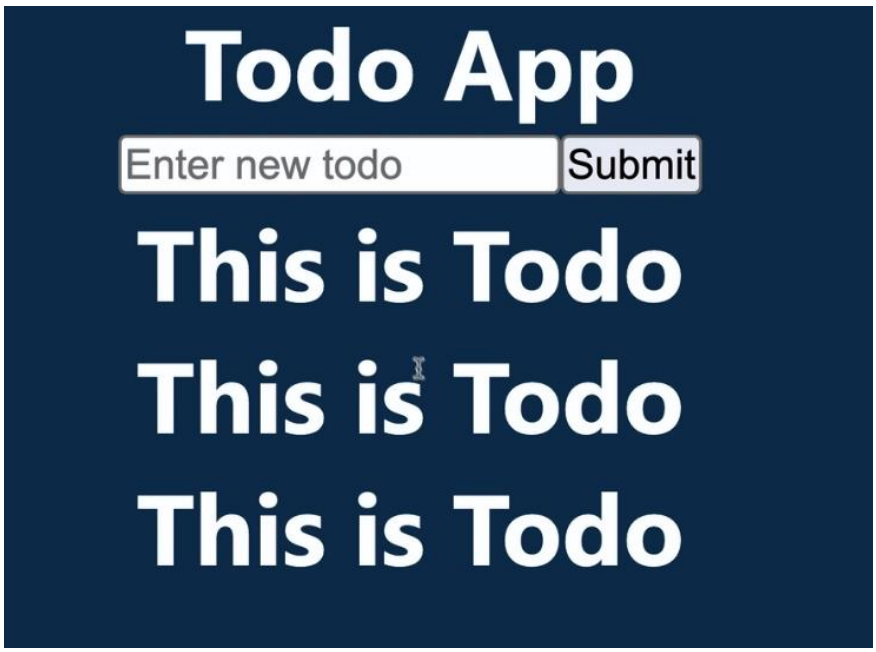
```
1  import TodoForm from './components/Todos/TodoForm';
2  import TodoList from './components/Todos/TodoList';
3  import './App.css';
4
5  function App() {
6    return (
7      <div className="App">
8        <h1>TodoApp</h1>
9        <TodoForm />
10       <TodoList />
11     </div>
12   );
13 }
14
15 export default App;
16
```

Создание заготовок закончили и переходим к стилизации

index.css U

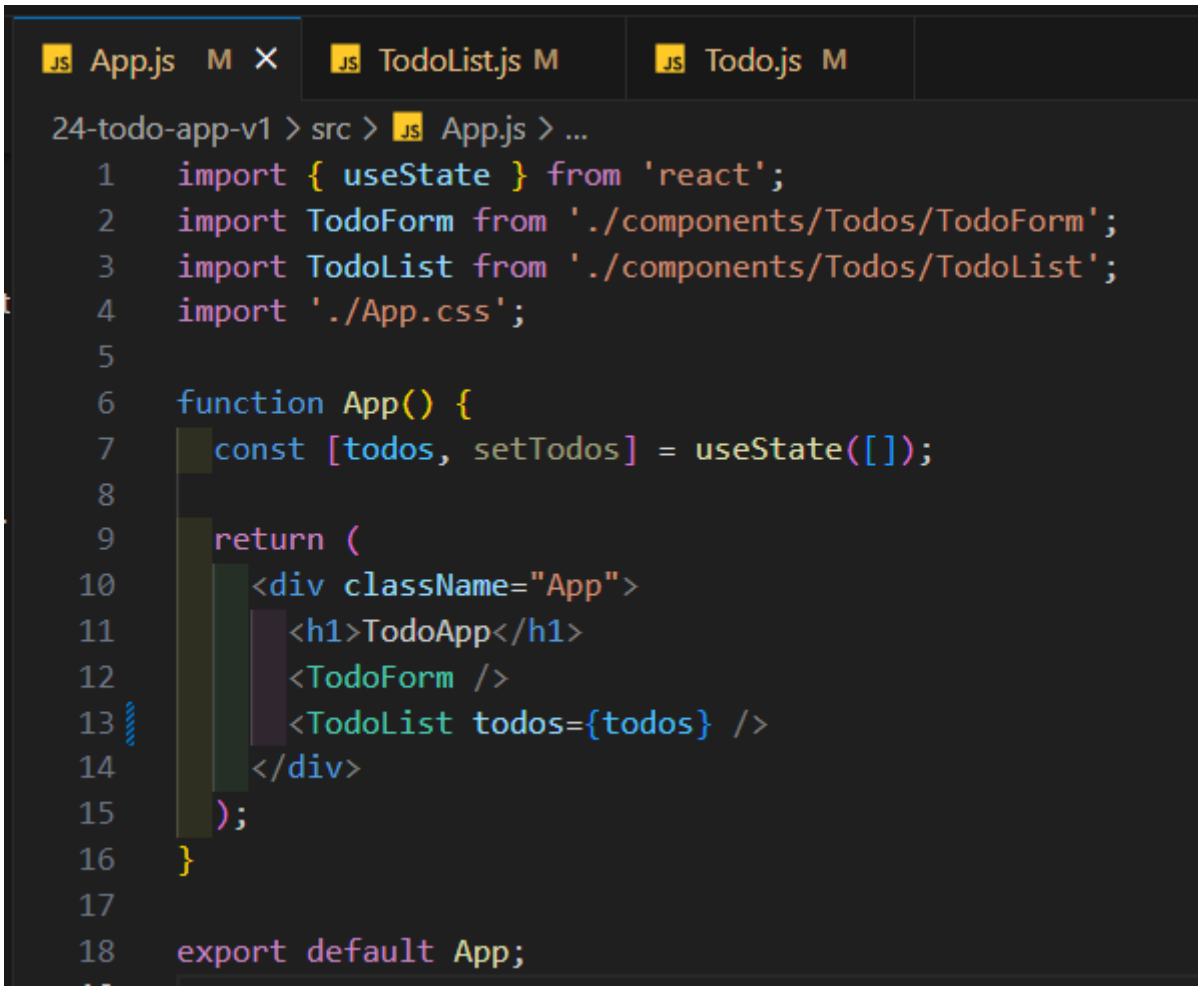
24-todo-app-v1 > src > index.css > ...

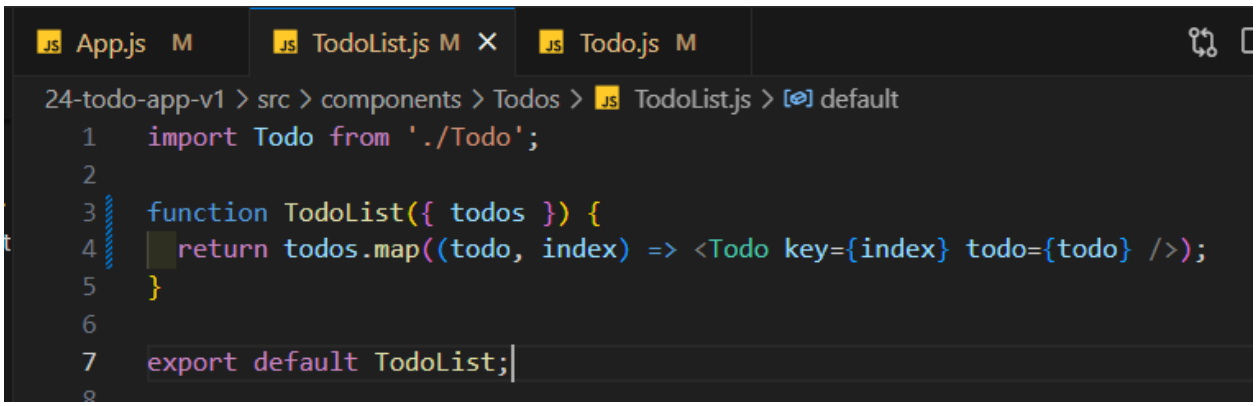
```css
* {
  margin: 0;
  padding: 0;
}

body {
  font-family: 'Segoe UI', 'Roboto', 'Oxygen',
  'Ubuntu', 'Cantarell',
    'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  color: white;
  background-color: #112d49;
}
```



Реализация базового функционала.

Сделаем так, чтобы форма работала, а ниже появлялись задачи. Для этого необходимо создать состояния, в котором мы будем хранить массив строк, т.е. каждая задача – это строка. Состояние разумно завести в App.js, т.к в нем есть дочерние компоненты TodoForm и TodoList. В TodoForm мы будем добавлять новые компоненты, а в TodoList будем их отображать. Нам необходимо иметь возможность изменять состояние в TodoForm, т.е. иметь возможность вызывать setTodos. Кроме этого мы должны передавать список задач в компонент TodoList.

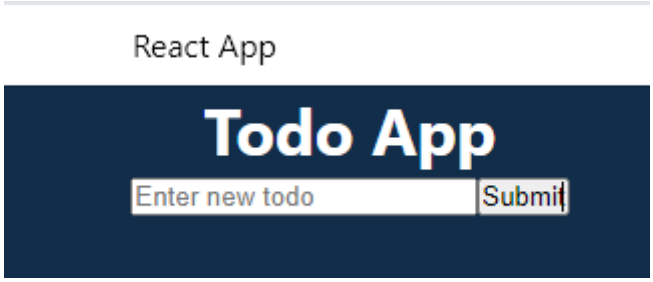




App.js M | TodoList.js M | Todo.js M | TodoForm.

24-todo-app-v1 > src > components > Todos > Todo.js > ...

```
1  function Todo({ todo }) {
2    return <h3>{todo}</h3>;
3  }
4
5  export default Todo;
6
```



Список задач пока пуст, но можно пока статически в массив добавить несколько задач:

App.js M | TodoList.js M | Todo.js M | TodoForm.js

24-todo-app-v1 > src > App.js > default

```
1  import { useState } from 'react';
2  import TodoForm from './components/Todos/TodoForm';
3  import TodoList from './components/Todos/TodoList';
4  import './App.css';
5
6  function App() {
7    const [todos, setTodos] = useState(['todo one', 'todo two']);
8
9    return (
10     <div className="App">
```

Реализуем функционал добавления задач с помощью формы

App.js M | TodoList.js M | Todo.js M | TodoForm.js M

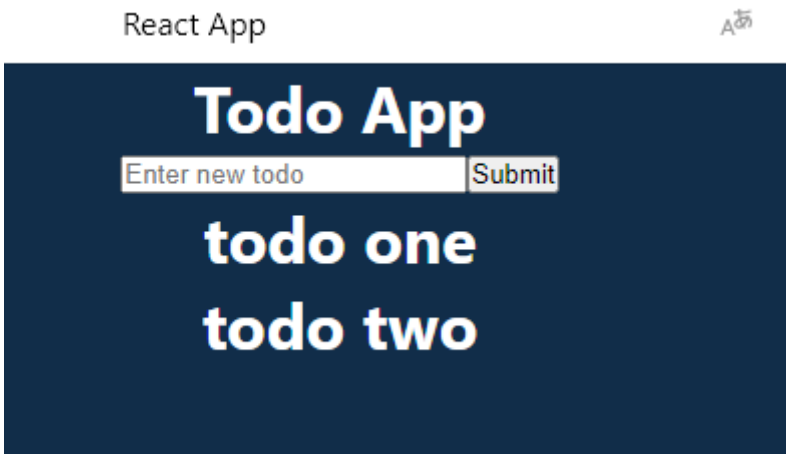
24-todo-app-v1 > src > App.js > ...

```
1   import { useState } from 'react';
2   import TodoForm from './components/Todos/TodoForm';
3   import TodoList from './components/Todos/TodoList';
4   import './App.css';
5
6   function App() {
7     const [todos, setTodos] = useState([]);
8     const addTodoHandler = (text) => {
9       setTodos([...todos, text]);
10    };
11
12    return (
13      <div className="App">
14        <h1>Todo App</h1>
15        <TodoForm addTodo={addTodoHandler} />
16        <TodoList todos={todos} />
17      </div>
18    );
19  }
20
21  export default App;
22
```

TodoForm.js M | App.js M | TodoList.js M | Todo.js M

24-todo-app-v1 > src > components > Todos > TodoForm.js > default

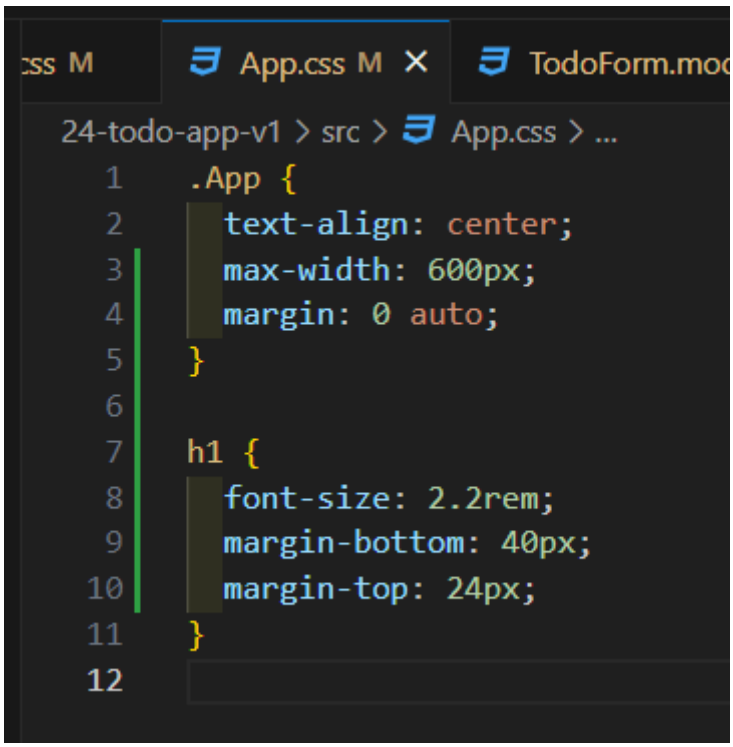
```jsx
import { useState } from 'react';

function TodoForm({ addTodo }) {
  const [text, setText] = useState('');

  function onSubmitHandler(event) {
    event.preventDefault();
    addTodo(text);
    setText('');
  }

  return (
    <form onSubmit={onSubmitHandler}>
      <input
        placeholder="Enter new todo"
        value={text}
        onChange={(e) => setText(e.target.value)}
      />
      <button type="submit">Submit</button>
    </form>
  );
}

export default TodoForm;
```



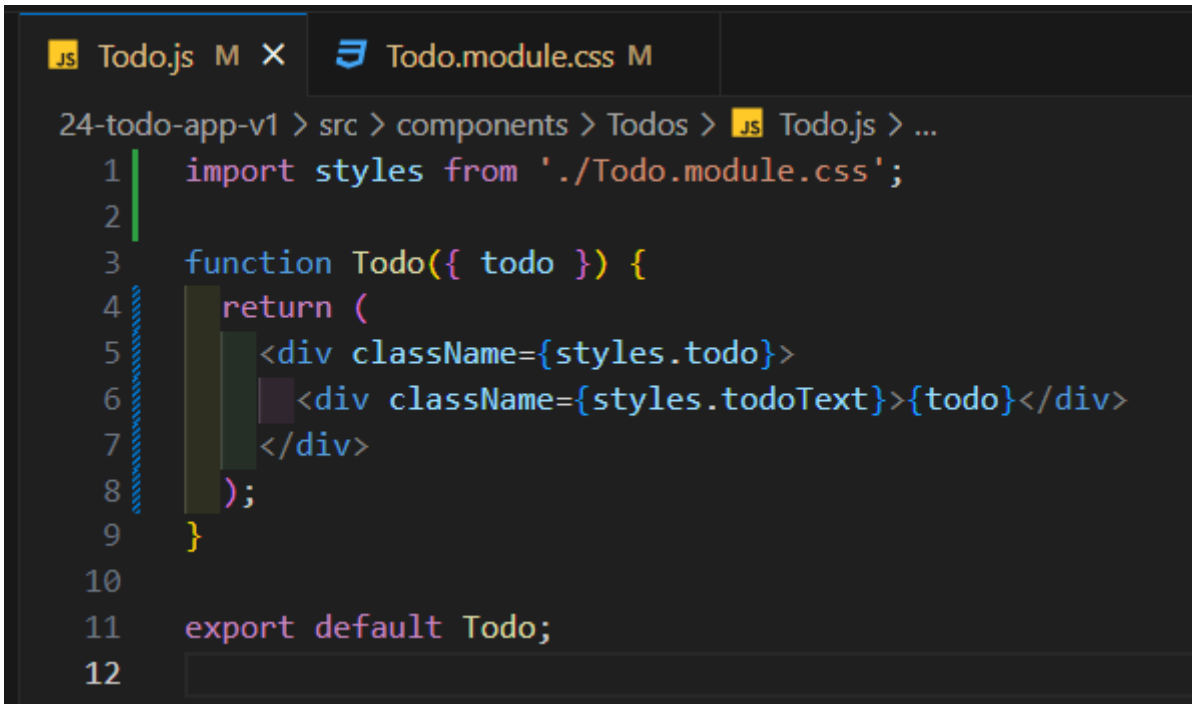
Добавление стилей CSS для компонента TodoForm TodoForm.module

Здесь наша задача локализовать свойства через module.css что бы они не распространялись на весь проект, а только на форму. Для этого обернем форму вместе с ее содержимым в дополнительный и назначим ему свойство по классу .todoFormContainer, затем каждому JSX тегу укажем локализацию - .todoFormContainer


css M
TodoForm.module.css M
TodoForm.js M
App.
24-todo-app-v1 > src > components > Todos > TodoForm.module.css >
.todoFormContainer {
margin-bottom: 30px;
}
.todoFormContainer form {
display: flex;
align-items: center;
justify-content: center;
}
.todoFormContainer input {
width: 45%;
height: 30px;
font-size: 1.3rem;
}
.todoFormContainer button {
margin-left: 20px;
height: 50px;
cursor: pointer;
background-color: beige;
font-size: 1.5rem;
}
.todoFormContainer button:hover {
background-color: rgb(240, 240, 155);
}
.todoFormContainer input,
.todoFormContainer button {
padding: 10px 15px;
border: none;
border-radius: 5px;
}



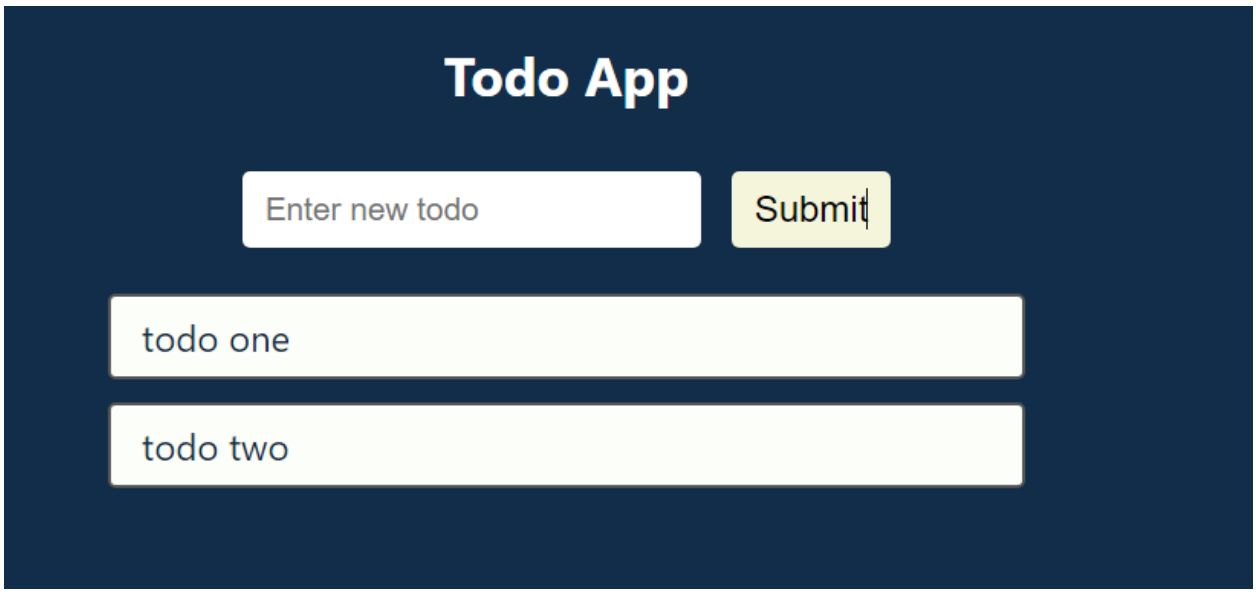
Добавление стилей CSS для компонента Todo Todo.module



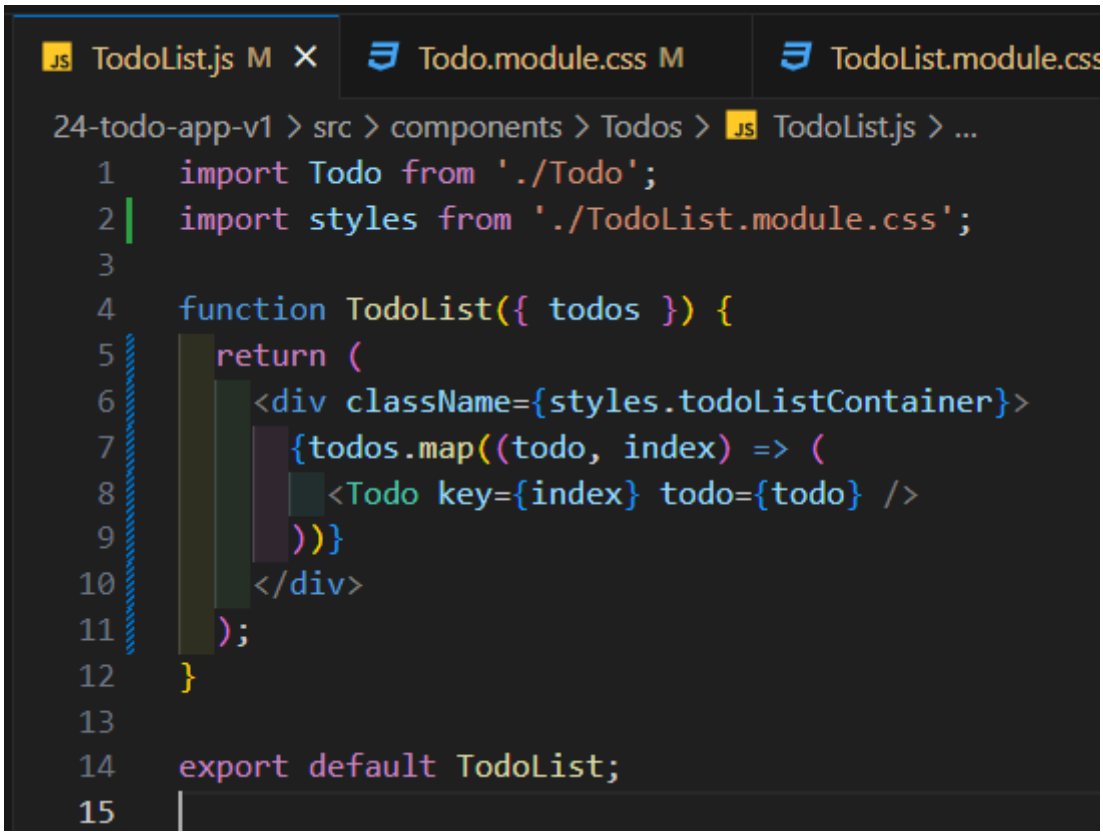
Todo.module.css M Todo.js M

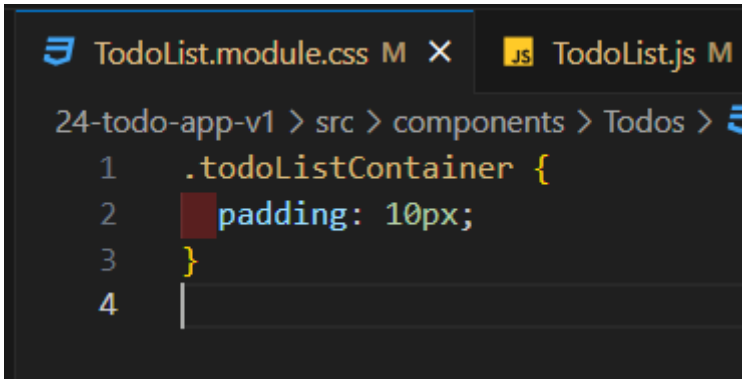
24-todo-app-v1 > src > components > Todos > Todo.mc

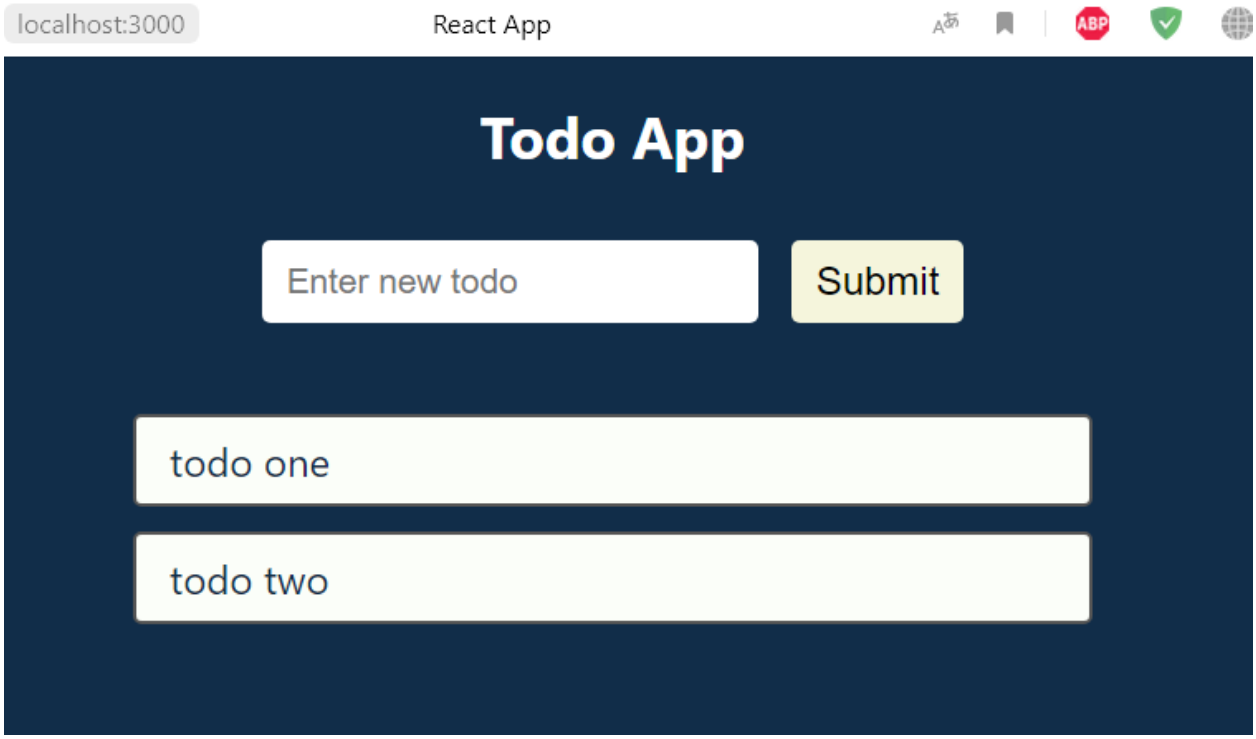
```css
.todo {
  display: flex;
  align-items: center;
  padding: 10px 20px;
  margin: 15px 0;
  font-size: 1.5rem;
  border-radius: 5px;
  border: 2px solid #555;
  color: #112d49;
  background-color: #fbfef9;
  user-select: none;
}

.todoText {
  width: 100%;
  text-align: left;
}

```



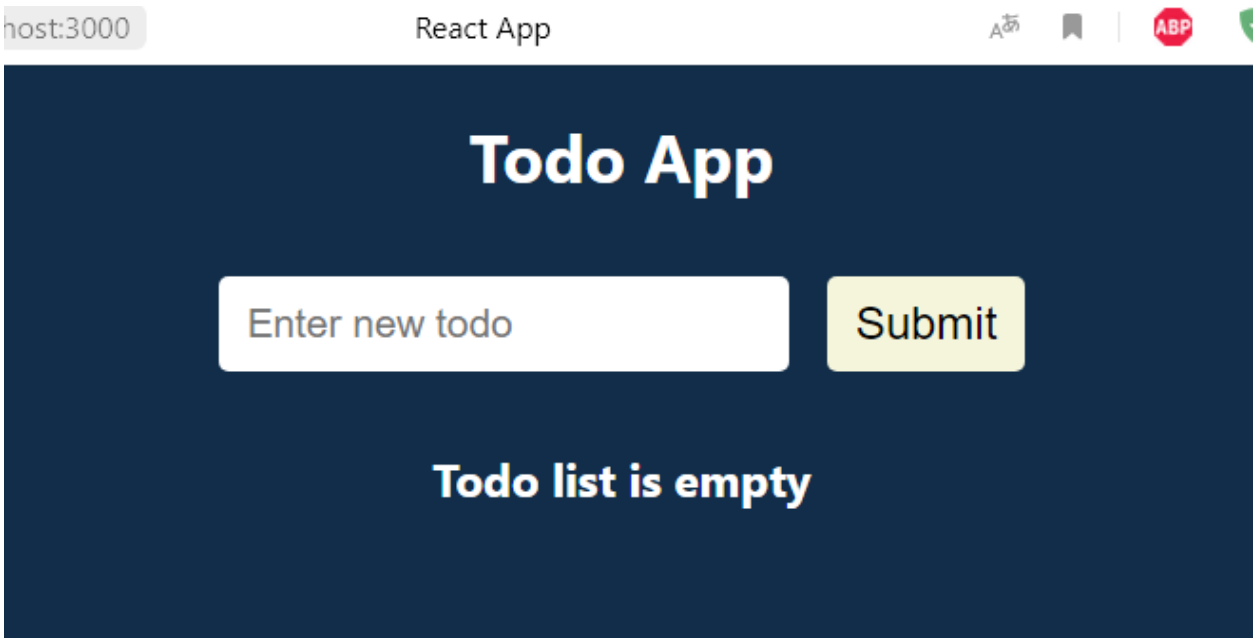
Добавим и подключим стили для todoList.js






localhost:3000
React App
Todo App
Enter new todo
Submit
todo one
todo two

Отображение текста об отсутствии задач



TodoList.js M

24-todo-app-v1 > src > components > Todos > TodoList.js > ...

```
1   import Todo from './Todo';
2   import styles from './TodoList.module.css';
3
4   function TodoList({ todos }) {
5     return (
6       <div className={styles.todoListContainer}>
7         {!todos.length && <h2>Todo list is empty</h2>}
8         {todos.map((todo, index) => (
9           <Todo key={index} todo={todo} />
10        ))}
11      </div>
12    );
13  }
14
15  export default TodoList;
16
```

Завершение задачи двойным кликом

Для этого добавим обработчик события «двойной щелчок мыши» на задаче. Это делается в компоненте Todo, добавим свойство onDubleClick, ему передадим функцию, которая будет отвечать за удаление определенной задачи из списка. Но сами задачи находятся в состоянии компонента App.js. Значит добавим обработчик удаления задачи, аналогично функции addTodoHandler, которая добавляет задачи. Удалять определенную задачу мы будем по ее индексу в списке задач.

Функция deleteTodoHandler с одним параметром index. В этой функции вызовем функцию setTodos, которая изменит состояние компонента App.js. В эту функцию нужно предать массив, в котором будет отсутствовать задача с index, который мы передает в функцию. Для этого используем метод filter, который возвращает новый массив, а в его вызове используем callback функцию, которая будет использована для каждого элемента в массиве Todos.

В callback функции будет два параметра: значение элемента в массиве и индекс элемента в массиве. Далее мы будем сравнивать этот индекс с индексом, который мы передаем в вызове deleteTodoHandler. Мы оставим все элементы за исключением того, индекс которого совпадает с index.

```
const deleteTodoHandler = (index) => {
  setTodos(todos.filter((todo, idx) => idx !== index));
};
```

Т.к. переменная todo не используется, то можно записать ее как _.

```
const deleteTodoHandler = (index) => {
  setTodos(todos.filter((_, idx) => idx !== index));
};
```

Теперь этот обработчик нужно передать в компонент TodoList, а он в свою очередь передаст его в компонент Todo

App.js M | TodoList.js M | Todo.js M

24-todo-app-v1 > src > App.js > App

```jsx
import { useState } from 'react';
import TodoForm from './components/Todos/TodoForm';
import TodoList from './components/Todos/TodoList';
import './App.css';

function App() {
  const [todos, setTodos] = useState([]);

  const addTodoHandler = (text) => {
    setTodos([...todos, text]);
  };

  const deleteTodoHandler = (index) => {
    setTodos(todos.filter((_, idx) => idx !== index));
  };

  return (
    <div className="App">
      <h1>Todo App</h1>
      <TodoForm addTodo={addTodoHandler} />
      <TodoList todos={todos} deleteTodo={deleteTodoHandler} />
    </div>
  );
}

export default App;
```

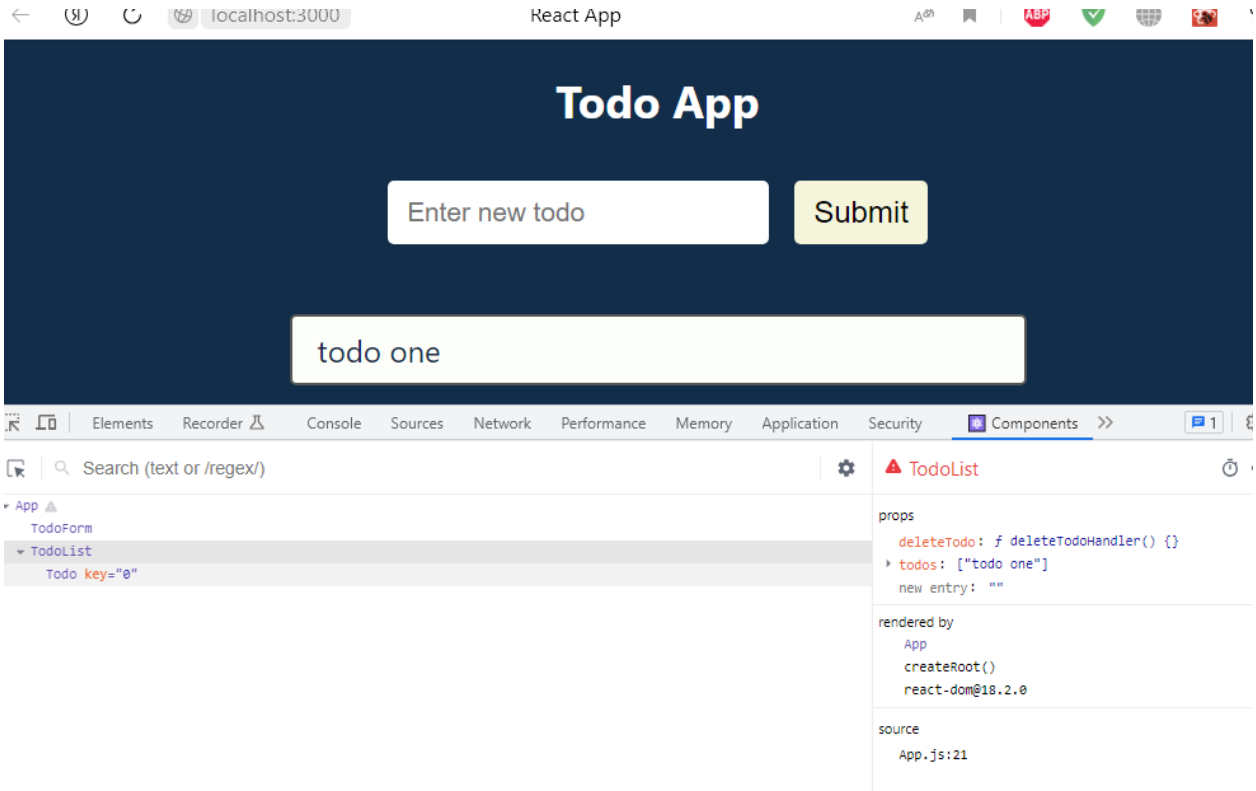
TodoList.js M | App.js M | Todo.js M

24-todo-app-v1 > src > components > Todos > TodoList.js > TodoList

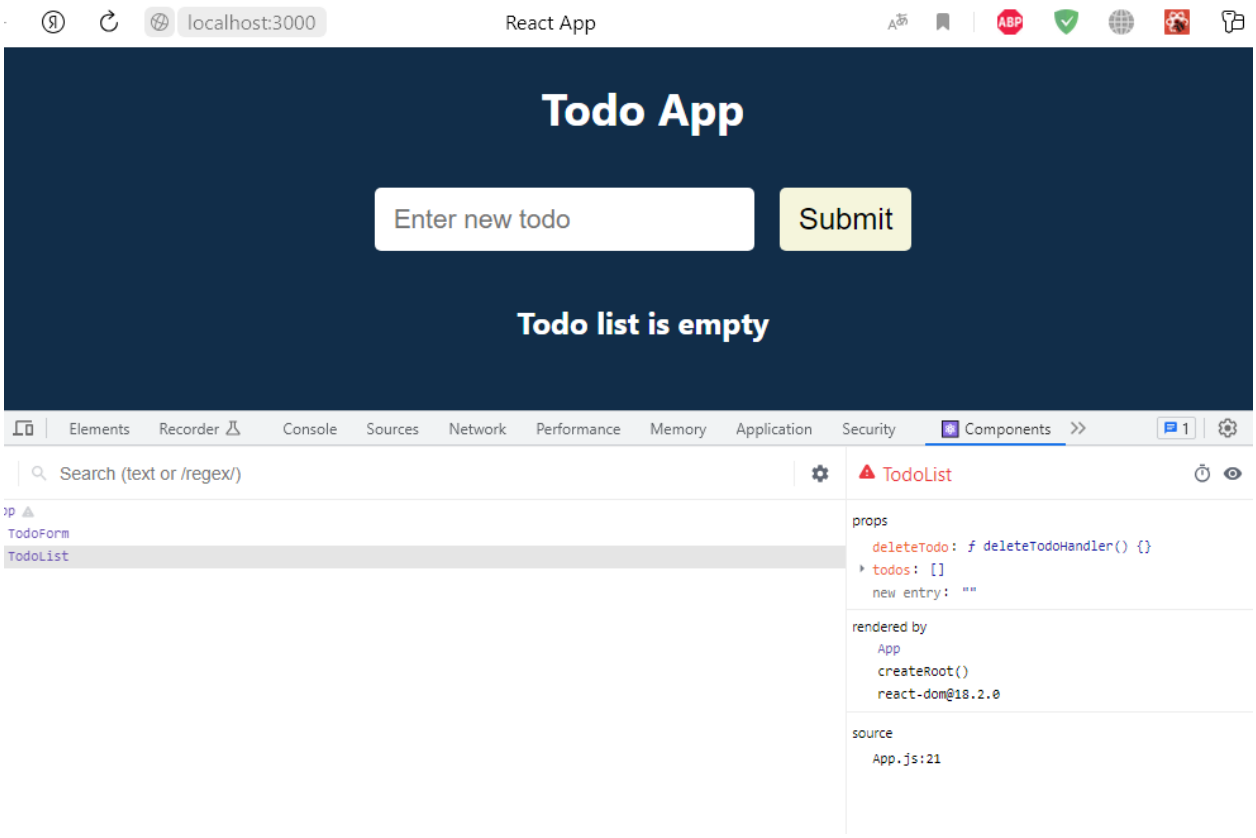
```jsx
import Todo from './Todo';
import styles from './TodoList.module.css';

function TodoList({ todos, deleteTodo }) {
  return (
    <div className={styles.todoListContainer}>
      {!todos.length && <h2>Todo list is empty</h2>}
      {todos.map((todo, index) => (
        <Todo key={index} todo={todo} index={index} deleteTodo={deleteTodo} />
      ))}
    </div>
  );
}

export default TodoList;
```

```
Todo.js M   TodoList.js M   App.js M
24-todo-app-v1 > src > components > Todos > Todo.js > Todo
1   import styles from './Todo.module.css';
2
3   function Todo({ todo, index, deleteTodo }) {
4     return (
5       <div className={styles.todo} onDoubleClick={() => deleteTodo(index)}>
6         <div className={styles.todoText}>{todo}</div>
7       </div>
8     );
9   }
10
11  export default Todo;
12
```

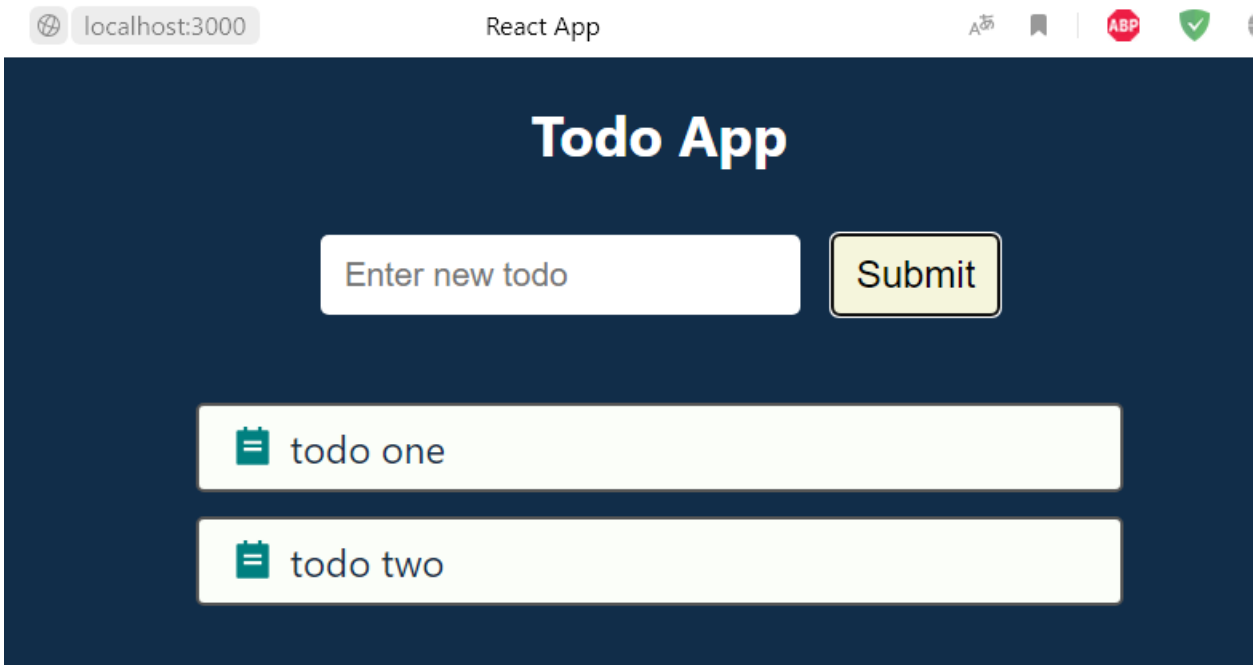
Проверим состояние компонента в момент добавления задачи



…. И после ее удаления



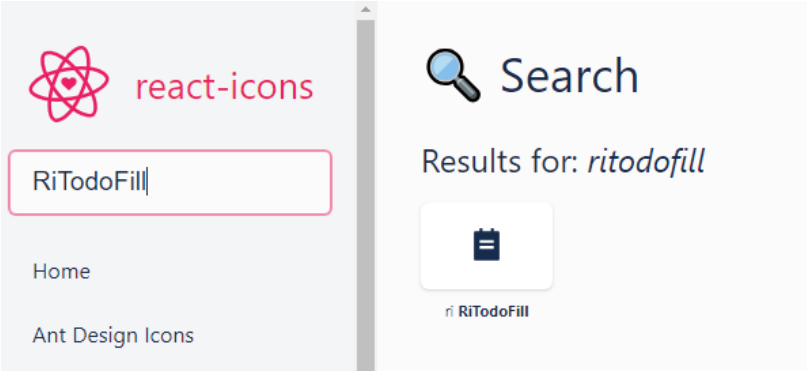
Добавление иконки возле каждой задачи



https://www.npmjs.com/package/react-icons

https://react-icons.github.io/react-icons/icons?name=ri

Найдем RiTodoFill



Добавим React-icons в проект путем инсталляции его в качестве зависимостей (!!!Предварительно остановить сервер). А затем импортировать нужную иконку.

```
npm install react-icons
```

```
npm start
```

Выполним импорт компонента RiTodoFill и добавим его в наше приложение

Todo.js M | Todo.module.css M

24-todo-app-v1 > src > components > Todos > Todo.js > Todo

```
1  import { RiTodoFill } from 'react-icons/ri';
2  import styles from './Todo.module.css';
3
4  function Todo({ todo, index, deleteTodo }) {
5    return (
6      <div className={styles.todo} onDoubleClick={() => deleteTodo(index)}>
7        <RiTodoFill className={styles.todoIcon} />
8        <div className={styles.todoText}>{todo}</div>
9      </div>
10   );
11 }
12
13 export default Todo;
```


Todo.module.css M
Todo.js M
24-todo-app-v1 > src > components > Todos > Todo
1 .todo {
2 display: flex;
3 align-items: center;
4 padding: 10px 20px;
5 margin: 15px 0;
6 font-size: 1.5rem;
7 border-radius: 5px;
8 border: 2px solid #555;
9 color: #112d49;
10 background-color: #fbfef9;
11 user-select: none;
12 }
13
14 .todoText {
15 width: 100%;
16 text-align: left;
17 }
18 .todoIcon {
19 font-size: 1.8rem;
20 margin-right: 10px;
21 color: teal;
22 }