1. Кратко опишите основные отличия `state` от `props`.

Ответ:Если `props` представляет входные данные, которые передаются в компонент извне, то state хранит такие объекты, которые создаются в компоненте и полностью зависят от компонента.Также в отличие от `props` значения в `state` можно изменять.

2. Какую ошибку я допустила в следующем коде и как ее исправить:

```
handleChange = () => {
  const chechedArr = this.state.arr; // [1,2,3]
                chechedArr.push(5); // [1,2,3,5]
  this.setState({
    arr: chechedArr,
  });
};
```

Ответ:в функции handleChange вы изменяете массив, который находится в состоянии компонента, напрямую в строке chechedArr.push(5). В React это не рекомендуется, так как изменение состояния напрямую может привести к неожиданному поведению компонента.
Чтобы исправить это, вам следует создать новый массив, содержащий изменения, а затем вызвать метод setState() с обновленным массивом. Вот как это можно сделать:

```
handleChange = () => {
  const updatedArr = [...this.state.arr, 5]; // создаем новый массив на основе текущего состояния и добавляем в него число 5
  this.setState({
    arr: updatedArr, // устанавливаем новый массив в качестве состояния компонента
  });
};
```

3. Можно ли повесить на один элемент несколько обработчиков событий?

Ответ: Да, на один элемент можно повесить несколько обработчиков событий. Для этого можно использовать несколько атрибутов событий в JSX.

4. Каким образом можно изменить `state`?

Ответ: Вызов `setState()` приводит к обновлению состояния компонента и вызову функции `render()` для перерисовки компонента в соответствии с новым состоянием.

5. Попробуйте предсказать какие сообщения будут в консоли и объяснить результат:

```
handleChange = () => {

  console.log(this.state.checked) //#1 false

  this.setState({

    checked: !this.state.checked, //true

  });

  console.log(this.state.checked) //#2 ?
```

};

Ответ:Вторая строка (#2) выведет значение свойства `checked` из состояния компонента после вызова метода `setState()`. Поскольку мы устанавливаем свойство `checked` в противоположное значение с помощью оператора `!`, то значение второй строки будет равно `true`.

6. Какими способами можно задать функцию `handleChange` и какой из них является самым правильным?

Ответ: В React есть несколько способов задать функцию-обработчик события `handleChange`, в зависимости от того, как вы определили компонент и как используете методы жизненного цикла.

1. Стандартный синтаксис функции: Но такой подход плохо работает в React
2. Синтаксис стрелочной функции, привязанной к свойству класса:
3. Прямо в вызове события, используя стрелочную функцию:

Этот подход удобен, если вам не нужно хранить функцию-обработчик в состоянии компонента.

4. Использование функции-обработчика как колбэка:

Этот подход предпочтительнее, когда вы работаете с большим количеством элементов, для которых требуется один и тот же обработчик.

5. Чем отличаются классовые и функциональные компоненты? Какие из них предпочтительнее в 2021 году?

Ответ: Классовые компоненты и функциональные компоненты - это два основных типа компонентов в React. Классовые компоненты наследуются от базового класса `React.Component`, имеют состояние и методы жизненного цикла, а также могут иметь методы. Функциональные компоненты - это простые функции, которые возвращают React-элементы и принимают пропсы в качестве аргументов. В 2021 году предпочтительнее использовать функциональные компоненты с хуками, так как они более простые, легче для чтения и понимания, а также могут быть более эффективными и быстрыми в работе.

6. Есть ли `this` в функциональных компонентах? Как можно получить к нему доступ?

Ответ: В функциональном компоненте нам недоступен `this`, поэтому мы не можем задать или считать состояние через `this.state`. Вместо этого мы вызываем хук `useState` напрямую изнутри нашего компонента. В `useState` мы передаем исходное состояние. Вызов `useState` вернёт пару значений: текущее состояние и функцию, обновляющую состояние

7. Можно ли использовать `props` и `state` одновременно?

Ответ: да

8. Где можно задать `state` без использования команды `this.setState`?

Ответ: В конструкторе компонента можно задать начальное значение для объекта состояния `this.state`.

9. Может ли состояние классового компонента не быть объектом? А функционального?

Ответ: классовом компоненте состояние должно быть объектом, в функциональном компоненте с использованием хука useState состояние может быть любого типа данных.

10. Способны ли функциональные компоненты самостоятельно хранить состояние?
Ответ: До появления хуков в React функциональные компоненты не могли хранить свое состояние самостоятельно и были ограничены только входными свойствами (props), которые передаются в компонент как аргументы. Однако с появлением хуков, таких как useState, функциональные компоненты могут теперь хранить и изменять свое состояние.

11. Как использовать хуки в классовых компонентах?
Ответ: Хуки не работают внутри классов, а используются вместо них.

12. А как задать начальное состояние `props`, если они еще не были переданы?
Ответ: В React предусмотрен способ устанавливать значения пропсов по умолчанию — `defaultProps`.