React Class vs Func Components

- class Bus extends React.Component {
  render() {
  return <h2>This is a bus.</h2>;
  }
  }
  ……..
- function Bus() { //function based component is popular
  return <h2>This is a bus.</h2>;
  }
  ……
  root.render(<Bus />);

useContext(child can access var)

- ```
  import { useState, createContext, useContext } from "react“
  import Child from "./Child";
  export const UserContext = createContext();
  export default function App13() {
  const [user, setUser] = useState("John");
    const [email, setEmail] = useState("john@gmail.com")
    return (
      <>
      <UserContext.Provider value={{user, email}}>
        <h2>Hello {user} from App13 component</h2>
        <Child />
      </UserContext.Provider>
      </>
    );
  }
  .....
  import {UserContext} from "./App13“
  import { useContext } from "react";
  export default function Child() {
    const {user, email} = useContext(UserContext);
    return (
      <>
        <h2>Hello {user} from Child component</h2>
      </>
    );
  }
  ```
  ```

Components Life Cycle

- **Mounting, unmounting and updating**
  comment strictmode to prevent rerender twice
- useEffect(() => {
    console.log("Component: Mounting");
    return () => {
      console.log("Component: Unmounting");
    };
  },[cnt,result]);

  ……

  function Result(props) {
    const r = props.r;
    if (r) {
      return <Pass/>;
    }
    return <Fail/>;
  }

useNavigate (react-router-dom)

```
import React from 'react’
import { useNavigate } from 'react-router-dom’
export default function Post() {
  const Navigate = useNavigate()
  const goToFeeds = () => {
    Navigate('/’)
  }
  return (
    <div>Post
      <button onClick={goToFeeds}>Go to Feeds</button>
    </div>
  )
}
```

React Memo

```
import React from 'react'
import { memo } from 'react' //will not rerender if count is same or
function ExpComp({count}) {
  console.log("ExpComp Rendered",count,"times")
  return (
    <div>ExpComp</div>
  )
}
export default memo(ExpComp)
export default function App12() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <ExpComp count={count} />
      <button onClick={() => setCount((prevState) => prevState + 1)}>
        {count}
      </button>
    </div>
  );
}
```

useMemo

```
import React from "react";
import { useMemo, useState } from "react";
export default function Main30() {
  const [count, setCount] = useState(0);
  const [flag, setFlag] = useState(10);
  const f1 = () => {
    let i;
    for (i = 0; i < flag ** 2; i++) {}
    console.log("result function called");
    return i;
  };
  //   const result = useMemo(() => f1(), [flag]);
  const result = f1();
  return (
    <div>
      <h1>useMemo</h1>
      <button onClick={() => setCount(count + 1)}>Count:{count}</button>
      <button onClick={() => setFlag(flag + 1)}>Flag:{flag}</button>
      <p>Result:{result}</p>
    </div>
  );
}
```

React Styling – Style Object

```
const Header = () => {
  const myStyle = {
    color: “red",
    backgroundColor: “Blue",
    padding: "12px",
    fontFamily: "Sans-Serif“
  };
  return (
    <>
      <h1 style={myStyle}>Hello World!</h1>
      <p>Heading</p>
    </>
  );
}
```

Using Children Props

```
import App19child from './App19child’
export default function App19() {
  return (
    <div>
       <App19child>
          <h1>Hello World</h1>
       </App19child>
    </div>
  )
}
.............................
export default function App19child(props) {
  return (
    <div style={{backgroundColor:'silver’}}>
       {props.children}
    </div>
  )
}
```

Adding Custom Env Variables

Create .env file in the root folder of the project

Every variable you define should start with REACT_APP_

Example : REACT_APP_PATH=/ecomm-react

Now you can use the variable in any of your components as shown below

const PATH = process.env.REACT_APP_PATH

Restart the local server by running npm start

<title>%REACT_APP_NAME%</title>

useRef (storing previous value)

- import { useState, useEffect, useRef } from "react";
  import ReactDOM from "react-dom/client";
  function App() {
    const [name, setName] = useState("");
    const prevName = useRef(0);
    useEffect(() => {
      prevName.current = name;
    });
- return (
    <>
      <input
        type="text“
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <h1>Current Name: {name}</h1>
      <h1>Previous Name: {prevName.current}</h1>
    </>
  );

useRef (instead id / no rerender)

- export default function App25() {
  const divRef = useRef(null);
  const handleColorChange = () => {
  divRef.current.style.backgroundColor = 'silver’;
  };
  return (

  Hello! Welcom to Mars
  <button onClick={handleColorChange}>Click</button>

  );
  };
  ……
  inputRef.current.value = 10 //where inputRef is the ref of a textbox
  headerRef.current.textContent = text; //where headerRef is a div or p

Fetch API (.then example)

- export default function App27() {
  const [data, setdata] = useState([]);
  const url = "https://jsonplaceholder.typicode.com/users";
  useEffect(() => {
    fetch(url)
      .then((response) => response.json())
      .then((result) => setdata(result))
      .catch((err) => console.log(err));
  }, []);
  return (

      {data && data.map((elem) => {elem.name})}

  );
}

Fetch API (async/await example)

- export default function App28() {
  const [data, setdata] = useState([]);
  const url = "https://jsonplaceholder.typicode.com/users";
  const fetchData = async (url) => {
  try {
  const response = await fetch(url);
  const result = await response.json();
  setdata(result);
  } catch (err) {
  console.log(err);
  }
  };
  useEffect(() => {
  fetchData(url);
  }, []);
  return (

  {data && data.map((elem) => {elem.name})}

  );
  }

customHook – useFetch - parent

- ```
  import { useFetch } from "./useFetch";
  export default function App29() {
    const url = "https://jsonplaceholder.typicode.com/users/";
    const [data] = useFetch(url);
    console.log(data.length)
    return (

        {data && data.map((elem) => (<li
  key={elem.id}>{elem.name}</li>))}

    );
  }
  ```
  ```

useFetch.js – fetch.then

```
export const useFetch = (url) => {
  const [data, setData] = useState([]);
  useEffect(() => {
    fetch(url)
      .then((res) => res.json())
      .then((data) => setData(data));
  }, [url]);
  return [data];
};
```

Facebook-react

- project to use fetch function
- Accept userid thru a form