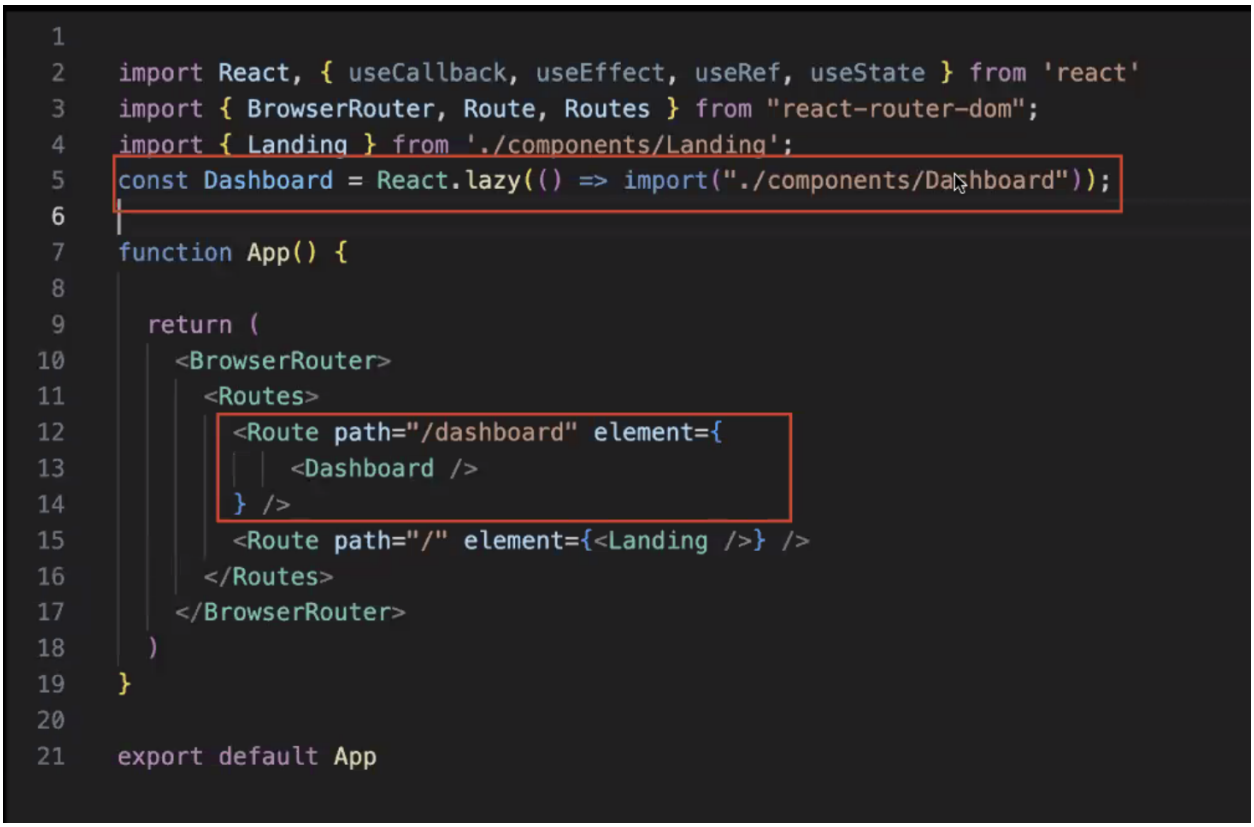
Day 20 - Routing, Prop Drilling, Context API

1. Lazy Loading :
   Which states that you can change your code a little bit , which will allow users visiting page 1 to only see page 1 but not page 2. This reduces the memory bandwidth.

```
React.lazy()
```

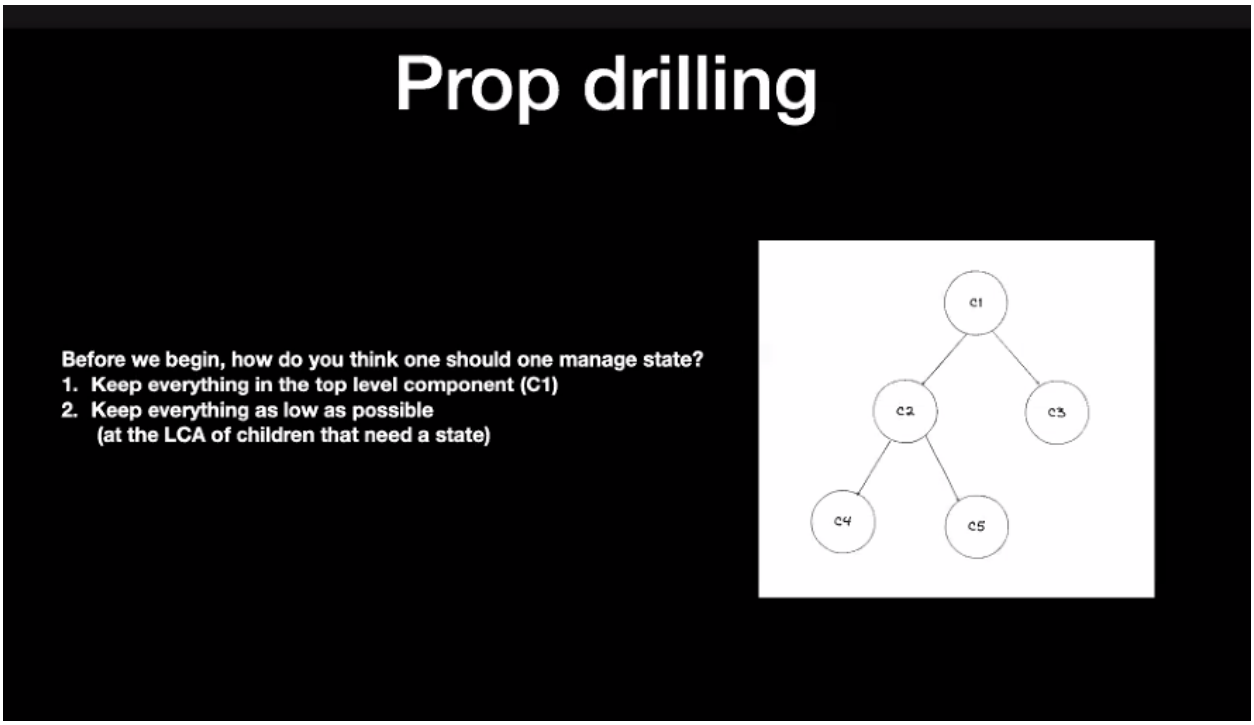


```
1
2   import React, { useCallback, useEffect, useRef, useState } from 'react'
3   import { BrowserRouter, Route, Routes } from "react-router-dom";
4   import { Landing } from './components/Landing';
5   const Dashboard = React.lazy(() => import("./components/Dashboard"));
6
7   function App() {
8
9     return (
10      <BrowserRouter>
11        <Routes>
12          <Route path="/dashboard" element={
13              <Dashboard />
14          } />
15          <Route path="/" element={<Landing />} />
16        </Routes>
17      </BrowserRouter>
18    )
19  }
20
21  export default App
```

Only for large applications, that too wrap it inside a suspense.

2. Prop Drilling

~ Anti Pattern in react.



3. prop drilling code:
   Anti pattern : Taking from one and passing it down.
   Makes the code visually un-appealing.

```
//prop drilling -- drilling down the props.

import { useState } from "react";

function App() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <Count count={count} setCount={setCount} />
    </div>
  );
}
```

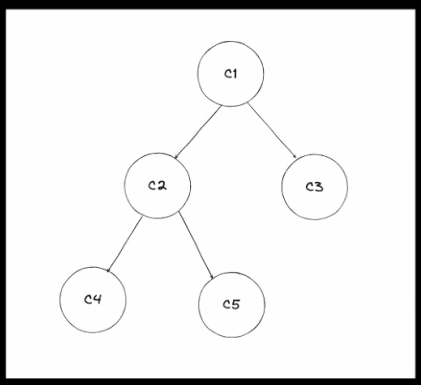
```jsx
function Count({ count }) {
  return (
    <div>
      {count}
      <Buttons count={count} setCount={setCount}></Buttons>
    </div>
  );
}

function Buttons() {
  return (
    <div>
      <button onClick={() => {}}>Increase</button>

      <button onClick={() => {}}>Decrease</button>
    </div>
  );
}

export default App;
```

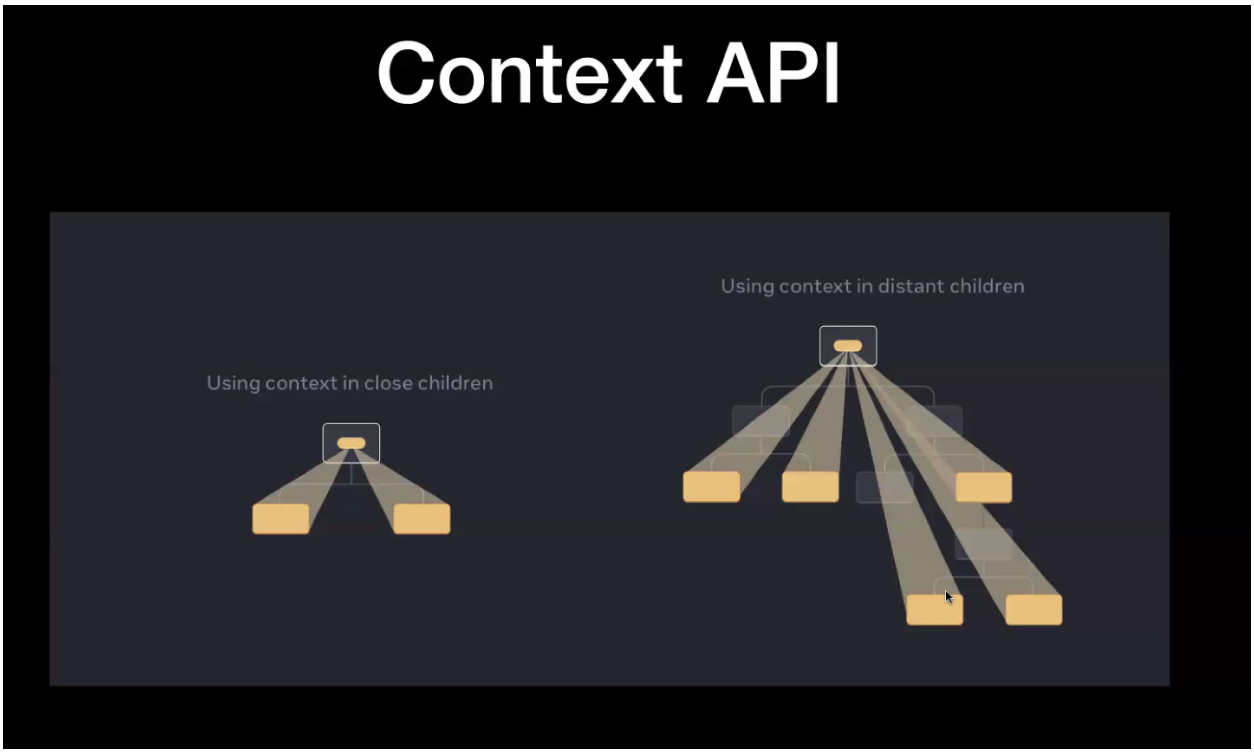
Prop drilling doesn't mean that parent re-renders children
It just means the syntactic uneasiness when writing code



? —> *how ugly it looks, nothing with rerendering*

4. Context API : makes you fix props driling

> 💡 Context API **allows data to be passed through a component tree without having to pass props manually at every level**. This makes it easier to share data between components. A diagram illustrating how Context API works.



Teleport children in different directions to get rid of prop drilling!
Let’s you keep all logic outside.

Context API vs Recoil

> 💡 The Context API is simple and built into React, but it might not scale well for large applications. Recoil offers a more granular and React-like approach to state management, which can lead to better performance in large applications, but it's still relatively new and may lack some ecosystem support

5. After this Redux, React etc were introduced for mass production state management.
6. JARGONS : Reducer, Use Reducer Hooks