

Формы

Цель работы

Познакомить с работой с формами в React.

Задания для выполнения

1. В коде, где обновляется значение при вводе с помощью обработчика события `change` изменить способ получения введенных данных, с использованием атрибута **ref**.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <meta charset="utf-8" />
```

```
  <title>Формы в React</title>
```

```
</head>
```

```
<body>
```

```
  <div id="app"></div>
```

```
                                <script                                crossorigin  
src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
```

```
                                <script                                crossorigin  
src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></sc  
ript>
```

```
                                <script  
src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.25.0/babel.min.js  
></script>
```

```
  <script type="text/babel">
```

```

class UserForm extends React.Component {
  constructor(props) {
    super(props);
    var name = props.name;
    var nameIsValid = this.validateName(name);
    var age = props.age;
    var ageIsValid = this.validateAge(age);

    this.state = {name: name, age: age, nameValid: nameIsValid, ageValid:
ageIsValid};

    this.onNameChange = this.onNameChange.bind(this);
    this.onAgeChange = this.onAgeChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  validateAge(age){
    return age>=0;
  }
  validateName(name){
    return name.length>2;
  }
  onAgeChange(e) {
    var val = e.target.value;
    var valid = this.validateAge(val);
    this.setState({age: val, ageValid: valid});
  }
  onNameChange(e) {

```

```

var val = e.target.value;

console.log(val);

var valid = this.validateName(val);

this.setState( {name: val, nameValid: valid});
}

```

```

handleSubmit(e) {
  e.preventDefault();

  if(this.state.nameValid ===true && this.state.ageValid===true){
    alert("Имя: " + this.state.name + " Возраст: " + this.state.age);
  }
}

```

```

render() {
  // цвет границы для поля для ввода имени
  var nameColor = this.state.nameValid===true?"green":"red";

  // цвет границы для поля для ввода возраста
  var ageColor = this.state.ageValid===true?"green":"red";

  return (
    <form onSubmit={this.handleSubmit}>
      <p>
        <label>Имя:</label><br />
        <input type="text" value={this.state.name}
                                onChange={this.onNameChange}
style={{borderColor:nameColor}} />
      </p>
    </form>
  );
}

```

```

    <p>
      <label>Возраст:</label><br />
      <input type="number" value={this.state.age}
                                onChange={this.onAgeChange}
style={{borderColor:ageColor}} />
    </p>
    <input type="submit" value="Отправить" />
  </form>
);
}
}
ReactDOM.render(
  <UserForm name="" age="0" />,
  document.getElementById("app")
)
</script>
</body>
</html>

```

2. Загрузить созданную страницу на GitHub в репозиторий `tera`, используя формат в названии `Фамилия (латинскими буквами)_8`.

Методические указания

1. Для обращения к компонентам у каждого установите атрибут `ref::`

```

<NameField value="" ref="nameField" />
<AgeField value="5" ref="ageField" />

```

И в дальнейшем по значению атрибута мы можем ссылаться на эти компоненты, в том числе получать их состояние:

```

handleSubmit(e) {

```

```
e.preventDefault();  
var name = this.refs.nameField.state.value;  
var age = this.refs.ageField.state.value;  
if(this.refs.nameField.state.valid && this.refs.ageField.state.valid){  
    alert("Имя: " + name + " Возраст: " + age);  
}  
}
```

Контрольные вопросы

1. Атрибут **ref** может применяться к любому компоненту React?
2. Для чего служит атрибут **defaultValue**?

Дополнительные задания

1. Добавьте стили на страницу html.