[Lesson 8]

[DAN.IT EDUCATION]

[Что мы рассмотрим:]

- Контролируемые и неконтролируемые компоненты
- Базовое создание форм
- Создание форм с использованием redux-form

[DAN.IT] EDUCATION

[Не контролируемые компоненты]

Неконтролируемый компонент - это тот, который хранит свое собственное внутреннее состояние, и вы запрашиваете DOM с помощью ref, чтобы найти его текущее значение, когда Вы нуждаетесь в этом. Это немного больше похоже на традиционный HTML.

[Контролируемые компоненты]

Контролируемый компонент - это тот, который принимает текущее значение через ***props*** и уведомляет изменения через обратные вызовы, такие как ***onChange***. Родительский компонент "контролирует" его, обрабатывая обратный вызов и управляя своим собственным состоянием и передавая новые значения в качестве реквизитов контролируемого компонента. Вы также можете назвать это "немым компонентом".

Форма обратной связи

./src/components/Feedback/index.js

```
import React, { Component } from 'react'

export default class Feedback extends Component {
  render() {
    return (
      <div>
        <form>
            <input type="text" name="email" placeholder="Your e-mail..." />
            <textarea name="msg" placeholder="Message" />
            <input type="submit" name="submit" value="Send" />
        </form>
      </div>
    )
  }
}
```

[Форма обратной связи]

./src/components/Feedback/index.js

```
state = {
    email: ‘’,
    msg: ‘’,
}
onEmailChg = (event) => {
   this.setState({ email: event.target.value })
}
onMsgChg = (event) => {
   this.setState({ msg: event.target.value})
}
…
   <input type="text" value={this.state.email} onChange={this.onEmailChg} name="email" />
   <textarea name="msg“ value={this.state.msg} onChange={this.onMsgChg} />
   <input type="submit" name="submit" value="Send" />
…
```

[DAN.IT] EDUCATION

[Форма обратной связи]

./src/components/Feedback/index.js

```
state = {
    email: ‘’,
    msg: ‘’,
}

onChg = (event) => {
   this.setState({ [event.target.name]: event.target.value })
}

…
   <input type="text" value={this.state.email} onChange={this.onChg} name="email" />
   <textarea name="msg“ value={this.state.msg} onChange={this.onChg} />
   <input type="submit" name="submit" value="Send" />
…
```

[DAN.IT] EDUCATION

[Форма обратной связи]

./src/components/Feedback/index.js

```
…
     <select onChange={this.onChg} name=“department” value={this.state.department}>
            <option value=“support”>Support service</option>
            <option value=“billing”>Billing information</option>
     </select >
…
```

[Redux-form]

```
npm i redux-form --S
```

./src/reducers/index.js

```
import {combineReducers} from 'redux’;
import {reducer as formReducer} from 'redux-form';
import mails from './mails';

const rootReducer = combineReducers({
    mails,
    form: formReducer
});

export default rootReducer;
```

[DAN.IT] EDUCATION

[Redux-form]

./src/components/Feedback/FeedbackForm.js

```
import {Field, reduxForm} from 'redux-form’;
...
const {handleSubmit} = this.props;
return (
  <form onSubmit={handleSubmit}>
    <Field name="email" component="input"
      type="text" placeholder="Your e-mail...“  />
    <Field name="msg" component="textarea"
      type="textarea" placeholder="Message" />
    <button type="submit" label="submit">Send</button>
  </form>
);
...
export default reduxForm({ form: 'feedback', })(FeedbackForm);
```

[DAN.IT] EDUCATION

[Redux-form]

./src/components/Feedback/index.js

```
import React, { Component } from 'react'
import FeedbackForm from './FeedbackForm'

class Feedback extends Component {
  submit = values => {
    console.log(JSON.stringify (values));
  };

  render() {
    return (
      <div>
        <h1>Feedback</h1>
        <FeedbackForm onSubmit={this.submit} />
      </div>
    )
  }
}
export default Feedback;
```

[DAN.IT] EDUCATION

[Redux-form]

./src/components/Feedback/index.js

```
getInitialValues = () => {
  return {
    email: ‘test@email.com',
    msg: ‘Some user message',
  };
}
...

<FeedbackForm
  onSubmit={this.submit}
  initialValues={this.getInitialValues()}
/>
```

[DAN.IT] EDUCATION

[Client-Side Validation]

./src/validation/index.js

```
export const feedbackValidate = inputs => {
  const errors = {};
  if (!inputs.email) {
    errors.email = 'Введите ваш email';
  }

  if (!inputs.msg || !inputs.length) {
    errors.msg = 'Вы забыли ввести сообщение';
  }
  return errors;
};
```

[Client-Side Validation]

./src/components/Feedback/FeedbackForm.js

```
import {feedbackValidate} from '../../validation';

...

export default reduxForm(
    {
        form: 'feedback',
        validate: feedbackValidate
    })(FeedbackForm);
```

[Client-Side Validation]

./src/components/Feedback/myInput.js

```
import React, { Fragment } from 'react';

const myInput = (props) =>{
  const {input, type, placeholder, meta} = props;
  return (
     <Fragment>
       <input {...input} type={type} placeholder={placeholder}/>
       {meta.error && meta.touched && <div>{meta.error}</div>}
   </Fragment>
  )
}
export default myInput;
```

[DAN.IT] EDUCATION

[Client-Side Validation]

./src/components/Feedback/FeedbackForm.js

```
import myInput from ‘./myInput’;

...

<Field name="email" component={myInput}
            type="text" placeholder="Your e-mail..."  />
```

[Field-Level Validation]

./src/validation/index.js

```
...
export const requiredInput = (input) => {
  return input ? undefined : `Требуется ввод`;
}

export const correctInput = (input) => {
     return input.length < 3 ? 'Слишком короткий email' : undefined;
}
```

Field-Level Validation

./src/components/Feedback/FeedbackForm.js

```
import {requiredInput, correctInput} from '../../validation';

...

<Field name="email" component={myInput}
    type="text" placeholder="Your e-mail..."
       validate={[requiredInput, correctInput]}
/>
...

export default reduxForm({ form: 'feedback',})(FeedbackForm);
```

[Доступ к значениям других полей]

./src/components/Feedback/FeedbackForm.js

```
...
<Field name="confirm-email" component={myInput}
    type="text" placeholder="Your e-mail..."
       validate={[requiredInput, matchInput]}
/>
```

./src/validation/index.js

```
...
export const matchInput = (input, allInputs) => {
  return input === allInputs.email ? undefined :‘Emailы не совпадают';
}
```

[DAN.IT] EDUCATION

Валидация при Submit’e

./src/components/Feedback/index.js

```
import {SubmissionError} from 'redux-form’;

...

submit = (values) => {
  if(['test@test.com', 'some@test.ua’].includes(values.email)){
    throw new SubmissionError ({
      email: "Недавно с этого email'a уже была осуществлена отправка заявки",
    });
  }else{
    console.log(JSON.stringify (values));
  }
};
```

Когда использовать redux-form

- В вашем приложении большие и сложные формы
- В вашем приложении множество различных форм
- У вас есть сложные валидации касающиеся нескольких форм
- Введенные в форму данные необходимо связать со стором редакса