1. В компонент передаются атрибуты `description` и `title` . Могу ли я их сложить как на примере, чтобы получить одну строку и вывести в компоненте?

```
import React from "react";

import styles from './button.css'


export default class Example extends React.Component {

   render() {

                                    let {id,title} = this.props

                                    title += description;

       return (

          <span>{title}</span>

       );

   }

}
```

Ответ: Да, вы можете сложить атрибуты description и title вместе, чтобы получить одну строку. Однако, перед тем, как выполнить операцию сложения, вам необходимо убедиться, что у вас есть доступ к атрибуту description и он не является пустым значением.
Также, в коде, который вы предоставили, у вас нет доступа к атрибуту description. Если вы хотите использовать его в операции сложения, вам нужно добавить этот атрибут к вашим свойствам (props).

2. С помощью какого метода можно отловить изменение `props`?

Ответ: componentDidUpdate()

3. Оператор расширения часто используется также для клонирования объекта. Подумайте, чем отличаются эти две записи и какую проблему решает здесь оператор расширения:

```
const initialObj = { title:'Hello', text:'World' }


//№1

const firstObj = initialObj


//№2
```

const secondObj = {…initialObj}

Ответ: В первой записи создается ссылка на существующий объект, а во второй записи создается новый объект на основе существующего объекта. Использование оператора расширения позволяет избежать проблем с передачей по ссылке и изменением неожиданных свойств объекта.

4. В каком из методов жизненного цикла лучше всего использовать методы вызова API и обращения к веб-хранилищам, если они должны быть вызваны всего один раз при загрузке страницы?

Ответ: componentDidMount()

5. С помощью какого метода можно отловить и отрисовать для пользователя возникшую в компоненте ошибку?

Ответ: componentDidCatch

6. Какой код обычно пишут в конструкторе? Для каких задач он используется?

Ответ: Конструктор в классовом компоненте используется для инициализации состояния компонента и привязки методов к текущему экземпляру класса. В конструкторе можно использовать метод super() для вызова конструктора базового класса. В конструкторе также можно создавать ссылки на DOM элементы или на другие компоненты. Обычно конструктор содержит следующие типы действий:

- Инициализация состояния компонента с помощью метода this.state.
- Привязка методов к текущему экземпляру класса с помощью метода bind.
- Инициализация переменных и ссылок на другие компоненты или DOM элементы.

7. Что делает функция `render`()? Что может ее вызвать?

Ответ:Функция `render()` в классовом компоненте определяет, как компонент будет отображаться в пользовательском интерфейсе. Она вызывается автоматически React-ом при необходимости обновления пользовательского интерфейса, например, когда происходят изменения в состоянии компонента или его свойствах. Функцию `render()` нельзя вызвать вручную. Она должна возвращать корневой элемент JSX. Функция `render()` не должна изменять состояние компонента или выполнять другие побочные эффекты.

8. Что нужно изменить в коде из урока (видео), чтобы начальные параметры у компонента приходили из пропсов, но если пропсы вообще не заданы, начальные значения были инициализированы нулями?

```
class Timer extends React.Component {
        constructor(props) {
                super(props);
                this.state = {
                        minutes: 0,
                        seconds: 0
                };
        }
}
```

Ответ:

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      minutes: props.minutes || 0,
      seconds: props.seconds || 0
    };
  }
  // остальной код компонента
}
```

В этом коде мы используем значения из пропсов, если они были переданы, иначе устанавливаем значения по умолчанию, равные 0. Обратите внимание, что мы используем `props.minutes || 0` и `props.seconds || 0`, чтобы использовать значения из пропсов, если они определены, иначе установить значение по умолчанию равное 0.

9. Можно ли несколько раз использовать хук `useEffect` внутри одного компонента?

Ответ: Да, можно использовать несколько раз хук `useEffect` внутри одного компонента.

10. Можно ли не передавать второй аргумент в хук `useEffect`, что тогда произойдет?

Ответ: Да, можно не передавать второй аргумент в хук `useEffect`, в этом случае эффект будет запускаться при каждом изменении любой переменной в компоненте, включая состояние (state) и пропсы (props). В большинстве случаев необходимо передавать второй аргумент - массив зависимостей, чтобы определить, при изменении каких переменных в компоненте должен запускаться эффект. Если массив зависимостей пустой, то эффект будет выполнен только один раз, при монтировании компонента. Если в массиве зависимостей указаны конкретные переменные, то эффект будет запускаться только при изменении этих переменных.

11. Что означает возвращение функции в теле эффекта?

Ответ: Возвращение функции из тела эффекта (callback-функции) является опциональным и предназначено для выполнения очистки или отмены эффекта, созданного при предыдущих вызовах эффекта.

12. Будут ли перерисованы дочерние элементы компонента при вызове метода `forceUpdate`()?

Ответ:Да, при вызове метода `forceUpdate()` происходит перерисовка всего компонента, включая его дочерние элементы, даже если нет изменений в состоянии или пропсах компонента.