1. Что такое props и можно ли использовать props в функциональных компонентах?

Ответ: Во многом компоненты ведут себя как обычные функции JavaScript. Они принимают произвольные входные данные (так называемые «пропсы») и возвращают React-элементы, описывающие, что мы хотим увидеть на экране.
Эта функция — компонент, потому что она получает данные в одном объекте («пропсы») в качестве параметра и возвращает React-элемент. Мы будем называть такие компоненты «функциональными», так как они буквально являются функциями.

2. Нужно ли выделять в отдельный компонент статью в блоге?

Ответ:Можно применить принцип единственной ответственности: каждый компонент должен заниматься какой-то одной задачей. Если функциональность компонента увеличивается с течением времени, его следует разбить на более мелкие подкомпоненты.

3. Можно ли использовать React без JSX?

Ответ: JSX не является обязательным для работы с React. React можно использовать без JSX. Это особенно удобно, когда вы не хотите настраивать транспиляцию в процессе сборки.

4. Чем отличается JSX от HTML?

Ответ:
По большей части синтаксис и структура JSX и HTML совпадают, но есть некоторые важные различия:

- Так как это похожий на XML синтаксис, одиночные теги в JSX должны быть закрыты: <hr />.
- Вместо атрибута class в JSX используется имя свойства в DOM: className.

5. Для чего нам нужны свойства (props) компонентов?

Ответ: Props представляет коллекцию значений, которые ассоциированы с компонентом. Эти значения позволяют создавать динамические компоненты, которые не зависят от жестко закодированных статических данных.

6. В примере с `CardList` чем можно было бы заменить `<React.Fragment>`?

Ответ: вы можете обернуть код в пустые скобки `<></>`

7. Можно ли сказать, что классовые и функциональные компоненты равнозначны по функциональности?

Ответ: Во многих старых статьях про React вы увидите фразу, что функциональные компоненты используются, когда нет внутреннего состояния, только props. Но в версии 16.8 команда React ввела hooks (хуки), не только ставя функциональные компоненты вровень с классовыми, но также делая их более лёгкими в написании и даже потенциально превосходящими своих старших собратьев.

Совсем недавно, к функциональному компоненту относились как к "младшему брату" классового. Функциональный компонент умел только принимать свойства в одном объекте («пропсы») в качестве параметра и возвращать React-элемент.

8. Можно ли полностью описать приложение, используя только функциональные компоненты?

Ответ: новый код должен быть в функциональном стиле с применением хуков;
старый код может продолжать использовать компоненты класса, если нету желания его переписывать

9. Какой командой мы делаем экспорт компонента для возможности его использования в других местах приложения?

Ответ: render()

10. Изучите структуру компонент в проекте https://github.com/alisa-tsvetkova/EthereumUI и напишите, какой именно компонент является самым верхним, а какой - самым «глубоким"?

Ответ: <Router> , <Route>

11. Какой командой можно сгенерировать разметку/компоненты на основе заранее заданного массива элементов? Приведите пример

Ответ: map

```
function NumberList(        ) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li>{number}</li>
  );
  return (
    <ul>{listItems}</ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
const        =                  .createRoot(                .getElementById('root'));
      .render(<NumberList numbers={numbers} />);
```