

React: Formik, Yup

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

Повторим;)

■ Какие аргументы принимает `useEffect`?

■ Что нужно передать в `useEffect` для привязки его к этапу цикла `update`?

■ Что возвращает `useState`?

ЦЕЛЬ

Изучить работу с формой с помощью библиотеки Formik

ПЛАН ЗАНЯТИЯ

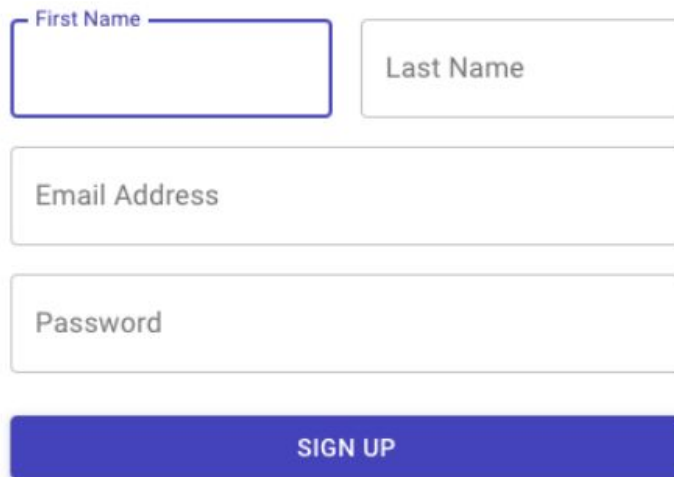
- 1. Знакомство с Formik
- 2. Изучение валидации с помощью Yup
- 3. Разбор проекта

Formik



Formik - это библиотека для управления состоянием форм в React-приложениях.

Она облегчает процесс создания и управления формами, предоставляя удобный интерфейс для работы с состоянием, отправкой данных и другими аспектами форм.



A registration form with the following fields and a button:

- First Name (text input)
- Last Name (text input)
- Email Address (text input)
- Password (text input)
- SIGN UP (button)

Создание формы с помощью formik

Шаг 1. Установка Formik

```
npm install formik
```

Шаг 2. Импортуем хук useFormik для создания формы

```
import { useFormik } from 'formik';
```


Создание формы с помощью formik

Шаг 3. Внутри компонента создаём настройку формы с помощью useFormik, в которой прописываем начальные значения и действие на submit

КОМПОНЕНТ

```
interface FormValues {  
  firstName: string;  
  lastName: string;  
  email: string;  
}
```

ТИПИЗАЦИЯ

```
const MyForm = () => {  
  const formik = useFormik({  
    initialValues: {  
      firstName: '',  
      lastName: '',  
      email: '',  
    } as FormValues,  
    onSubmit: (values: FormValues) => {  
      console.log(values);  
    },  
  });
```

Создание формы с помощью formik

Примечание: useFormik возвращает объект, который содержит несколько полезных свойств и методов для управления формой

1) values:

Это объект, содержащий текущие значения полей формы.

Мы используем его для установки значений в поля ввода и для доступа к значениям при отправке формы.

```
<input
  type="text"
  id="firstName"
  name="firstName"
  onChange={formik.handleChange}
  onBlur={formik.handleBlur}
  value={formik.values.firstName}
/>
```

Создание формы с помощью formik

2) handleChange:

Это функция, которую мы передаем в событие onChange каждого поля формы.

Она автоматически обновляет значения в объекте values.

```
<input  
  type="text"  
  id="firstName"  
  name="firstName"  
  onChange={formik.handleChange}  
  onBlur={formik.handleBlur}  
  value={formik.values.firstName}  
/>
```

Создание формы с помощью formik

3) handleBlur:

Это функция, которую мы передаем в событие **onBlur** каждого поля формы. Она помечает поле как "прикоснутое" (touched), что полезно в будущем для валидации.

```
<input
  type="text"
  id="firstName"
  name="firstName"
  onChange={formik.handleChange}
  onBlur={formik.handleBlur}
  value={formik.values.firstName}
/>
```

Создание формы с помощью formik

4) handleSubmit:

Это функция, которая вызывается при отправке формы. Мы передаем ее в событие onSubmit формы.

```
<form onSubmit={formik.handleSubmit}>  
  {/* Форма */}  
</form>
```

Другие полезные свойства можно узнать сделав `console.log(formik)`

Создание формы с помощью formik

5. Привязываем необходимые поля к формику и добавляем действие при submit формы

см. продолжение



```
return (  
  <form onSubmit={formik.handleSubmit}>  
    <label htmlFor="firstName">Имя:</label>  
    <input  
      type="text"  
      id="firstName"  
      name="firstName"  
      onChange={formik.handleChange}  
      onBlur={formik.handleBlur}  
      value={formik.values.firstName}  
    />  
  
    <label htmlFor="lastName">Фамилия:</label>  
    <input  
      type="text"  
      id="lastName"  
      name="lastName"  
      onChange={formik.handleChange}  
      onBlur={formik.handleBlur}  
      value={formik.values.lastName}  
    />  
  </form>  
)
```

Создание формы с помощью formik

5. Привязываем необходимые поля к формуле и добавляем действие при submit формы

```
<label htmlFor="email">Email:</label>
<input
  type="email"
  id="email"
  name="email"
  onChange={formik.handleChange}
  onBlur={formik.handleBlur}
  value={formik.values.email}
/>

<button type="submit">Отправить</button>
</form>
);
};

export default MyForm;
```

Валидация форм - Yup



Yup - это библиотека для валидации данных в JavaScript и TypeScript. Она обеспечивает простой и гибкий способ определения и применения правил валидации



Установка Yup:

Установите Yup с помощью npm

```
npm install yup
```

Добавьте Yup в свой компонент с формой

```
import * as Yup from 'yup';
```

Основные концепции Yup:

Схемы (Schema):


- В центре Yup находятся схемы, которые определяют правила валидации для объектов данных.
- Схемы могут быть вложенными, и они предоставляют множество методов для определения правил для каждого свойства объекта.

```
const schema = Yup.object().shape({  
  name: Yup.string().required('Имя обязательно'),  
  age: Yup.number().positive('Возраст должен быть положительным  
    числом').integer('Возраст должен быть целым числом'),  
  email: Yup.string().email('Некорректный формат  
    email').required('Email обязателен'),  
});
```

Основные концепции Yup:

Методы для определения объекта валидации

- **object()**: Определение типа объекта.
- **shape()**: Определение структуры объекта.



```
const schema = Yup.object().shape({  
  name: Yup.string().required('Имя обязательно'),  
  age: Yup.number().positive('Возраст должен быть положительным  
    числом').integer('Возраст должен быть целым числом'),  
  email: Yup.string().email('Некорректный формат  
    email').required('Email обязателен'),  
});
```

Основные концепции Yup:

Типы данных (Data Types):

- Yup предоставляет типы данных, такие как **string()**, **number()**, **boolean()**, **date()**, и т.д., которые используются при определении схем.

```
const schema = Yup.object().shape({  
  name: Yup.string().required('Имя обязательно'),  
  age: Yup.number().positive('Возраст должен быть положительным  
    числом').integer('Возраст должен быть целым числом'),  
  email: Yup.string().email('Некорректный формат  
    email').required('Email обязателен'),  
});
```

Основные концепции Yup:

Методы проверки (Validation Methods):

- Схемы в Yup определяются с использованием методов проверки, таких как **required()**, **min()**, **max()**, **matches()**, и другие.
- Эти методы устанавливают правила валидации для свойств данных.

```
const schema = Yup.object().shape({  
  name: Yup.string().required('Имя обязательно'),  
  age: Yup.number().positive('Возраст должен быть положительным  
    числом').integer('Возраст должен быть целым числом'),  
  email: Yup.string().email('Некорректный формат  
    email').required('Email обязателен'),  
});
```

Взаимодействие Yup с Formik

Для привязки валидации к Formik, созданную схему Yup добавляют к настройке формы через **useFormik** присваивая её свойству **validationSchema**

```
const formik = useFormik({  
  initialValues: {  
    name: '',  
    age: 0,  
    email: '',  
  },  
  validationSchema: schema, // Используем название schema  
  onSubmit: (values) => {  
    console.log(values);  
  },  
});
```

Взаимодействие Yup с Formik

Примечание: при создании схемы валидации с помощью **Yup**, можно сразу называть её **validationSchema** и в таком случае необязательно её писать. Настройка формика будет выглядеть следующим образом

```
const formik = useFormik({
  initialValues: {
    name: '',
    age: 0,
    email: '',
  },
  validationSchema, // Используем стандартное название validationSchema
  onSubmit: (values) => {
    console.log(values);
  },
});
```




Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH