1. В чем разница между контролируемыми и неконтролируемыми компонентами?

Ответ:Неконтролируемые компоненты действуют больше как традиционные элементы формы HTML. Данные для каждого элемента ввода хранятся в DOM, а не в компоненте. Вместо того, чтобы писать обработчик событий для всех ваших обновлений состояния, вы используете ref для получения значений из DOM.

В контролируемом компоненте данные формы обрабатываются состоянием внутри компонента. Состояние внутри компонента служит «единственным источником правды» для элементов ввода, которые отображаются компонентом.

2. Есть ли смысл использовать метод `shouldComponentUpdate()` в `PureComponent`?

Ответ:Метод shouldComponentUpdate() в PureComponent уже реализован таким образом, что выполняет поверхностное сравнение для определения необходимости перерисовки. Использование shouldComponentUpdate() в PureComponent может привести к избыточному коду и потенциальным ошибкам при неправильной реализации.

3. Будет ли перерисовываться данный компонент?

```
class PureComponent extends React.PureComponent {
  state = {
    item: {
                    count:0
          },
  }

  handleClick= () =>  {
    const item = this.state.item;
    item.count = this.state.item.count ++ ;
    this.setState({item});
  }
  render() {
    return <h2>{this.state.item.count}</h2>
  }
}
```

Ответ:Данный компонент PureComponent будет перерисовываться только при изменении его пропсов или состояния.

В данном случае, при каждом вызове handleClick значение count в объекте item увеличивается на 1 и затем обновляется состояние с помощью setState. Однако, сравнение ссылок в shouldComponentUpdate в PureComponent покажет, что состояние не изменилось, и компонент не будет перерисовываться.

4. Что будет, если чекбоксу не передать свойство '`checked`'?

Ответ:

Если чекбоксу в React не передать свойство checked, то чекбокс будет отображаться в исходном состоянии, которое устанавливается по умолчанию в зависимости от его defaultChecked (если оно задано) или false.

Без явного указания свойства checked, React не будет отслеживать состояние чекбокса и не будет обновлять его автоматически при изменении состояния компонента. В результате, если пользователь отметит или снимет отметку с чекбокса, React не будет знать об этом изменении и не отразит его в компоненте.

Если вам требуется контролировать состояние чекбокса и реагировать на его изменения, необходимо использовать свойства checked для установки начального значения и onChange для обработки события изменения состояния чекбокса.

5. В чем главное преимущество использования `PureComponent`?

Ответ: Главное преимущество использования PureComponent в React - автоматическая оптимизация перерисовки компонентов. PureComponent проверяет изменения в пропсах и состоянии компонента и прерывает перерисовку, если они остаются неизменными. Это улучшает производительность и упрощает код компонентов.

6. Что будет, если компоненту `input` передать метод `onChange`, но не передать `value`? А что будет, если компоненту `input` передать `value`, но не передать метод `onChange`?

Ответ: Если компоненту <input> в React передать метод onChange, но не передать value, то поле ввода станет неконтролируемым. Это означает, что значение в поле ввода будет управляться самим DOM-элементом <input>, и React не будет отслеживать или обновлять это значение. В этом случае, при изменении значения в поле ввода, вызовется метод onChange, но для доступа к текущему значению поля ввода нужно использовать другие способы, например, прямое обращение к DOM-элементу.

Если компоненту <input> передать value, но не передать метод onChange, то поле ввода будет иметь фиксированное значение, которое будет задано через value и не будет изменяться пользователем. Пользователь не сможет вносить изменения в поле ввода, так как отсутствует обработчик события onChange, который бы обновлял состояние компонента или выполнял другие действия при изменении значения поля ввода.

7. Как сделать из обычного `select` список с несколькими выбранными значениями (мультиселект)?

Ответ: В атрибут `value` можно передать массив, что позволит выбрать несколько опций в теге `select`

- `<select multiple={true} value={['Б', 'В']}>`

8. Напишите пример валидации текстового поля на React - чтобы оно было не пустым

Ответ: В компоненте возвращается форма с полем ввода типа "text", связанная с состоянием value и обработчиком handleChange. Если есть ошибка (error не пусто), выводится соответствующий параграф с сообщением об ошибке. При отправке формы вызывается handleSubmit, где происходит валидация и дальнейшие действия.

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

const TextField = () => {
  const [value, setValue] = useState('');
  const [error, setError] = useState('');

  const handleChange = (event) => {
    setValue(event.target.value);
    setError('');
  };

  const handleSubmit = (event) => {
    event.preventDefault();
```

```
    if (value.trim() === '') {
      setError('Поле не может быть пустым');
    } else {
      // Действия при успешной валидации
      console.log('Значение:', value);
      setValue('');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" value={value} onChange={handleChange} />
      {error && <p>{error}</p>}
      <button type="submit">Отправить</button>
    </form>
  );
};

export default TextField;
```

9. Приведите пример простейшей формы логина на сторонних компонентах (Formic, Material или Bootstrap на ваш выбор)

- **Ответ:**

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Formik, Field, Form } from 'formik';

const Basic = () => (
  <div>
    <h1>Sign Up</h1>
    <Formik
      initialValues={{
        firstName: '',
        lastName: '',
        email: '',
      }}
      onSubmit={async (values) => {
        await new Promise((r) => setTimeout(r, 500));
        alert(JSON.stringify(values, null, 2));
      }}
    >
      <Form>
        <label htmlFor="firstName">First Name</label>
        <Field id="firstName" name="firstName" placeholder="Jane" />

        <label htmlFor="lastName">Last Name</label>
        <Field id="lastName" name="lastName" placeholder="Doe" />

        <label htmlFor="email">Email</label>
        <Field
          id="email"
          name="email"
```

```
          placeholder="jane@acme.com"
          type="email"
        />
        <button type="submit">Submit</button>
      </Form>
    </Formik>
  </div>
);

ReactDOM.render(<Basic />, document.getElementById('root'));
```