1. Как вы думаете, чем плохо задание атрибута компонента `key` функцией Math.random()?

Ответ:Использование функции Math.random() для генерации ключей компонентов в React может привести к неожиданным результатам и проблемам с производительностью.
Во-первых, генерация случайных ключей может привести к созданию дубликатов, что может привести к непредсказуемому поведению компонентов. Это может произойти, например, если компоненты перерисовываются в случайном порядке, и ключи генерируются с использованием Math.random().
Во-вторых, генерация случайных ключей может затруднить оптимизацию производительности при рендеринге компонентов. Ключи помогают React понимать, какие элементы должны быть обновлены при изменении состояния компонентов. Если ключи генерируются случайно, то React не может использовать свои внутренние оптимизации для ускорения процесса рендеринга.

2. Как будет выглядеть этот пример, если мы кроме id будем передавать еще один параметр title?

Ответ:

```
onClick = (id, title) => (e) => {
  console.log('Действие на строке ' + id + ' с заголовком "' + title + '"');
  e.preventDefault();
  this.setState({ count: this.state.count + 1 });
};
render() {
  const { id, title } = this.props;
  const { count } = this.state;
  return <a href="#" onClick={this.onClick(id, title)}>click {count}</a>;
}
```

3. В чем отличие VirtualDOM от обычного DOM? Ответ:

- Эффективность: виртуальный DOM позволяет избежать ненужных операций с реальным DOM, таких как перерисовка или изменение стилей, и снизить затраты на производительность. Виртуальный DOM работает более эффективно, чем обычный DOM, потому что он выполняет меньше операций непосредственно с браузером, а также обладает улучшенными алгоритмами для определения, какие части страницы необходимо перерисовать.
- Рендеринг: VirtualDOM позволяет оптимизировать рендеринг, то есть отрисовку изменений на странице. Обычный DOM требует перерисовки всей страницы при любом изменении, что может привести к заметному снижению производительности и задержкам в работе пользовательского интерфейса.
- Абстрактность: VirtualDOM абстрагирует от браузера, что делает его более универсальным и позволяет использовать один и тот же код на разных платформах. Это также упрощает тестирование, потому что тестировщики могут имитировать виртуальный DOM, чтобы проверить правильность отрисовки.
- Управление состоянием: VirtualDOM позволяет управлять состоянием приложения с помощью реактивных компонентов, которые изменяются в соответствии с данными, и никогда не напрямую изменяют DOM.

4. В каком порядке выйдут сообщения в консоли и почему?

```
handleChange = () => {
  console.log('foo');
```

```
  this.setState({
        checked: !this.state.checked,
        },
        ()=>{console.log('baz')}
  );
  console.log('bar')
};I
```

Ответ:

- 'foo' - это сообщение выведется первым, так как вызов console.log('foo') находится перед вызовом setState.
- 'bar' - это сообщение выведется вторым, так как вызов console.log('bar') находится после вызова setState.
- 'baz' - это сообщение выведется третьим, так как функция обратного вызова, переданная в setState в качестве второго параметра, будет вызвана после того, как изменения в состоянии компонента будут выполнены.

5. Какую проблему решает использование рефов?

Ответ: Рефы позволяют получать доступ к DOM-элементам или компонентам, созданным в React-коде, и управлять ими через JavaScript. Их использование позволяет решать различные задачи, такие как управление фокусом, обработка событий пользователей, интеграция с внешними библиотеками и оптимизация производительности

6. Как вы думаете почему вызов методов ребенка из родительского компонента противоречит философии реакта?

Ответ: Вызов методов дочерних компонентов из родительского компонента нарушает принцип однонаправленного потока данных и усложняет отладку кода. Вместо этого в React рекомендуется использовать подъем состояния (Lifting State Up) для перемещения общего состояния в ближайший общий родительский компонент. Это упрощает код, повышает его читабельность и переиспользуемость, а также предотвращает неожиданные побочные эффекты.

7. Можно ли с помощью хука `useRef` передать `ref` дочерним элементам?

Ответ: Да, можно передать ref дочерним элементам с помощью хука useRef в React. Для этого нужно создать реф в родительском компоненте, а затем передать его как свойство дочерним компонентам. В дочернем компоненте можно получить доступ к рефу через свойство и использовать его для получения доступа к DOM-элементу или другому компоненту.

8. Что дает нам использование кастомных хуков?

Ответ: Использование кастомных хуков в React позволяет вынести повторяющуюся логику из компонентов в отдельные функции, которые можно переиспользовать в разных частях приложения. Это делает код более модульным и улучшает его читаемость и поддерживаемость.