1. Props это объект, содержащий в себе всю передаваемую в компонент информацию, которую необходимо отразить на странице. Props могут использоваться в функциональных компонентах, содержа данные, указанные как атрибуты или вложенные элементы, и затем возвращаемые в элементах DOM.
2. Можно выделить, особенно если планируется использовать аналогичный формат такой статьи повторно, и менять только содержание.
3. Можно, c чистым JS, т.к. JSX обычно используется для удобства написания команд создания DOM-элементов и не является уникальным способом, отличным от JS
4. Атрибуты в JSX-выражениях прописываются в стиле camelCase, слешем можно закрывать пустые теги, можно регулировать экранирование содержания передаваемых в элемент данных.
5. Props нужен для создания набора данных, которые могут использоваться в компоненте. Получая извне данные об объекте, мы можем динамически изменять содержание выводимого элемента, используя инкапсулированные свойства этого объекта с помощью Props.
6. использовать не свойства, а импортировать fragment как компонент, либо пустые скобки <></>
7. До появления обновлений эти компоненты имели отличия, в классовых компонентах, например, можно было устанавливать объект состояния для интерактивных приложений, а в фукцион. – нет, далее, чтобы захватить какое-либо событие, в классовых комп. нужно использовать метод render, т.к. иначе входные данные могли бы изменяться в процессе реализации действия компонента и итог не зависел бы от первого переданного аргумента. При использовании нововведенных хуков такое неполезное поведение классовых компонентов может компенсироваться использованием функц. компонент, и в большинстве ситуаций они могут быть полными аналогами классовых, уменьшающие количество и качество кода.
8. на данный момент наличие хуков позволяет использовать функциональные компоненты практически для любых целей, предпочтение классовым компонентам возможно, например, при использовании предохранителей, т.к. только в них может быть встроена функция отлова ошибок в JS.
9. export default название компонента;
10. верхний: App, глубокий: Block
11. const НовыйМассив = старыйМассив.map(e => <component atribute1={ component.свойство} atribute2={ component.свойство } />)

return (

<div>

{ НовыйМассив }

</div>

)

}