Функциональная компонента — это буквально функция. Должна возвращать разметку в любом случае. Главное отличие в том, что когда функциональная компонента меняет свое состояние, то она перерисовывается полностью. Функциональную компоненту унаследовать не получится. Использование функциональных компонент в разметке других компонент аналогично классовым. Если в классовой компоненте props в конструкторе, то в функциональной компоненте это ее аргумент. Функциональные компоненты работают намного быстрее классовых. Для функциональных компонент применимо жесткое правило инкапсуляции, то есть вне функциональной компоненты в файле ничего не должно быть, кроме примитивных констант.

Hook’и это стандартные или самописные функции, которые расширяют и дополняют функционал функциональной компоненты.

import {useState} from ‘react’;

useState это функция которая возвращает массив из двух элементов, первый элемент это константа-значение текущего состояния, второй элемент это функция с помощью которого изменяется состояние. В аргумент функции можно передать начальное значение состояния.

const App = () => {

const [pageName, setPageName] = useState(‘Graph3D’);

return (<>

<Header setPageName = {setPageName}

{pageName ===’Graph3D’ && <Graph3D />}

…

</>;

}

useRef – возвращает ссылочный объект который прописывается в атрибут Ref HTML элемента. Далее его использование ничем не отличается. Handler’ы событий для разметки которые возвращют элементы разметки. Для того чтобы каждый раз не создавать обработчик события. Есть возможность запомнить созданную функцию с помощью useCallback. В этот хук передается функция которую необходимо запомнить и возвращает ссылку на эту же самую функцию.

const clickHandler = useCallback(()=>{…}, []);

Hook который может запомнить состояние – useMemo, возвращает ссылку.

useEffect – хук который вызывает перерисовку компоненты, в случае изменения зависимостей, которые в него переданы. Если в этот хук не переданы никакие зависимости, то он будет вызван один раз после завершения рендера разметки компоненты.

Первый аргумент функция, которая сработает когда useEffect сработает.

Второй аргумент это массив зависимостей.

useEffect’ов в компоненте может быть несколько, они могут быть с разными зависимостями. Если useEffect используется без зависимостей, то функция передаваемая в useEffect может возвращать еще одну функцию. Которая будет вызвана при уничтожении компоненты.

Запрещено хуки использовать друг в друге.

Нужно проинициализировать useEffect когда у нас появится разметка.

const Graph2D = () => {

let graph = null;

…

//callbacks можно достать сюда

useEffect(()=>{

graph = new Graph({…});

return () => {

graph = null;

}

});

}

Тоже самое касается Graph3D там же анимация, requestAnimationFrame и тд. const Calc = () => {

const aRef = useRef();

const bRef = useRef();

…

return (<>

<textarea ref = {aRef} />

<textarea ref = {bRef} />

</>);

}

Домашка: Необходимо переписать на функциональные компоненты: App, Header, Calc, Graph2D и Graph3D.

В math3D все фигуры должны быть переписаны на классы отнаследованные от surface.

class Cube extends Surface {

constructor(options) {

super(options);

const points = […];

…

this.points = points;

this.edges = edges;

}

}

Задание со звездочкой: Написать hook useGraph который возвращает массив из трех элементов:

1. Ссылка на граф
2. Callback для renderScene
3. Функция деинициализатор, которая останавливает requestAnimFrame и деинициализирует граф

В хуке useGraph requestAnimFrame нужно вызывать.

Хук useGraph должен быть написан в модулях и лежать рядом с графом.

Задание с 2 звездочками: Все css’ники переписать на .scss

css поддерживает несколько систем исчисления. На одном проекте должна быть строго применена одна система координат.