Урок 3.

В данных нам обязательно нужны ID. Вручную их прописывать опасно, т.к. они могут повториться. Чтобы не повторялись установим библиотеку, которая будет генерировать ID:

**yarn add uuid** (или npm install uuid )

А также надо установить TypeScript типы для этого пакета:

**yarn add @types/uuid** (или npm install @types/uuid)

После этого мы можем использовать функцию **v1()**, также нужно сделать импорт:

import {v1} from 'uuid';

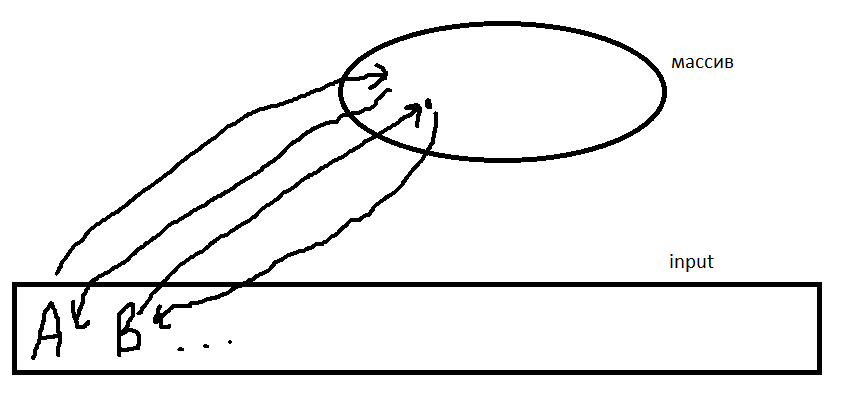
При вызове функции v1() мы получим значение типа строка 0ceae3d0-b309-11eb-add6-85e39954a937.

Тип данных id: **STRING, т.к. мы вызываем функцию, которая возвращает строку.**

**Концепция контролируемых INPUT**

**Идея такая:** если пользователь введет хоть какой-то символ, мы его сразу пишем в хранилище и возвращаем в инпут. Таким образом, инпут будет показывать только сохраненные данные.

Когда пользователь нажмет кнопку Добавить, данные отправятся в массив



**Сделаем добавление таска по нажатию на Click**

- создадим в компоненте TodoList стейт

const [title, setTitle] = useState<string>('')

- добавим в input value={title} из локального стейта. Т.е. закидываем данные из стейта в инпут

<input value={title}/>

- теперь нужно сделать так, чтобы инпут закидывал данные в стейт при изменении поля. Для этого добавим слушатель onChange, который будет забирать значение value у currentTarget

<input value={title} onChange={(e)=>{setTitle(e.currentTarget.value)}}/>

заносим значение переменной title в кнопку для добавления таска:

<button onClick={() => props.addTask(title)}>add task</button>

- для красоты будем очищать инпут. Для этого модифицируем вызов функции onClick на кнопке (создадим функцию onClickAddTask в компонене TodoList). В ней сначала забираем значение task, затем обнуляем значение task (все отправляем в useState):

const onClickAddTask = ()=> {  
 props.addTask(title)  
 setTitle('')  
}

и поменяем коллбек на клике кнопки:

Было:

<button onClick={() => props.addTask(title)}>add task</button>

стало:

<button onClick={onClickAddTask}>add task</button>

**Если добавлять функцию сразу в обработчик, то типизировать ее не надо. Но если функцию вынести отдельно, то надо!**

**Сделаем добавление таска по нажатию на ENTER**

* используем обработчик события **onKeyPress** на input

**было:**

<input   
 value={title}   
 onChange={(e) => {setTitle(e.currentTarget.value)}}   
 />

**стало:**

<input   
 value={title}   
 onChange={(e) => {setTitle(e.currentTarget.value)}}   
 **onKeyPress={(e) => {if (e.key === 'Enter') {onClickAddTask()}}}**  
/>

Для красоты вынесем функцию добавления по нажатию в отдельную функцию. Все функции нужно создавать в компоненте!

**Если добавлять функцию сразу в обработчик, то типизировать ее не надо. Но если функцию вынести отдельно, то надо!**

Создадим две отдельные функции и сделаем типизацию e (KeyboardEvent нужно добавить React):

const onChangeTitle =  
 (e: ChangeEvent<HTMLInputElement>) => {  
 setTitle(e.currentTarget.value)  
 }  
  
const onPressTitle =  
 (e: React.KeyboardEvent<HTMLInputElement>) => {  
 if (e.key === 'Enter') {  
 onClickAddTask()  
 }  
 }

и внесем названия функций в input:

<input type="checkbox" checked={task.isDone}/>  
 <span>{task.title}</span>  
 {/\*<button onClick={RT}>X</button>\*/}  
 <button onClick={function () {  
 props.removeTask(task.id)  
 }}>X  
 </button>  
</li>