40. Хуки в React

Цель:

Познакомиться со следующими понятиями и возможностями react:

- что такое хуки
- основные хуки react

Хуки:

Хуки — нововведение в React 16.8, которое позволяет использовать состояние и другие возможности React без написания классов.

Особенности хуков:

- Хуки позволяют вам повторно использовать логику состояния, не затрагивая дерево компонентов. Благодаря этому, хуки легко использовать в разных компонентах и делиться ими с сообществом
- В классовых сложных компонентах часто можно было встретить смесь не связанной логики. Что приводило к сложной поддержке кода и наличию багов. Чтобы решить эту проблему, хуки позволяют разбить один компонент на маленькие функции по их назначению (например, подписке или загрузке данных), а не на основе методов жизненного цикла.
- Классовые компоненты были одним из барьеров для изучения реакта
- react с использованием хуков еще больше приближает нас к “функциональному” подходу

Хук состояния. useState:

React **useState** Hook позволяет нам отслеживать состояние в функциональном компоненте.

Состояние обычно относится к данным или свойствам, которые необходимо отслеживать в приложении.

React **useState** Hook принимает начальное состояние и возвращает два значения:

- Текущее состояние.
- Функция, обновляющая состояние.

```
const [season, setSeason] = useState("summer");

return (
  <>
    <h1>My favorite time of year is {season}!</h1>
    <button
      type="button"
      onClick={() => setSeason("winter")}
    >
      Winter
    </button>
  </>
);
```

Хук эффекта. useEffect:

useEffect(<function>, <dependency>)

React **useEffect** Hook позволяет выполнять побочные эффекты в ваших компонентах. Некоторые примеры побочных эффектов: выборка данных, прямое обновление DOM и таймеры.

```
useEffect(() => {
  let timer = setTimeout(() => {
    setCount((count) => count + 1);
  }, 1000);

  return () => clearTimeout(timer);
}, []);
```

Хук эффекта. useEffect:

useEffect(<function>, <dependency>)

<function> - функция, которая будет отрабатывать в зависимости от вида 2 аргумента (массива зависимостей)
<dependency> - массив зависимостей

Работа хука в зависимости от массива dependency:

1. **Зависимости не переданы:** функция будет выполняться при каждом рендере.
2. **Пустой массив:** хук отработает только после первого рендера.
3. **Массив зависимостей:** хук будет отрабатывать
    a. при первом рендере
    b. при изменении значений массива депенденси

Хук useCallback:

useCallback возвращает мемоизированный колбэк

```
const memoizedCallback = useCallback(
  () => {
    doSomething(a, b);
  },
  [a, b],
);
```

Это хук полезен при передаче колбэков оптимизированным дочерним компонентам, которые полагаются на равенство ссылок для предотвращения ненужных рендеров (например, shouldComponentUpdate)

Хук useMemo:

- Возвращает мемоизированное значение.

- Данный хук позволяет улучшать производительность. useMemo работает только тогда , когда один из его депенденси меняет значение.

```
const calculation = useMemo(
  () => expensiveCalculation(count),
  [count]
);
```