Неделя 22 Домашняя работа

**Ответы на вопросы:**

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

props (пропсы) — это входные данные React-компонентов, передаваемые от родительского компонента дочернему компоненту. props предназначены только для чтения.

props можно использовать в функциональных компонентах. Параметр props, который передается в функцию компонента, инкапсулирует свойства объекта.

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

Да, лучше выделить.

1. Можно ли использовать React без JSX?

Можно, но лучше их использовать вместе.

1. Чем отличается JSX от HTML?

JSX выглядит как HTML, но под капотом он преобразуется в обычные объекты JavaScript.

JSX более строгий и имеет несколько больше правил, чем HTML.

Например:

* Чтобы не добавлять в разметку лишние <div>, вместо них можно написать <> и </>, так называемые «фрагменты». Фрагменты позволяют группировать элементы, не оставляя никаких следов в HTML-дереве браузера.
* JSX требует, чтобы теги были явно закрыты: самозакрывающиеся теги, такие как <img> должны стать <img />, а оберточные теги типа <li>oranges должны быть записаны как <li>oranges</li>.
* JSX превращается в JavaScript, а атрибуты, записанные в JSX, становятся ключами объектов JavaScript.
* Так как for — зарезервированное слово в JS, в React-элементах используется свойство htmlFor.
* Обычный HTML не очень безопасен. Любой текст, который должен оставаться текстом, необходимо экранировать перед выводом. Иначе если внутри содержится HTML, то он будет проинтерпретирован. Ситуация может стать опасной, если этот текст на сайт добавляют сами пользователи. JSX работает по-другому. Всё, что выводится обычным способом, безопасно по умолчанию и экранируется автоматически.
* Совсем по-другому работает атрибут style. Если в HTML это обычная строка, то в JSX только объект.
* Если свойство передаётся в компонент без значения, то оно автоматически становится равным true.

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

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

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

<> и </>

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

Да, с точки зрения React они эквивалентны. Однако за последние 3 года функциональные компоненты стали более компактными, простыми в написании и стали давать небольшую выгоду в производительности.

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

Да, можно.

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

export default

1. Изучи структуру компонент в проекте <https://github.com/alisa-tsvetkova/EthereumUI> и напиши, какой именно компонент является самым верхним, а какой самым «глубоким»?

Самый верхний компонент - blockViewer

Самый «глубокий» компонент - Table

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

С помощью map()

const TodoList = ({ post, title , value}) => {

return (

<div className="List">

<h1 style={{ textAlign: "center", }}>{title}</h1>

{ post.map((post) =>

<TodoItems post={post} key={post.id} value={value} />

)}

</div>

)

}

export default TodoList;