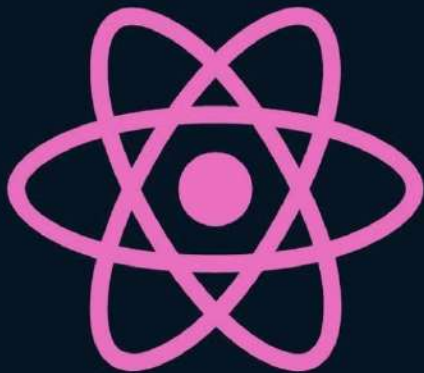




@pyplane_code

Context API



Introduction

Context API is a solution for a more elegant way of passing data between components by creating global variables, without the need to pass props down the component tree manually

TO GET STARTED:

create a react app:

```
npx create-react-app context-app
```

Add context folder with color-context.js inside

```
✓ context
  JS color-context.js
```

Add components folder with two sample files:

```
✓ components
  JS component-a.js
  JS component-b.js
```



What we are building

We will be able to access global color variable and change it directly from the components:

ComponentA



ComponentB



color-context.js

Let's create our global state inside the color-context.js file

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

export const ColorContext = createContext({
  selectedColor : null,
  onColorChange: (newColor) => {},
})

export const ColorContextProvider = ({children}) => {
  const [color, setColor] = useState('red')

  const handleColorChange = (c) => {
    setColor(c);
  }

  const contextValue = {
    selectedColor: color,
    onColorChange: handleColorChange,
  }

  return (
    <ColorContext.Provider value={contextValue}>
      {children}
    </ColorContext.Provider>
  )
}
```

Wrap our App

Now in order to use the color content in our application - we need to wrap our App with the ColorProvider

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import {ColorContextProvider} from './context/color-context';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <ColorContextProvider>
      <App />
    </ColorContextProvider>
  </React.StrictMode>
);
```

Component A

Let's take a look at an example how we could access and modify global state in component-a.js

```
import { useContext } from "react";
import { ColorContext } from "../context/color-context";

const ComponentA = () => {
  const colorCtx = useContext(ColorContext);
  const {selectedColor, onColorChange} = colorCtx;

  return (
    <>
      <h1 style={{color: selectedColor}}>
        hello world
      </h1>
      <button onClick={()=>onColorChange('green')}>
        change to green
      </button>
      <button onClick={()=>onColorChange('blue')}>
        change to blue
      </button>
    </>
  );
}

export default ComponentA;
```

Component B

Let's also take a look at a similar example where we could use our newly created color context in a different component

```
import { useContext } from "react";
import { ColorContext } from "../context/color-context";

const ComponentB = () => {
  const colorCtx = useContext(ColorContext)
  const {selectedColor, onColorChange} = colorCtx

  return (
    <>
      <div
        style={{
          backgroundColor: selectedColor,
          width: '100%', height:500
        }}
      >
      </div>
      <h3>selected color: {selectedColor}</h3>
      <button onClick={() => onColorChange('blue')}>
        switch to blue
      </button>
    </>
  );
}

export default ComponentB;
```