УДК 004.415.25

СОЗДАНИЕ ИНДИКАТОРА ПРОГРЕССА ПОДСЧЕТА СИМВОЛОВ

Д. А. ИВАНОВ
(Представлено: д-р техн. наук, проф. С. Г. ЕХИЛЕВСКИЙ)

В данной статье рассматривается создание индикатора прогресса подсчета символов с готовым исходным кодом и всеми этапами его написания.

Введение. В данной статье будет рассмотрено создание индикатора выполнения подсчета символов в React JS.

Обычно это используется в приложениях для заметок и списка задач, где вы хотите ограничить ввод пользователя фиксированным количеством символов. Мы можем визуализировать это, чтобы улучшить взаимодействие с пользователем, используя линейный индикатор выполнения.

Создание текстовой области. Создадим простую текстовую область в новом приложении для реагирования. Формат App.js представлен в листинге 1.1:

Листинг 1.1. – App.js

```
import { React, useState } from "react";
import "./styles.css";
export default function App() {
  return (
    <div className="App">
      <textarea cols="20" rows="5"></textarea>
      <div className="progress">
        <span className="charLeft"> characters left</span>
      </div>
    </div>
  );
}
```

Данный код представляет собой простое приложение для реагирования с текстовой областью и div, содержащим диапазон, который будет показывать количество оставшихся символов.

Ограничение количества символов. Максимальное количество символов, которые может ввести пользователь, можно легко установить с помощью maxLength textarea следующим образом:

Листинг 1.2. – ограничение количества символов

```
<textarea maxLength="100" cols="20" rows="5"></textarea>.
```

Сохранение введенного текста. Затем нам нужно сохранить ввод, который пользователь вводит в состояние, чтобы мы могли использовать его для индикатора выполнения и диапазона. Мы создадим простое состояние и функцию onchange, которая будет обновлять его каждый раз, когда пользователь что-то вводит.

Листинг 1.3. – App.js

```
import { React, useState } from "react";
import "./styles.css";
export default function App() {
  const [input, setInput] = useState("");
  const inputHandler = (e) => {
    setInput(e.target.value);
  };
  return (
    <div className="App">
      <textarea
        maxLength="100"
        cols="20"
        rows="5"
        onChange={inputHandler}
```

```
        ></textarea>
        <div className="progress">
          <span className="charLeft">
            characters left
          </span>
        </div>
      </div>
    );
  }
```

Отображение оставшихся символов. Теперь нам нужно отобразить количество оставшихся символов, которое будет равно 100 — длине ввода.

Листинг 1.4. – Отображение оставшихся символов

```
<span className="charLeft">
   {100 - input.length} characters left
</span>
```

Создание индикатора выполнения. Для линейного индикатора выполнения мы можем использовать индикаторы выполнения материала пользовательского интерфейса. Для этого сначала установите mui: npm install @mui/material.

Далее нам нужно импортировать компонент линейного прогресса:

Листинг 1.5. – импорт компонента линейного прогресса

```
import LinearProgress from "@mui/material/LinearProgress";
```

Значение или «прогресс» бара определяется значением prop, а тип бара определяется, который назначается вариантным реквизитом.

Листинг 1.6. – «прогресс» бара

```
<LinearProgress
  className="charProgress"
  variant="determinate"
  value={input.length}
/>
```

Заключение. Было визуализировано ограничение пользователя фиксированным количеством символов ввода для улучшения взаимодействия с пользователем, используя линейный индикатор выполнения.

ЛИТЕРАТУРА

1. Программирование на React. [Электронный ресурс].- Режим доступа: https://ru.wikipedia.org/wiki/React. - Дата доступа: 12.10.2022
2. Основы React. [Электронный ресурс].- Режим доступа: https://habr.com/ru/company/ruvds/blog/343022/. - Дата доступа: 12.10.2022