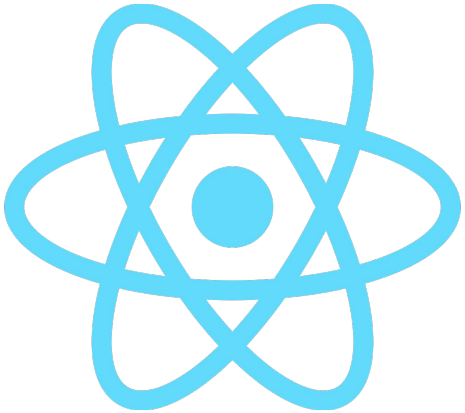
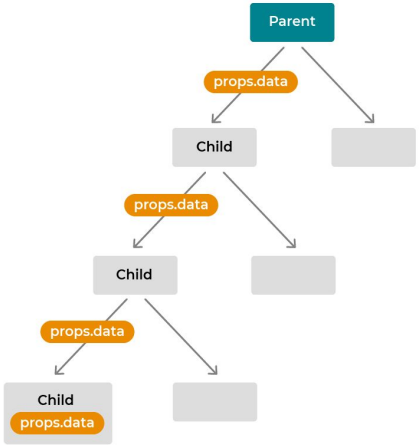


Тема занятия:

React:

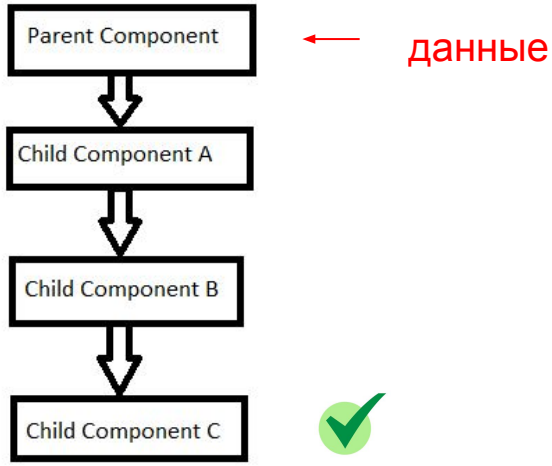
useContext

useContext


Parent
props.data
Child
props.data
Child
props.data
Child
props.data

useContext - это React Hook, предоставляющий способ получения значения из контекста в функциональных компонентах React.

Контекст в React используется для передачи данных глубоко вложенным компонентам без явной передачи пропс через каждый уровень.



Алгоритм создания

1. Создаём контекст в компоненте, который будет передавать

```
import { createContext, useState } from 'react';

// Создаем контекст с типом React.Context и значением по умолчанию
const MyContext = createContext<string>("");
```

создаём контекст

```
// Компонент-поставщик контекста
const MyContextProvider = ({ children }) => {
  const [sharedValue, setSharedValue] = useState('Значение из контекста')

  const updateValue = (newValue) => {
    setSharedValue(newValue);
  };

  return (
    <MyContext.Provider value={{ sharedValue, updateValue }}>
      {children}
    </MyContext.Provider>
  );
};

export { MyContext, MyContextProvider };
```

делаем компонент поставщиком контекста

Алгоритм создания

2. Соответствующий <Context.Provider> должен находиться выше компонента, которму нужнф данные из контекста. В данном случае MyContextProvider находится на самом верхнем уровне, но это необязательно

```
import React from 'react';
import { MyContextProvider } from './MyContextProvider';
import MyComponent from './MyComponent';

const App = () => {
  return (
    <MyContextProvider>
      <div>
        <h1>Мое React Приложение</h1>
        <MyComponent />
        {/* Другие компоненты в приложении */}
      </div>
    </MyContextProvider>
  );
};

export default App;
```

импортируем компонент с контекстом

Оборачиваем другие компоненты

Алгоритм создания

3. Передаём значение из компонента поставщика (провайдера) в другой компонент

```jsx
import React, { useContext } from 'react';
import { MyContext } from './MyContextProvider';

// Компонент, использующий значение из контекста с помощью useContext
const MyComponent = () => {
  const { sharedValue, updateValue } = useContext(MyContext);

  const handleButtonClick = () => {
    // Обновляем значение в контексте
    updateValue('Новое значение из компонента');
  };

  return (
    <div>
      <p>Значение из контекста: {sharedValue}</p>
      <button onClick={handleButtonClick}>Обновить значение</button>
    </div>
  );
};

export default MyComponent;
```

импортируем контекст

С помощью useContext получаем доступ к значению и функции для его изменения из компонента MyContextProvider

Когда значение в контексте обновляется в **MyContextProvider**, все компоненты, использующие **useContext(MyContext)**, автоматически получают новое значение, и их перерисовывается.