УДК 004

Создание TODO приложения на React.JS

Халиманенков Андрей Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается создание приложения TODO, в котором можно создавать, удалять и редактировать задачи для выполнения. Можно сказать, что это приложение является аналогом записной книжки.
Ключевые слова: todo, разработка сайтов, JavaScript, HTML, CSS, UI, ReactJS.

Creating a TODO application on React.JS

Khalimanenkov Andrey Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the creation of a TODO application in which you can create, delete and edit tasks. This application is an analogue of a notebook.
Keywords: todo, website development, JavaScript, HTML, CSS, UI, ReactJS.

Темп современной жизни достаточно высок, отчего люди часто забывают о своих небольших, но важных делах. Раньше использовали записные книжки, чтобы записывать туда весь свой список задач на сегодняшний день. Но в текущих реалиях этот способ кажется несколько устаревшим в виду наличия у людей компьютеров и смартфонов. Теперь роль записной книжки играют приложения или сайты, куда можно записывать дела. Такие сервисы называются общим термином «TODO». Написание TODO является хорошим способом изучить основы языка и выбранного фреймворка. Создание подобного приложения с помощью JavaScript и React будет рассмотрено в этой статье.

Цель исследования – разобрать важные моменты создания TODO приложения на JavaScript и React.

Вопрос разработки веб-приложений волнует некоторых исследователей и специалистов: Е. В. Пантелеева [1] отразила сущность разработки сайта с использованием языка разметки HTML, особенности таблицы стилей CSS и языка программирования JavaScript. Н. Д. Лушников и А. Д. Альтерман [2] рассмотрели основы (технические возможности) каскадных таблиц стилей CSS. Кроме того, освещены главные преимущества и принцип работы каскадных таблиц. Н. О. Айдарбаев [3] раскрыл понятие адаптивного дизайна

как одного из процессов веб разработки. Дал определения разновидностей фронтэнд фреймворков, используемых в веб разработке, и подробный анализ компонентов фреймворка Bootstrap. В. Е Селькин [4] в своей статье оценил эффективность модульного принципа на предмет временной задержки, которой обладают многосоставные приложения. C. Staff [5] разобрал путь Facebook при создании фреймворка React. E. Wohlgethan [6] сравнил три основных JS фреймворка в лице React, Angular и Vue. Привёл примеры их использования.

Для разработки веб-приложения использовался язык JavaScript [7] и библиотека React [8].

Главная страница приложения выглядит следующим образом (рис. 1).

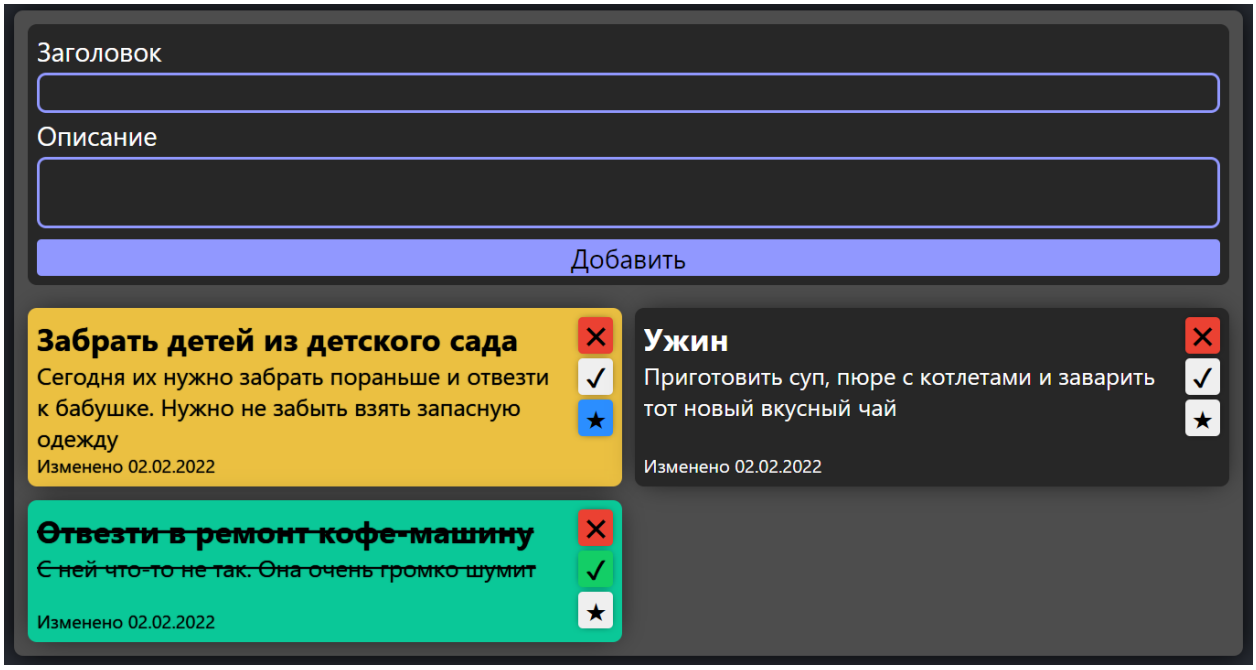

Рисунок 1 – Главная страница приложения

Важные задачи помечаются жёлтым фоном. За это отвечает кнопка со звёздочкой.

За рендер главной страницы приложения отвечает компонент Todo.jsx. Он возвращает следующие компоненты:

```
<div className={styles.todoApp}>
 <TodoAddForm
 />
 <TodoList
 />
</div>
```

TodoAddForm – компонент с формой для добавления новой задачи, а TodoList – компонент со списком задач.

Главным методом TodoAddForm является функция handleSubmit, которая по нажатию кнопки «Добавить» делает следующее:

1. Принимает событие «submit»;

2. Проводит валидацию формы, а именно input заголовка и textarea описания, на ограничение заданной длины текста;

3. Проверяет на наличие задачи с таким же заголовком и если такая существует, то выводит сообщение о невозможном создании одинаковых задач;

4. Формирует объект задачи основываясь на тексте, введённом в заголовок и описание. «new Date().toLocaleDateString()» обращается к глобальному объекту даты и берёт текущие число, месяц и год. Новая задача является неважной и невыполненной, о чём говорит значение false в ключах isDone и isImportant;

5. Добавляет объект новой задачи в общий массив задач с помощью функции addTask.

Код handleSubmit:

```
const handleSubmit = (e) => {
  e.preventDefault();
  //проверка на существование задачи с таким же тайтлом
  if (store.tasks.some(task => task.title.toLowerCase() === e.target.title.value.toLowerCase())) {
   return alert(`У вас уже есть задача с таким заголовком`);
  }
  //максимальные длины текста в заголовке и в описании
  const maxTextInputLength = 500;
  const maxTextAreaLength = 3000;

  if (!(e.target.title.value && e.target.description.value)) {
   return alert('Вам нужно написать что-нибудь в заголовок или       описание');
  }

  if (e.target.title.value.length <= maxTextInputLength && e.target.description.value.length <=
maxTextAreaLength) {
    const title = e.target.title.value;
    const description = e.target.description.value;
    const editedDate = `${new Date().toLocaleDateString()}`;
    const id = Date.now();
    const task = {
     title,
     description,
     isDone: false,
     isImportant: false,
     editedDate,
     id,
    }
    store.addTask(task);
    setTitleText('');
    setDescriptionText('');
  } else {
    alert(`You can write no more than ${maxTextInputLength} characters in Title and
${maxTextAreaLength} in Description sections`);
  }
 }
```

Компонент TodoList возвращает список задач с помощью метода массива «map». За вывод на экран каждой задачи из общего списка отвечает компонент TodoTask:

```
store.tasks.map(task => (
 <TodoTask
 key={task.id}
 {...task}
 />
))
```

Атрибут key помогает библиотеке React выполнять корректный рендер приложения при изменении элементов массива задач в state-хранилище.

Изначальный массив задач состоит из трёх объектов, в каждом из которых есть следующие ключи:

1. title - хранит в себе строку с заголовком задачи;
2. description – хранит в себе описание задачи;
3. isDone – true если задача выполнена и наоборот;
4. isImportant – true если задача важная и наоборот;
5. editedDate – время добавления или изменения задачи;
6. id – идентификатор задачи;

TodoTask содержит в себе текст заголовка и описания, а также дату изменения/создания задачи и пункт меню с кнопками для управления состоянием задачи – удалить задачу из списка, пометить выполненной и сделать важной. Компонент возвращает следующую JSX-разметку:

```
<li className={styles.taskContainer}>
 <div className={taskClassName}>
  <div className={styles.taskInfo}>
   <div className={titleTaskClassName}>
    {props.title}
   </div>
   <div className={descriptionTaskClassName}>
    {props.description}
   </div>
   <div className={styles.taskDate}>
    {`${language.task.edited} ${props.editedDate}`}
   </div>
  </div>
  <div
   className={styles.taskControl}
   onClick={(e) => e.stopPropagation()}
  >
   <DeleteButton
    id={props.id}
   />
   <DoneButton
   id={props.id}
    isDone={props.isDone}
   />
   <ImportantButton
    id={props.id}
    isImportant={props.isImportant}
   />
  </div>
 </div>
</li>
```

В компоненты DeleteButton, DoneButton и ImportantButton передаются id задачи для того, чтобы соответствующие функции могли выполнить свой алгоритм относительно той задачи, в теле которой они находятся. Атрибуты isDone и isImportant помогают кнопкам принимать CSS стили в зависимости от состояния задачи.

Компонент DeleteButton содержит в себе кнопку для удаления задачи и возвращает следующую разметку:

```
<button
 className={styles.deleteButton}
 onClick={() => store.deleteTask(props.id)}
>✕</button>
```

Функция deleteTask находит текущую задачу по ID в общем массиве задач и фильтрует массив, отсеивая задачу, которую нужно удалить. Возвращает следующую разметку:

```
deleteTask(id) {
 this.tasks = this.tasks.filter(task => task.id !== id);
}
```

Компонент DoneButton содержит в себе кнопку для изменения статуса «выполнено» задачи и возвращает следующую разметку:

```
<button
 className={DoneTasksClassName}
 onClick={() => store.isTaskPropertyToggle(props.id, 'isDone')}
>✓</button>
```

Компонент ImportantButton содержит в себе кнопку для изменения статуса важности задачи и возвращает следующую разметку:

```
<button
 className={ImportantTasksClassName}
 onClick={() => store.isTaskPropertyToggle(props.id, 'isImportant')}
>★</button>
```

У этих двух компонентов по клику кнопки срабатывает одинаковая функция, но в каждом элементе второй параметр функции различается. Это сделано для удобства, т.к. ключи isDone и isImportant являются булевыми значениями, а значит функция для переключения будет абсолютно одинаковая, за исключением указанного ключа в объекте задачи. Код функции isTaskPropertyToggle:

```
isTaskPropertyToggle = (id, property) => {
 this.tasks = this.tasks.map(task => {
  if (task.id === id) {
   return {
    ...task,
    [property]: !task[property],
   }
  }
  return task;
 });
}
```

Аргумент property принимает строку, чтобы определить ключ, который нужно изменить. Код «!task[property]» изменяет true на false и наоборот.

По итогу, в этой статье были рассмотрены базовые принципы создания простого ToDo приложения для браузера, в котором есть форма для внесения новых задачи и список всех задач. Он формируется на основе массива задач, в котором находятся объекты всех дел. В каждой задаче присутствуют кнопки для удаления задачи, изменения её статуса «выполнено» и важности. Каждый отдельный элемент списка реагирует на последние два статуса и меняет цвет фона на жёлтый, если задача важная и на зелёный, если задача выполнена.

Библиографический список

1. Пантелеева Е. В. Разработка сайта с использованием языка разметки HTML, таблицы стилей CSS и языка программирования JavaScript. // Информационные системы и технологии в образовании, науке и бизнесе. 2020. С. 94-96
2. Лушников Н. Д., Альтерман А. Д. Основы каскадных таблиц стилей (CSS). // Наука и образование: новое время. 2019. №. 1. С. 69-72.
3. Айдарбаев Н. О. Адаптивный дизайн веб-сайта с использованием фронтэнд-фреймворка Bootstrap // Молодой ученый. 2018. №. 21. С. 115-119
4. Селькин В. Е. Временной анализ модульной сборки пользовательского интерфейса на JavaScript //Студенческая наука XXI века. 2016. №. 2-1. С. 279-281.
5. Staff C. React: Facebook's functional turn on writing Javascript //Communications of the ACM. 2016. Т. 59. №. 12. С. 56-62.
6. Wohlgethan E. SupportingWeb Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js : дис. – Hochschule für Angewandte Wissenschaften Hamburg, 2018.
7. JavaScript URL: https://ru.wikipedia.org/wiki/JavaScript (дата обращения: 01.02.2022).
8. React URL: https://ru.reactjs.org (дата обращения: 01.02.2022).