6.1
React Deeper dive

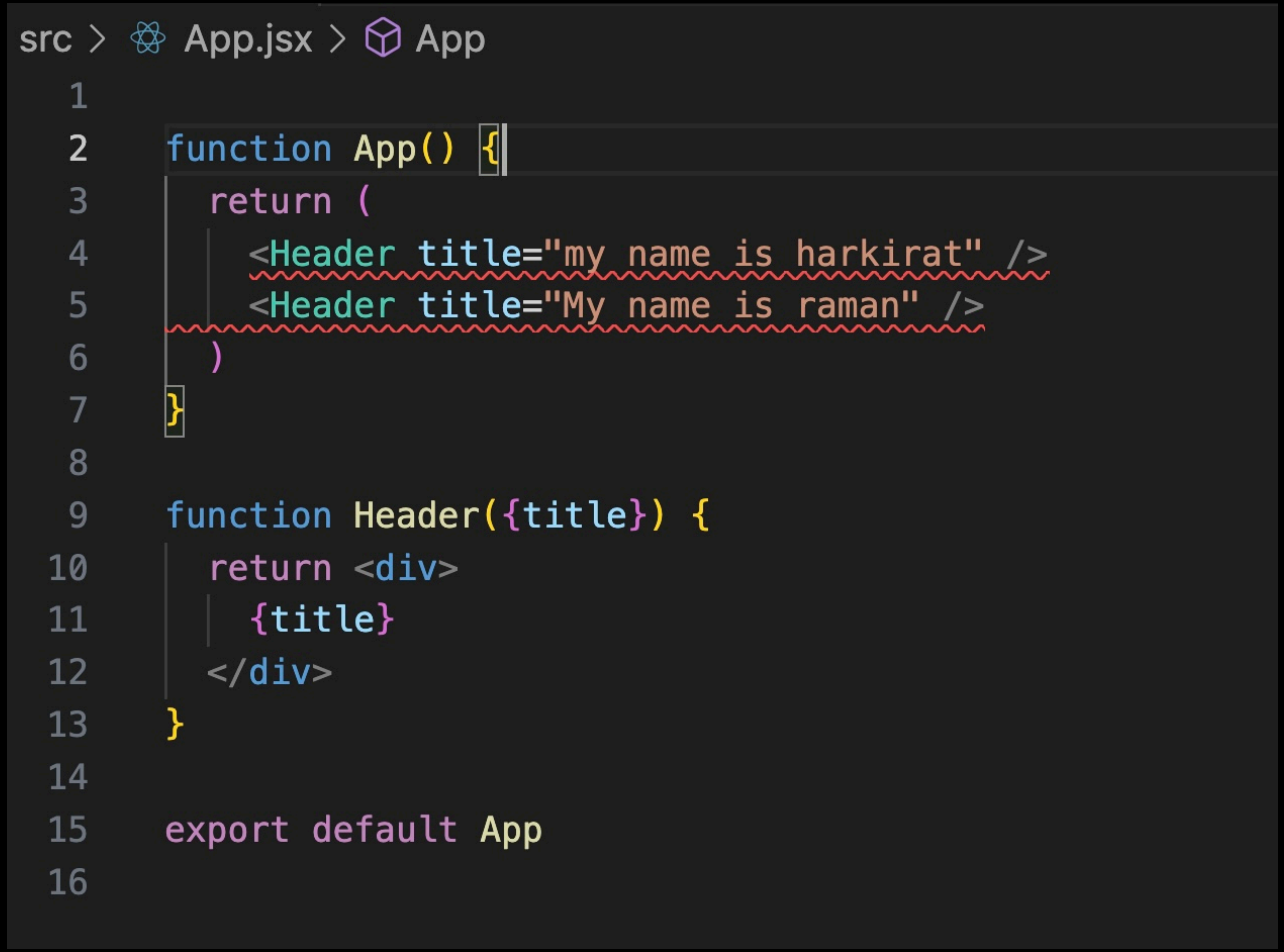
React returns, re-rendering, key, Wrapper components,
useEffect, useMemo, useCallback, useRef,
Prop Drilling

React component return

A component can only return a single top level xml

Why?

1. Makes it easy to do reconciliation
2.



src > App.jsx > App

```jsx

function App() {
  return (
    <Header title="my name is harkirat" />
    <Header title="My name is raman" />
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

React component return

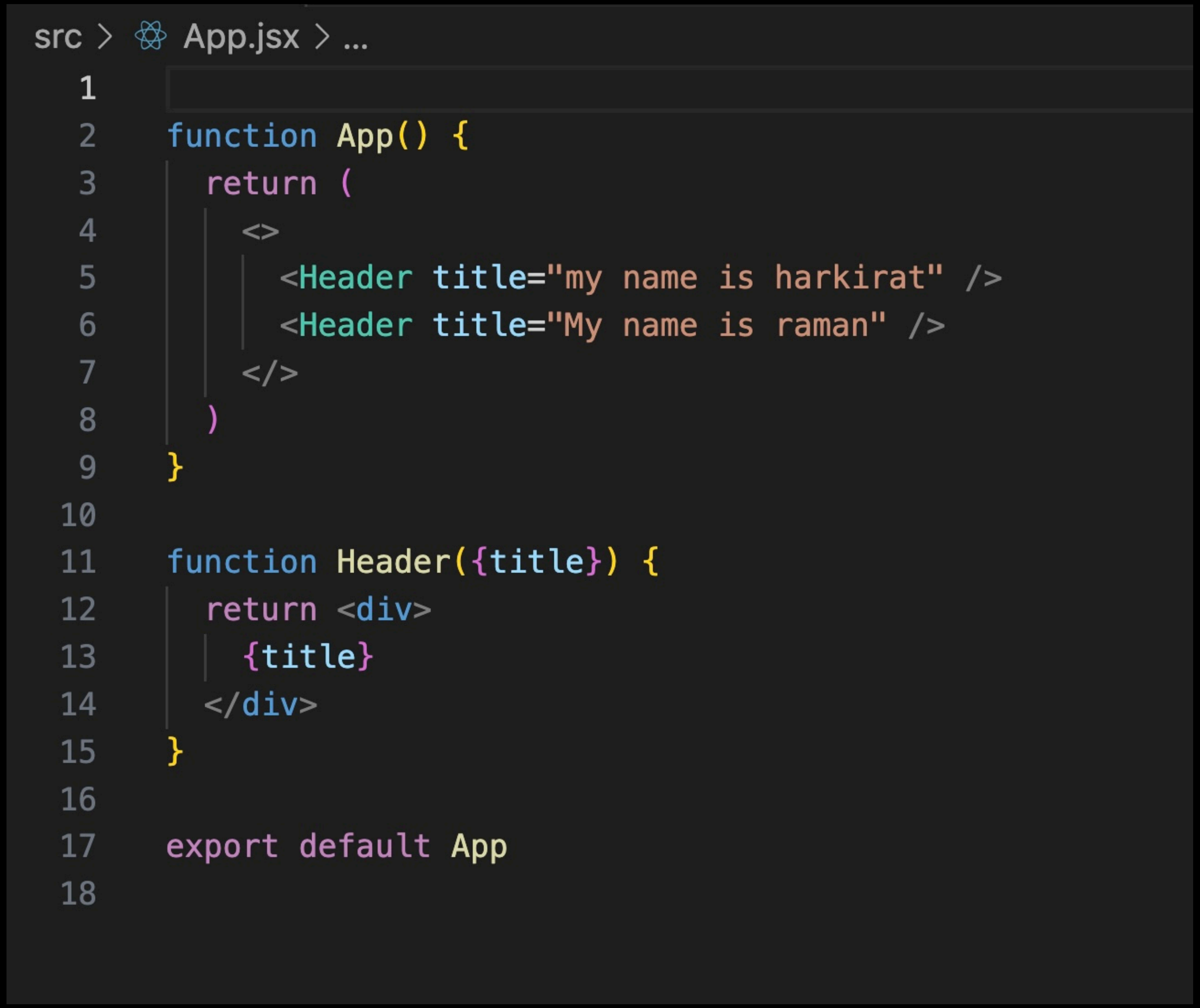
Create a react app that has a

1. Header component that takes a title as a prop and renders it inside a div
2. The top level App component renders 2 Headers

Solution
https://gist.github.com/hkirat/7ebe74aa564d01b331d7dfd4b627addc

(If you’re using this, please delete everything from index.css and App.css locally)

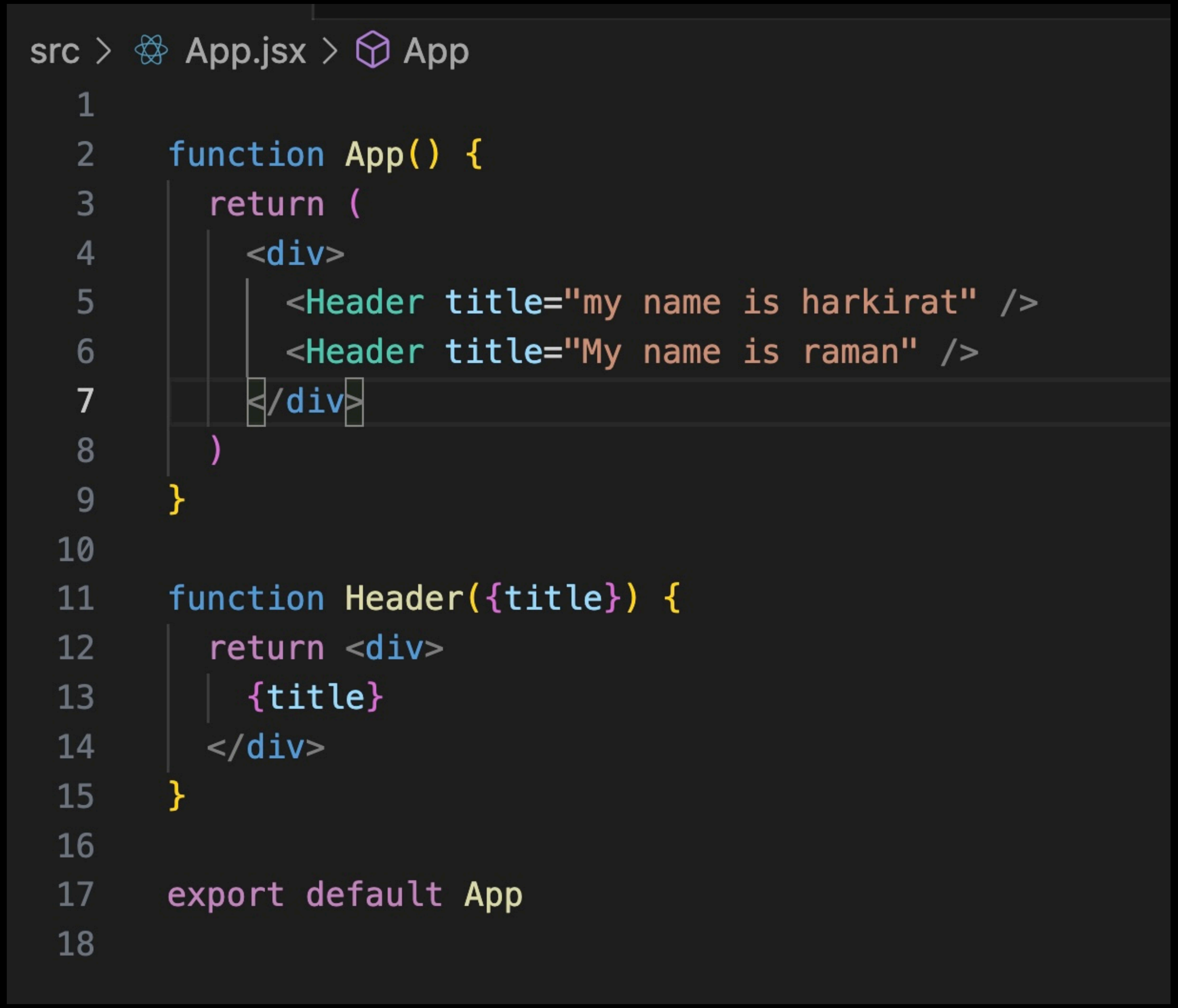
React component return



src > App.jsx > ...

```jsx

function App() {
  return (
    <>
      <Header title="my name is harkirat" />
      <Header title="My name is raman" />
    </>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

✓



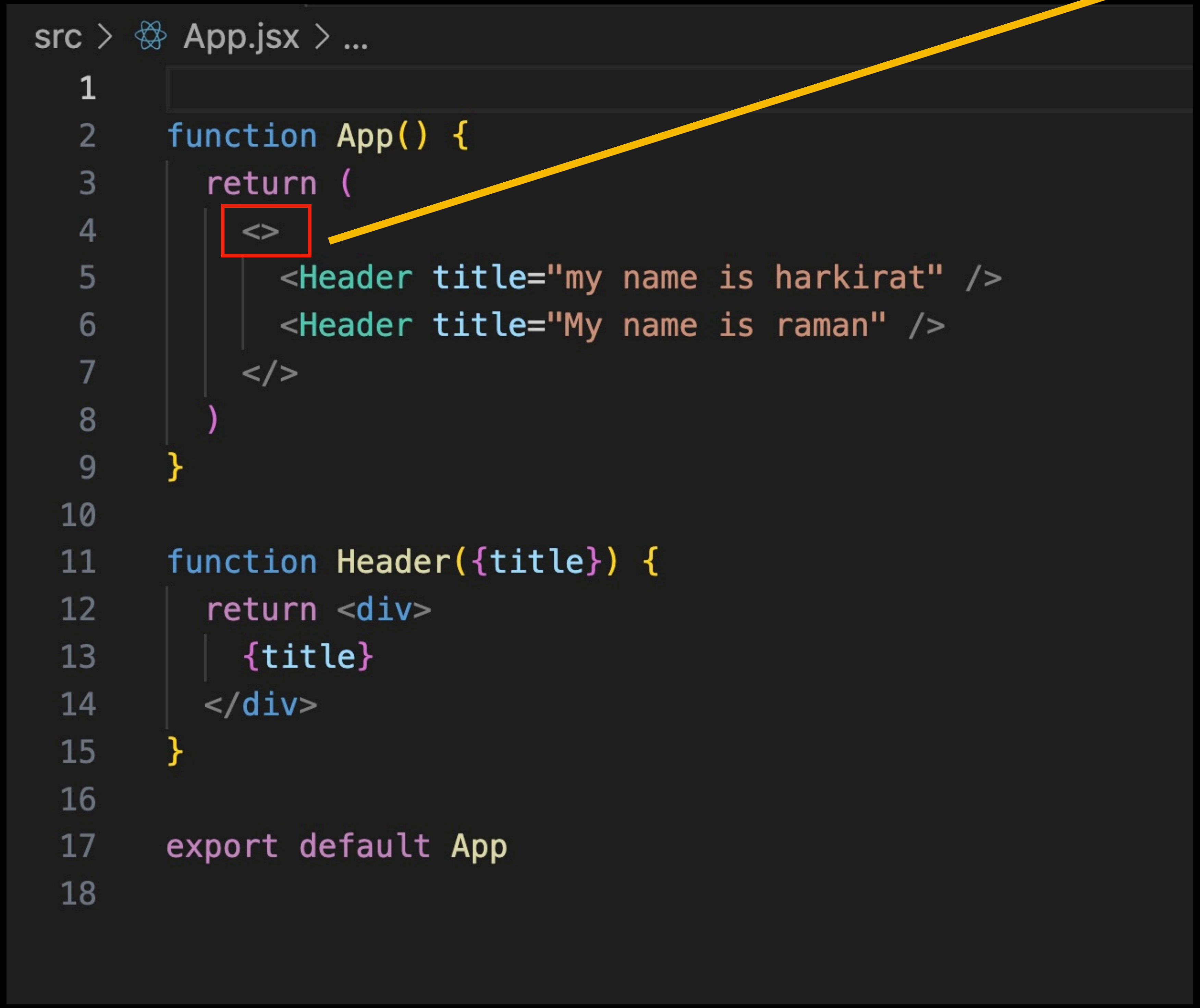
src > App.jsx > App

```jsx

function App() {
  return (
    <div>
      <Header title="my name is harkirat" />
      <Header title="My name is raman" />
    </div>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

✓

React component return

Doesn’t introduce an extra DOM element

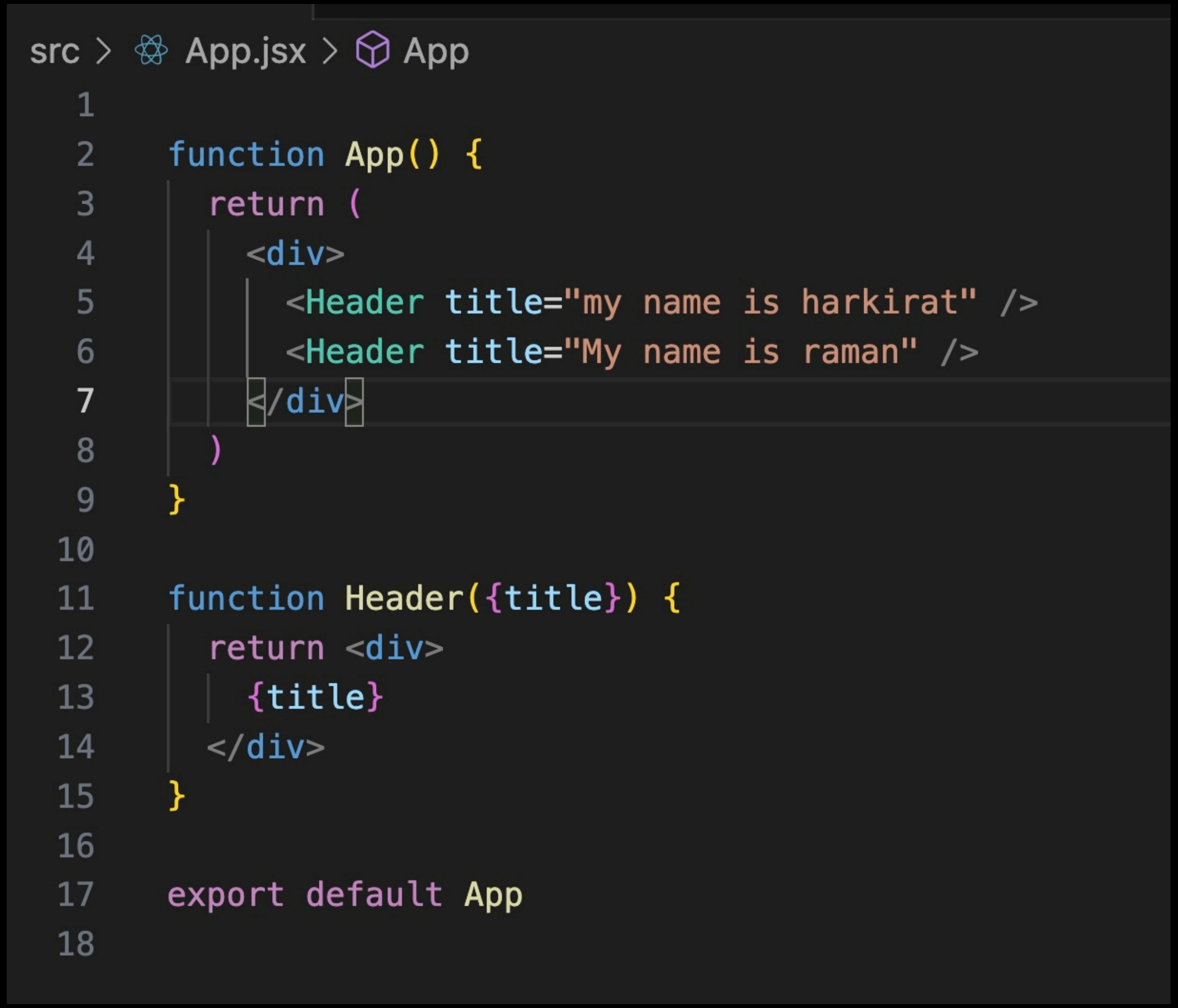


src > App.jsx > ...

```jsx

function App() {
  return (
    <>
      <Header title="my name is harkirat" />
      <Header title="My name is raman" />
    </>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

Slightly better

✓



src > App.jsx > App

```jsx

function App() {
  return (
    <div>
      <Header title="my name is harkirat" />
      <Header title="My name is raman" />
    </div>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

✓

https://react.dev/reference/react/Fragment

Re-rendering in react

What exactly is a re-render?

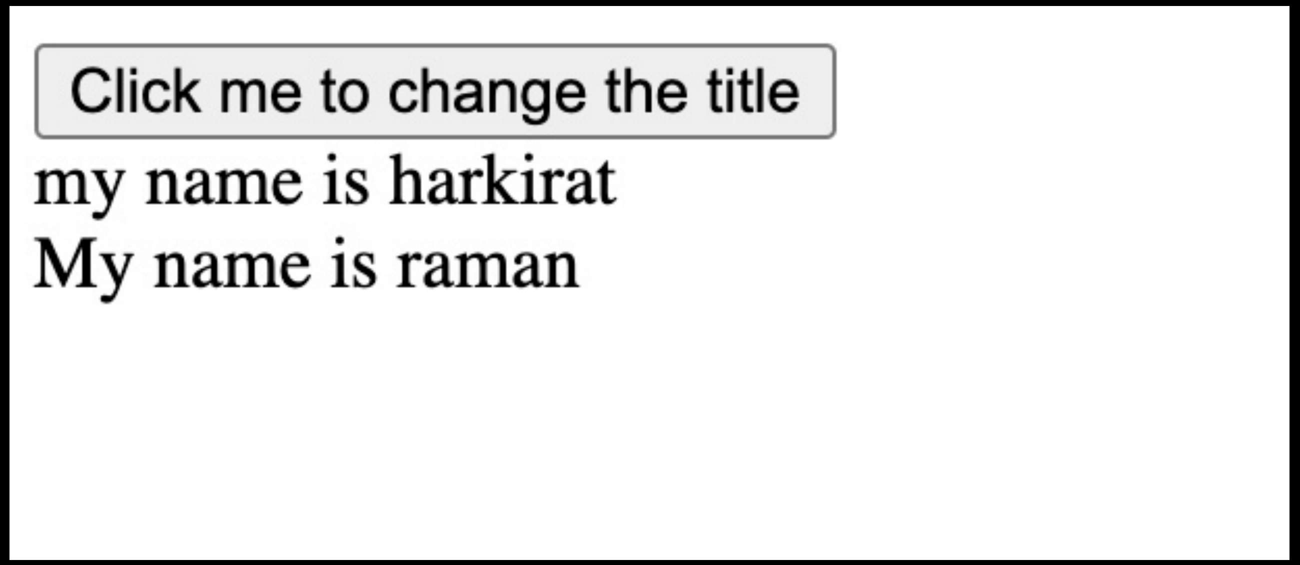
Please install react developer tools to visualise it

https://chromewebstore.google.com/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi

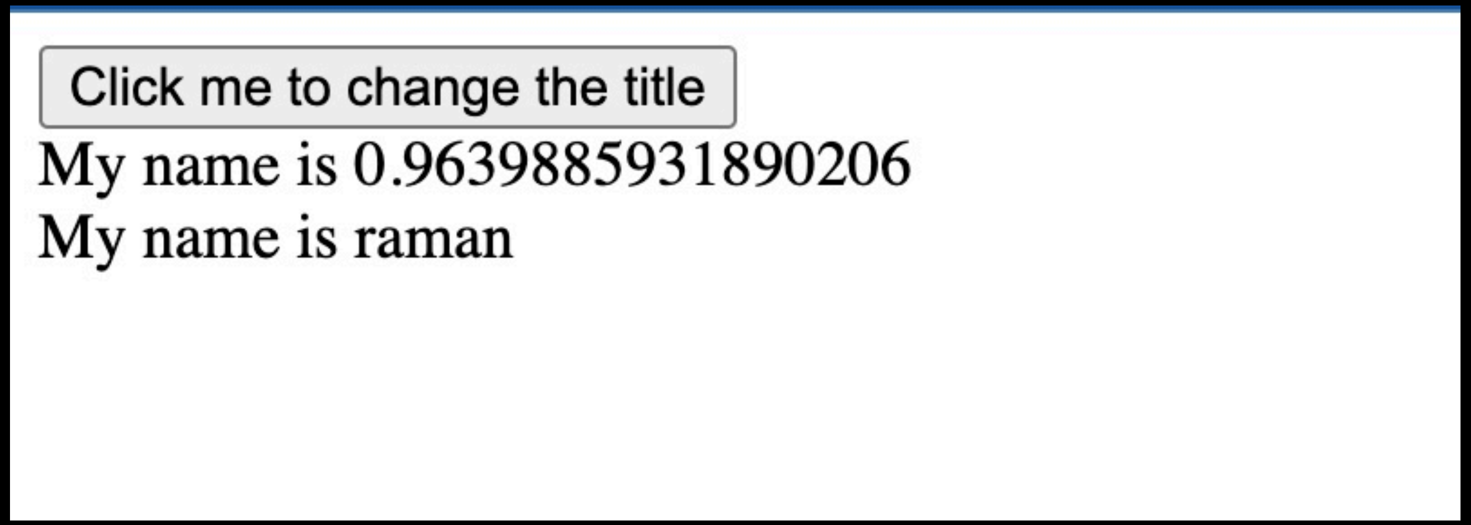
Re-rendering in react

What exactly is a re-render?

Update the last app to allow user to update the title of the first Header with a new title



Click me to change the title
my name is harkirat
My name is raman



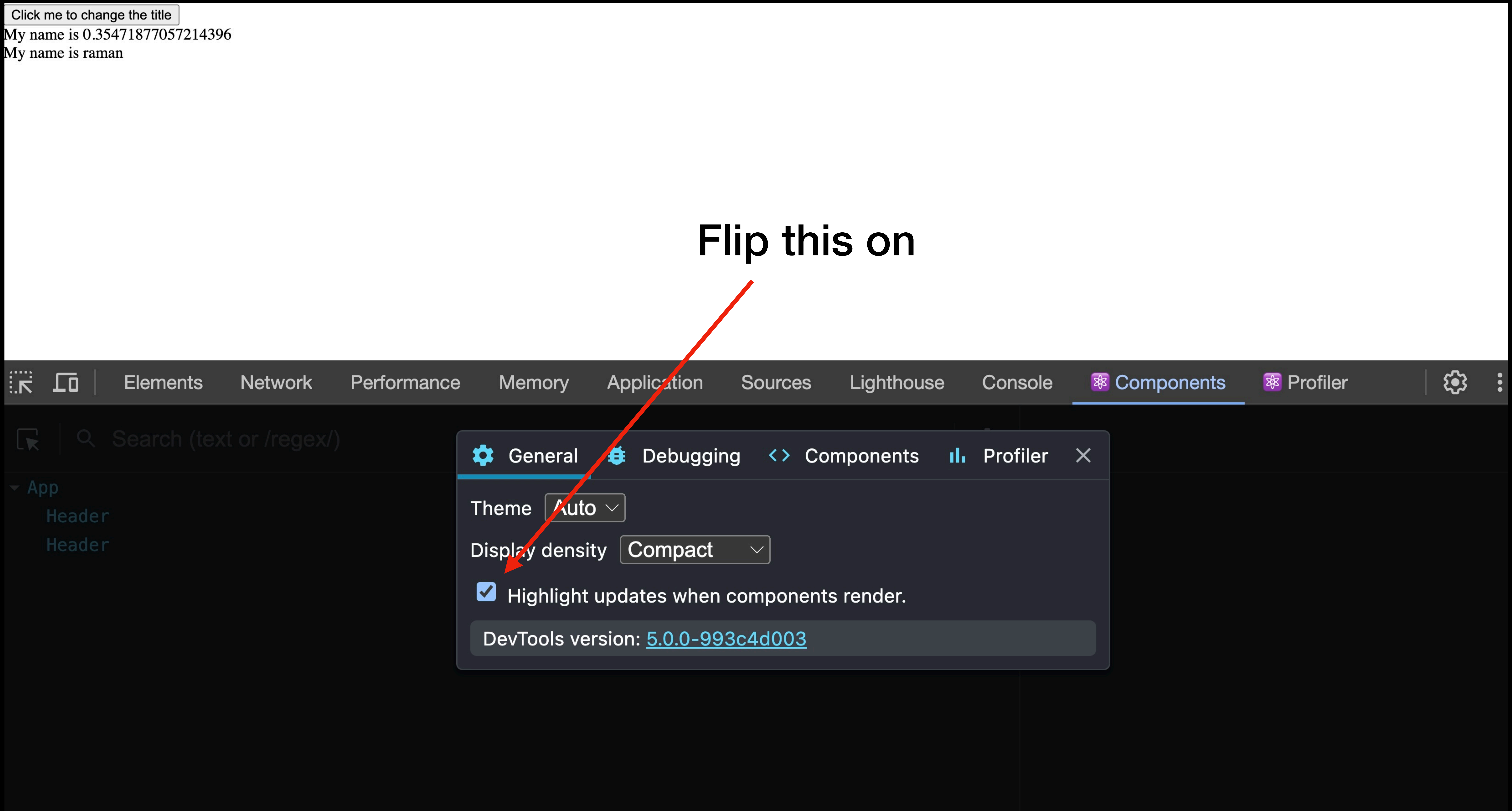
Click me to change the title
My name is 0.9639885931890206
My name is raman

Hint (Math.random() gives you a random number b/w 0-1)

https://gist.github.com/hkirat/290d17e7ca533898affbb6938ddcde56

Re-rendering in react

Update the last app to allow user to update the title of the first Header with a new title



Click me to change the title

My name is 0.35471877057214396

My name is raman

Flip this on

Theme: Auto

Display density: Compact

[✓] Highlight updates when components render.

DevTools version: 5.0.0-993c4d003

Re-rendering in react

A re-render means that

1. **React did some work to calculate what all should update in this component**
2. **The component actually got called (you can put a log to confirm this)**
3. **The inspector shows you a bounding box around the component**

It happens when

1. **A state variable that is being used inside a component changes**
2. **A parent component re-render triggers all children re-rendering**

Re-rendering in react

A re-render means that

1. React did some work to calculate what all should update in this component
2. The component actually got called (you can put a log to confirm this)
3. The inspector shows you a bounding box around the component

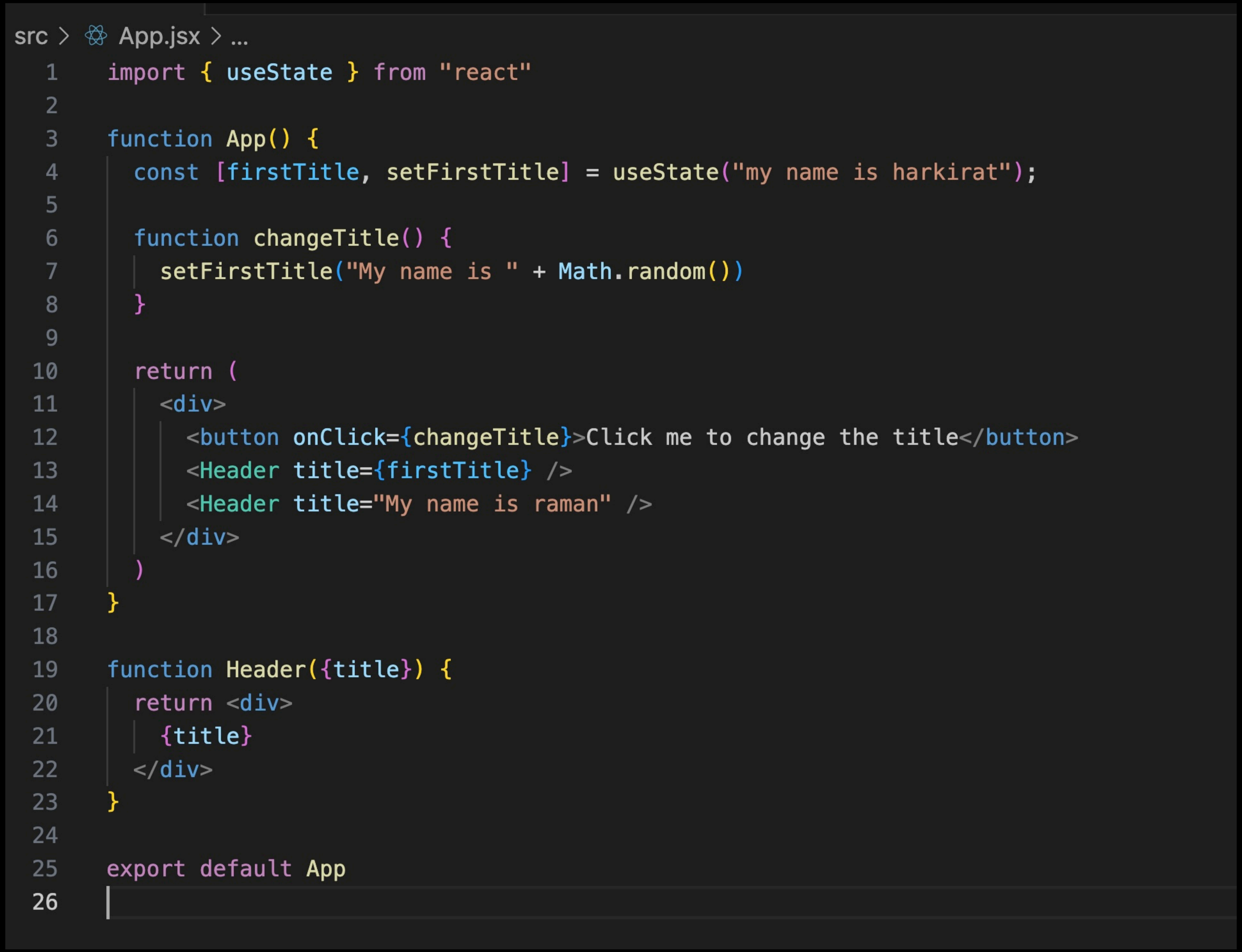
It happens when

1. A state variable that is being used inside a component changes
2. A parent component re-render triggers all children re-rendering

You want to minimise the number of re-renders to make a highly optimal react app
The more the components that are getting re-rendered, the worse

Re-rendering in react

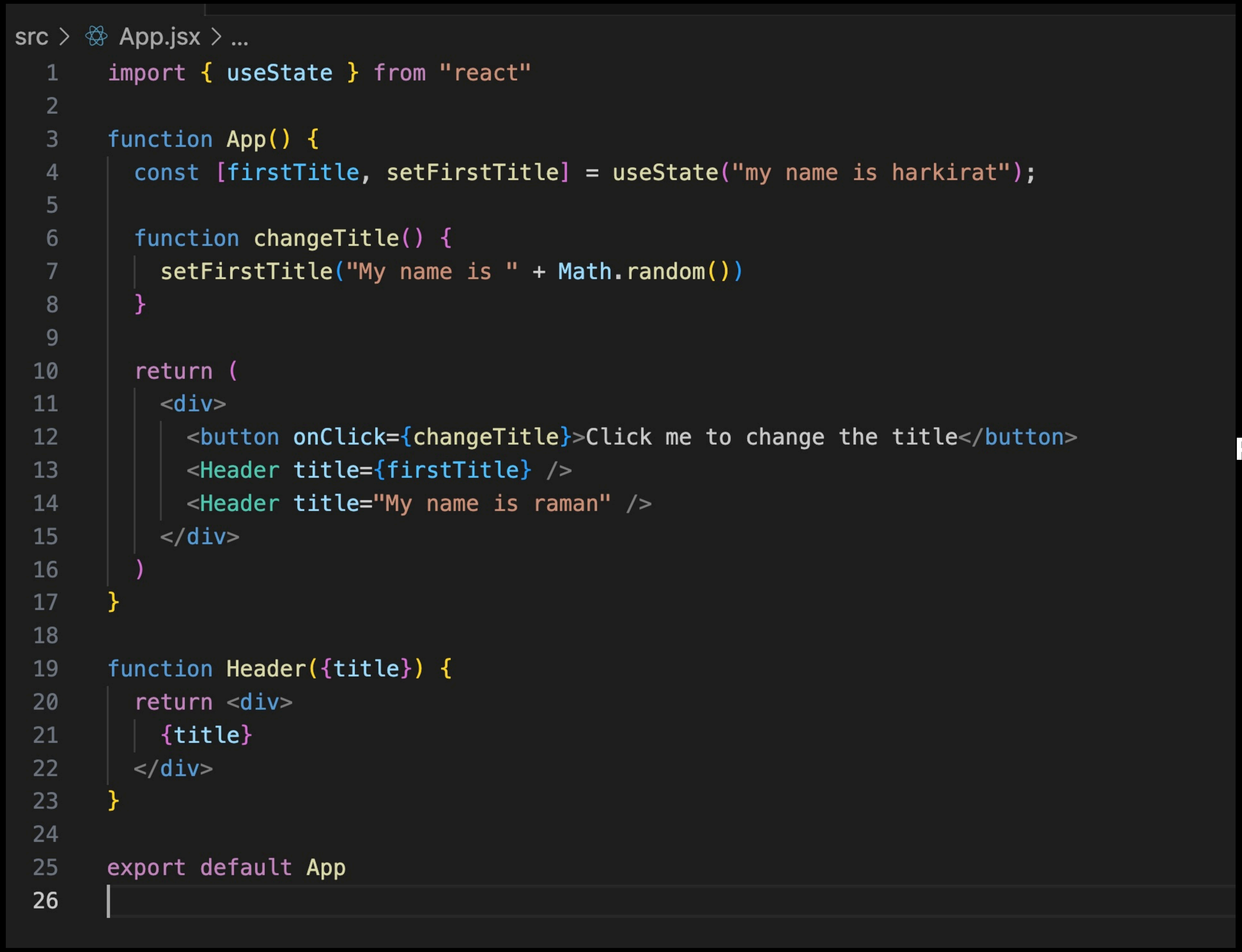
How can you minimise the number of re-renders in this app?


src > App.jsx > ...

```jsx
import { useState } from "react"

function App() {
  const [firstTitle, setFirstTitle] = useState("my name is harkirat");

  function changeTitle() {
    setFirstTitle("My name is " + Math.random())
  }

  return (
    <div>
      <button onClick={changeTitle}>Click me to change the title</button>
      <Header title={firstTitle} />
      <Header title="My name is raman" />
    </div>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

Re-rendering in react

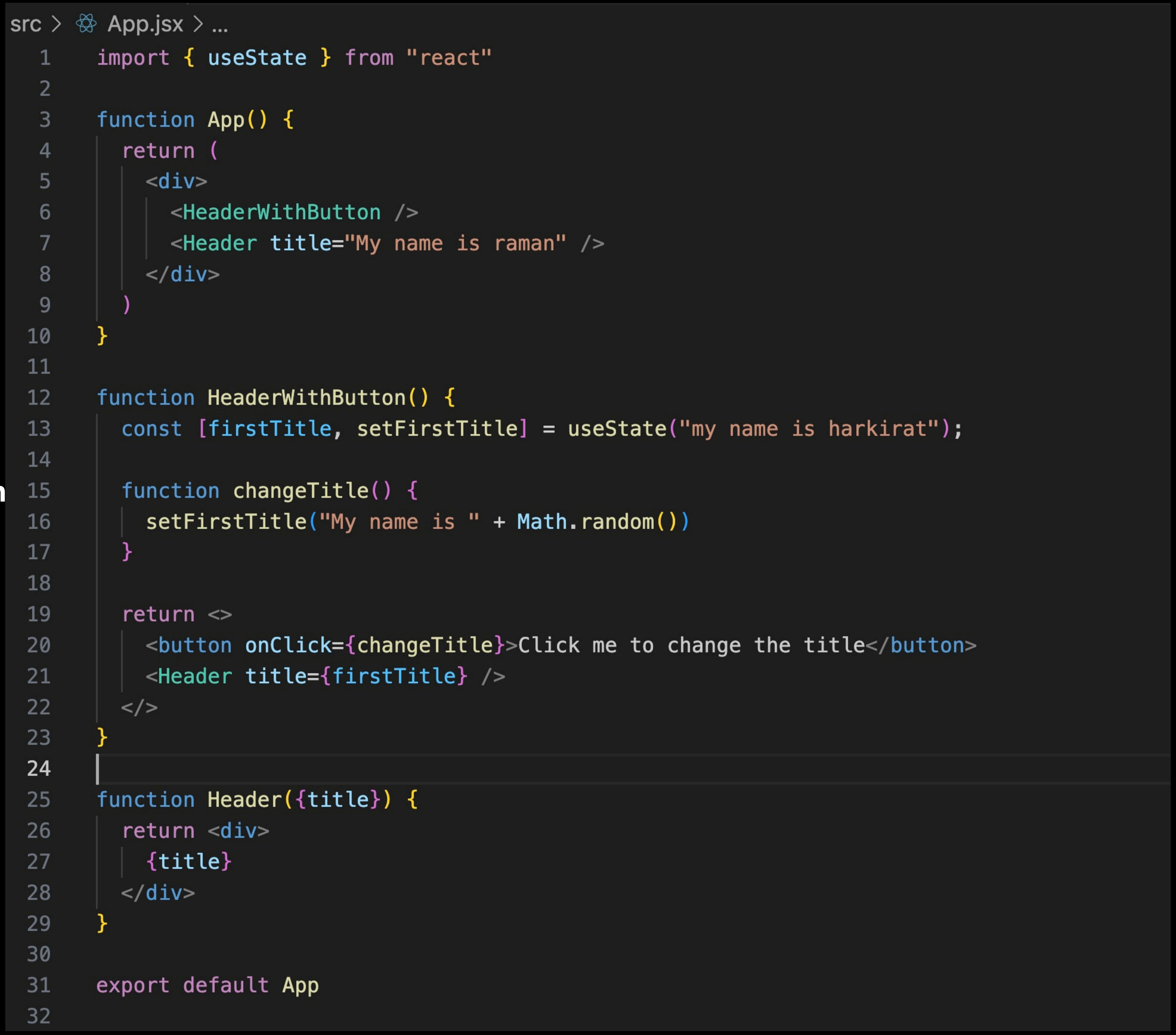
How can you minimise the number of re-renders in this app?



src > App.jsx > ...

```jsx
import { useState } from "react"

function App() {
  const [firstTitle, setFirstTitle] = useState("my name is harkirat");

  function changeTitle() {
    setFirstTitle("My name is " + Math.random())
  }

  return (
    <div>
      <button onClick={changeTitle}>Click me to change the title</button>
      <Header title={firstTitle} />
      <Header title="My name is raman" />
    </div>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

Pushing the state down



src > App.jsx > ...

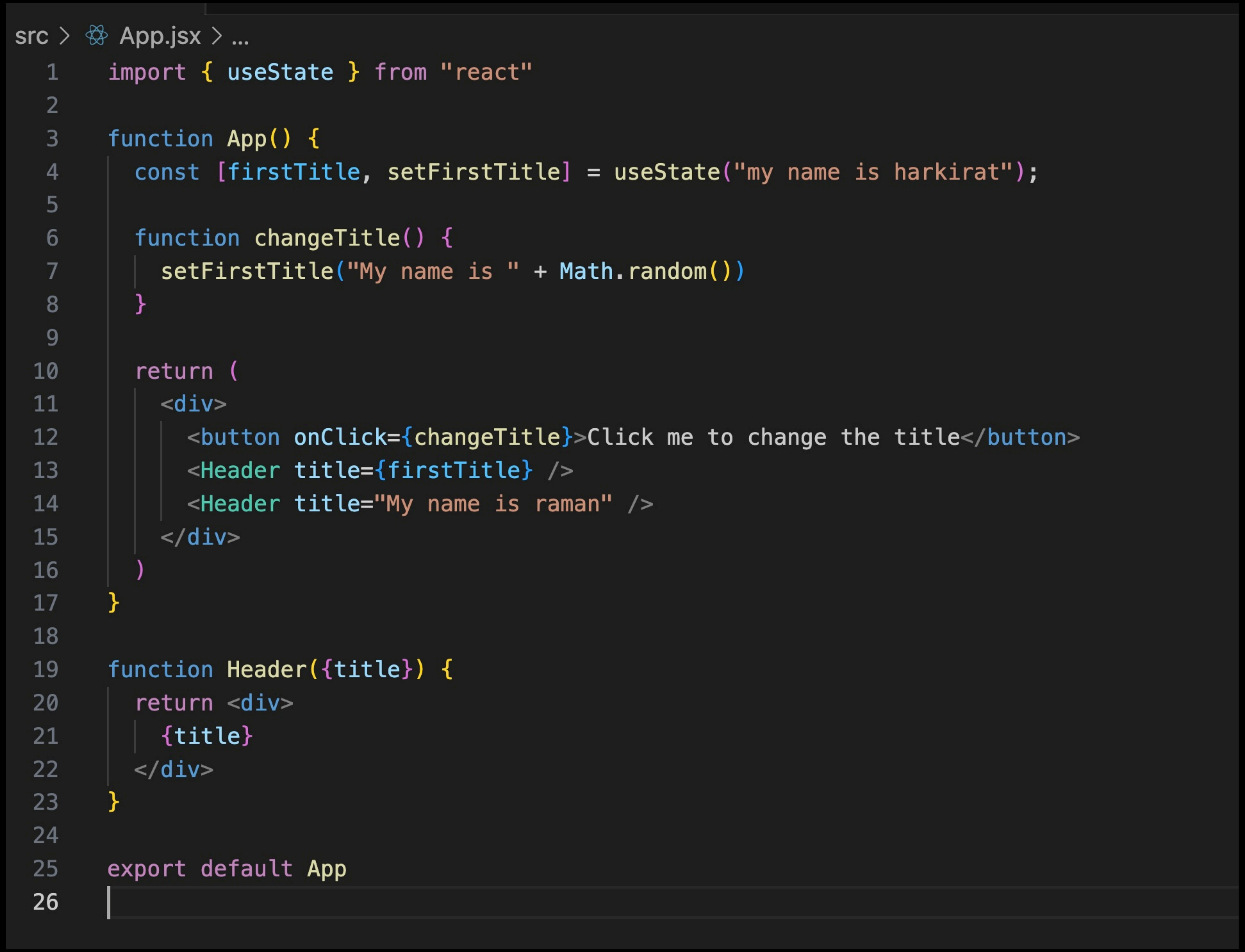
```jsx
import { useState } from "react"

function App() {
  return (
    <div>
      <HeaderWithButton />
      <Header title="My name is raman" />
    </div>
  )
}

function HeaderWithButton() {
  const [firstTitle, setFirstTitle] = useState("my name is harkirat");

  function changeTitle() {
    setFirstTitle("My name is " + Math.random())
  }

  return <>
    <button onClick={changeTitle}>Click me to change the title</button>
    <Header title={firstTitle} />
  </>
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

https://gist.github.com/hkirat/03449899d5497b1375790890c8a190b6

Re-rendering in react

How can you minimise the number of re-renders in this app?

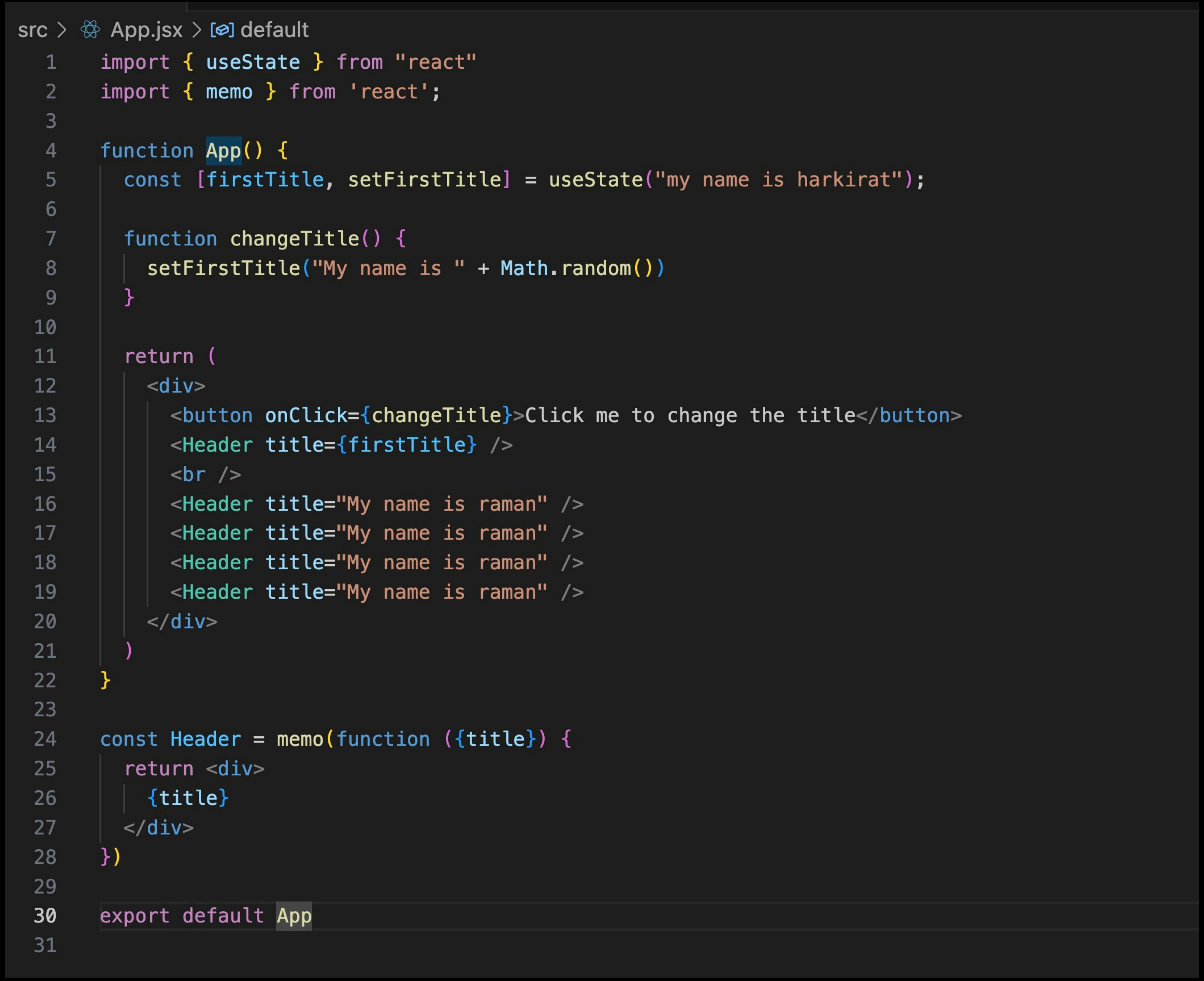
https://react.dev/reference/react/memo



src > App.jsx > ...

```jsx
import { useState } from "react"

function App() {
  const [firstTitle, setFirstTitle] = useState("my name is harkirat");

  function changeTitle() {
    setFirstTitle("My name is " + Math.random())
  }

  return (
    <div>
      <button onClick={changeTitle}>Click me to change the title</button>
      <Header title={firstTitle} />
      <Header title="My name is raman" />
    </div>
  )
}

function Header({title}) {
  return <div>
    {title}
  </div>
}

export default App

```

Use React.memo



src > App.jsx > default

```jsx
import { useState } from "react"
import { memo } from 'react';

function App() {
  const [firstTitle, setFirstTitle] = useState("my name is harkirat");

  function changeTitle() {
    setFirstTitle("My name is " + Math.random())
  }

  return (
    <div>
      <button onClick={changeTitle}>Click me to change the title</button>
      <Header title={firstTitle} />
      <br />
      <Header title="My name is raman" />
      <Header title="My name is raman" />
      <Header title="My name is raman" />
      <Header title="My name is raman" />
    </div>
  )
}

const Header = memo(function ({title}) {
  return <div>
    {title}
  </div>
})

export default App

```

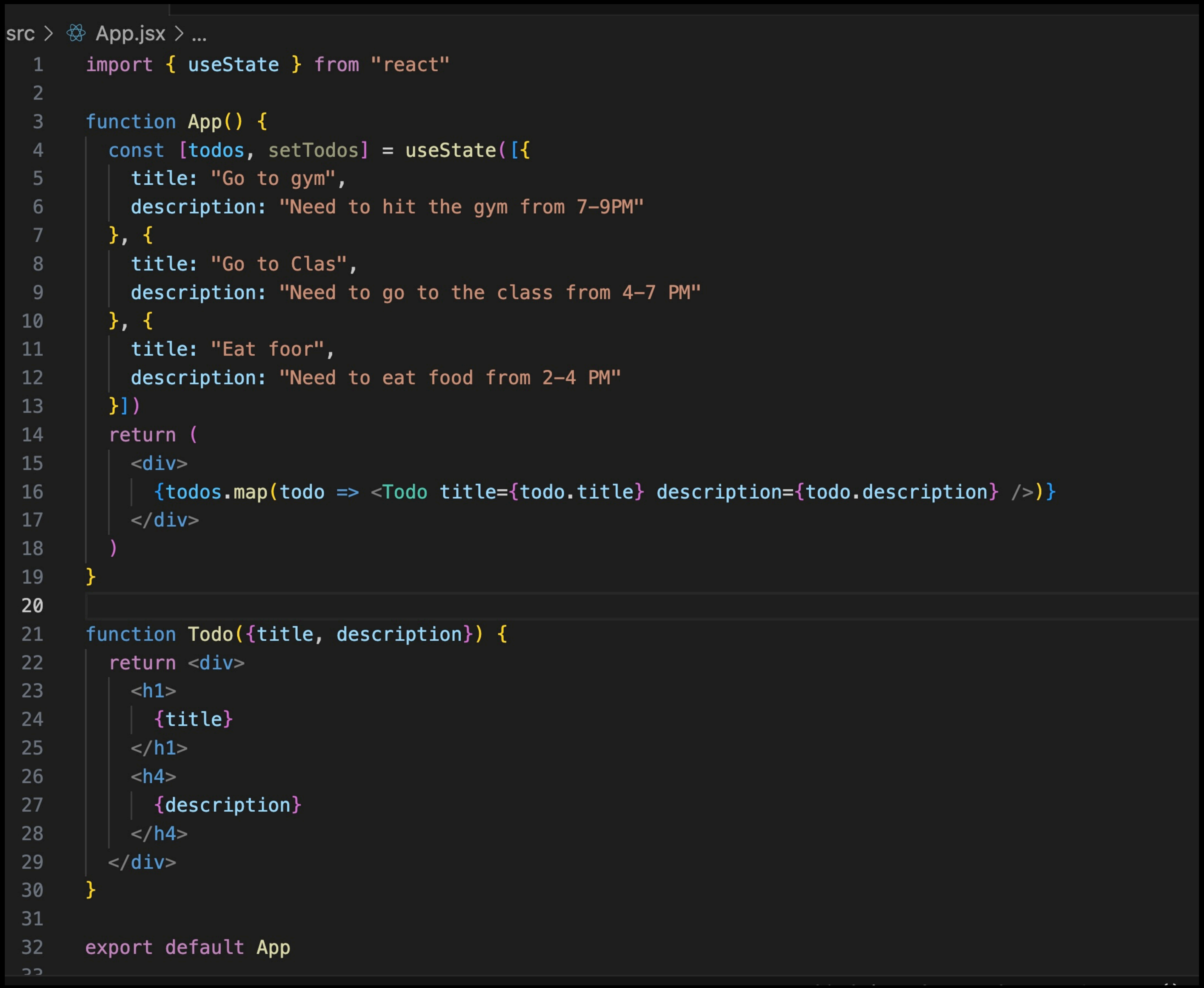
https://gist.github.com/hkirat/8934f0fd69686fd1e9e7e4af68899d2d

Keys in react

Lets create a simple todo app that renders 3 todos ✓

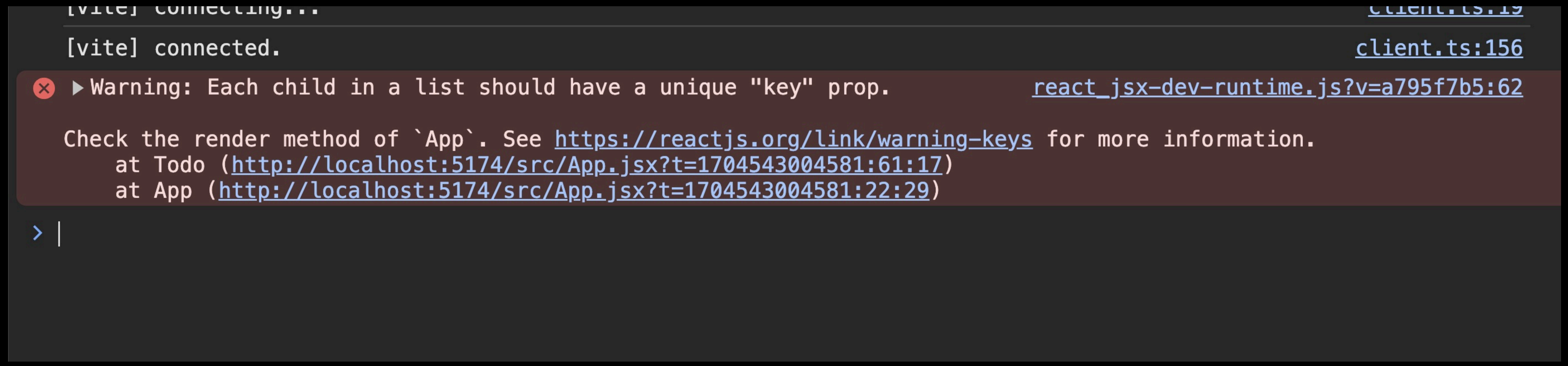
1. Create a Todo component that accepts title, description as input ✓
2. Initialise a state array that has 3 todos ✓
3. Iterate over the array to render all the TODOs ✓ .map()
4. A button in the top level App component to add a new TODO

Keys in react



src > App.jsx > ...

```jsx
import { useState } from "react"

function App() {
  const [todos, setTodos] = useState([{
    title: "Go to gym",
    description: "Need to hit the gym from 7-9PM"
  }, {
    title: "Go to Clas",
    description: "Need to go to the class from 4-7 PM"
  }, {
    title: "Eat foor",
    description: "Need to eat food from 2-4 PM"
  }])
  return (
    <div>
      {todos.map(todo => <Todo title={todo.title} description={todo.description} />)}
    </div>
  )
}

function Todo({title, description}) {
  return <div>
    <h1>
      {title}
    </h1>
    <h4>
      {description}
    </h4>
  </div>
}

export default App
```

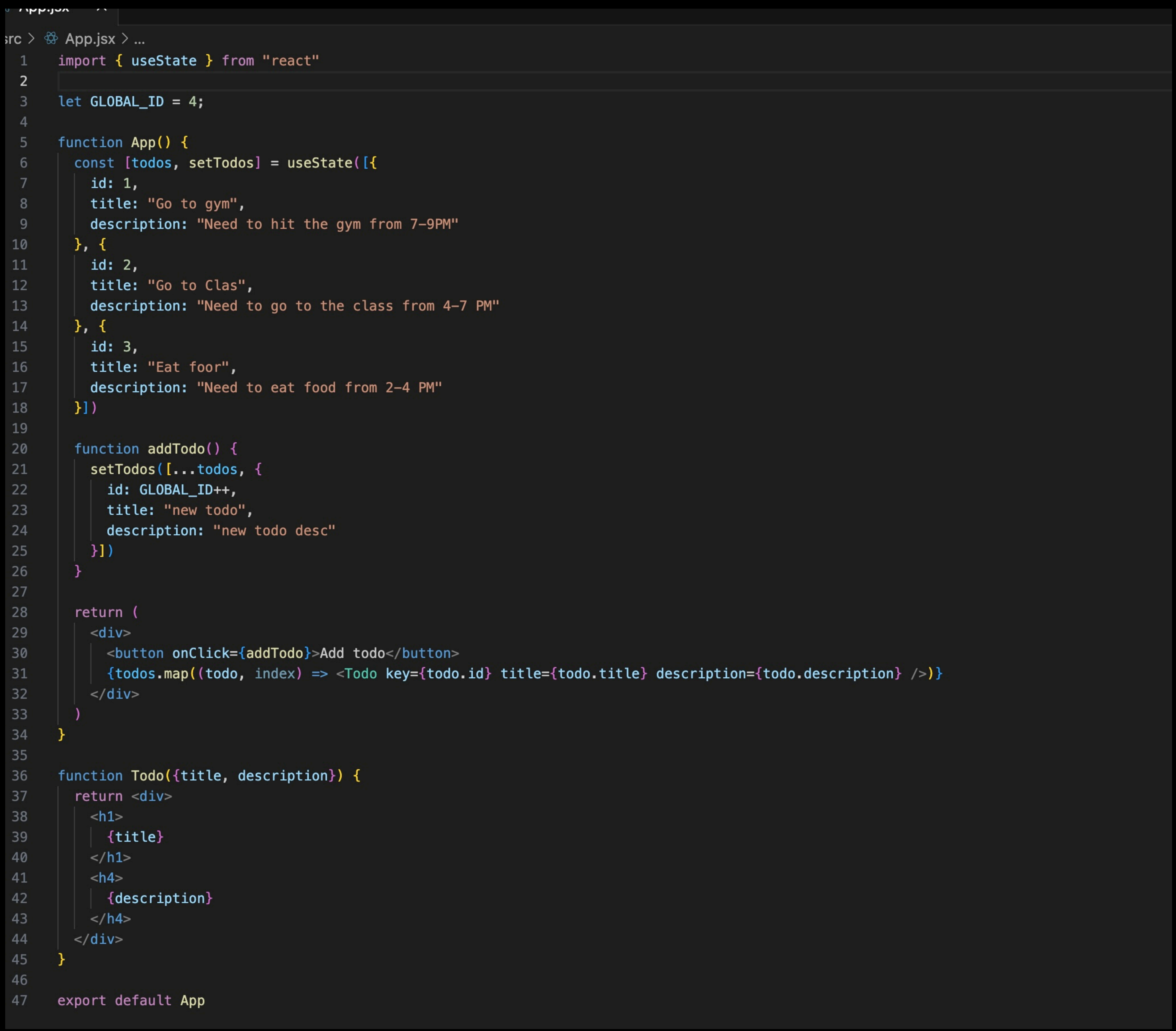


```
[vite] connecting...                                                          client.ts:19
[vite] connected.                                                             client.ts:156
▶Warning: Each child in a list should have a unique "key" prop.   react_jsx-dev-runtime.js?v=a795f7b5:62

Check the render method of `App`. See https://reactjs.org/link/warning-keys for more information.
    at Todo (http://localhost:5174/src/App.jsx?t=1704543004581:61:17)
    at App (http://localhost:5174/src/App.jsx?t=1704543004581:22:29)
>
```

https://gist.github.com/hkirat/693dfcd452c811bd35c171565e9e3d50

Keys in react

Keys let react figure out if a TODO has been update, which has been delete, which has been added

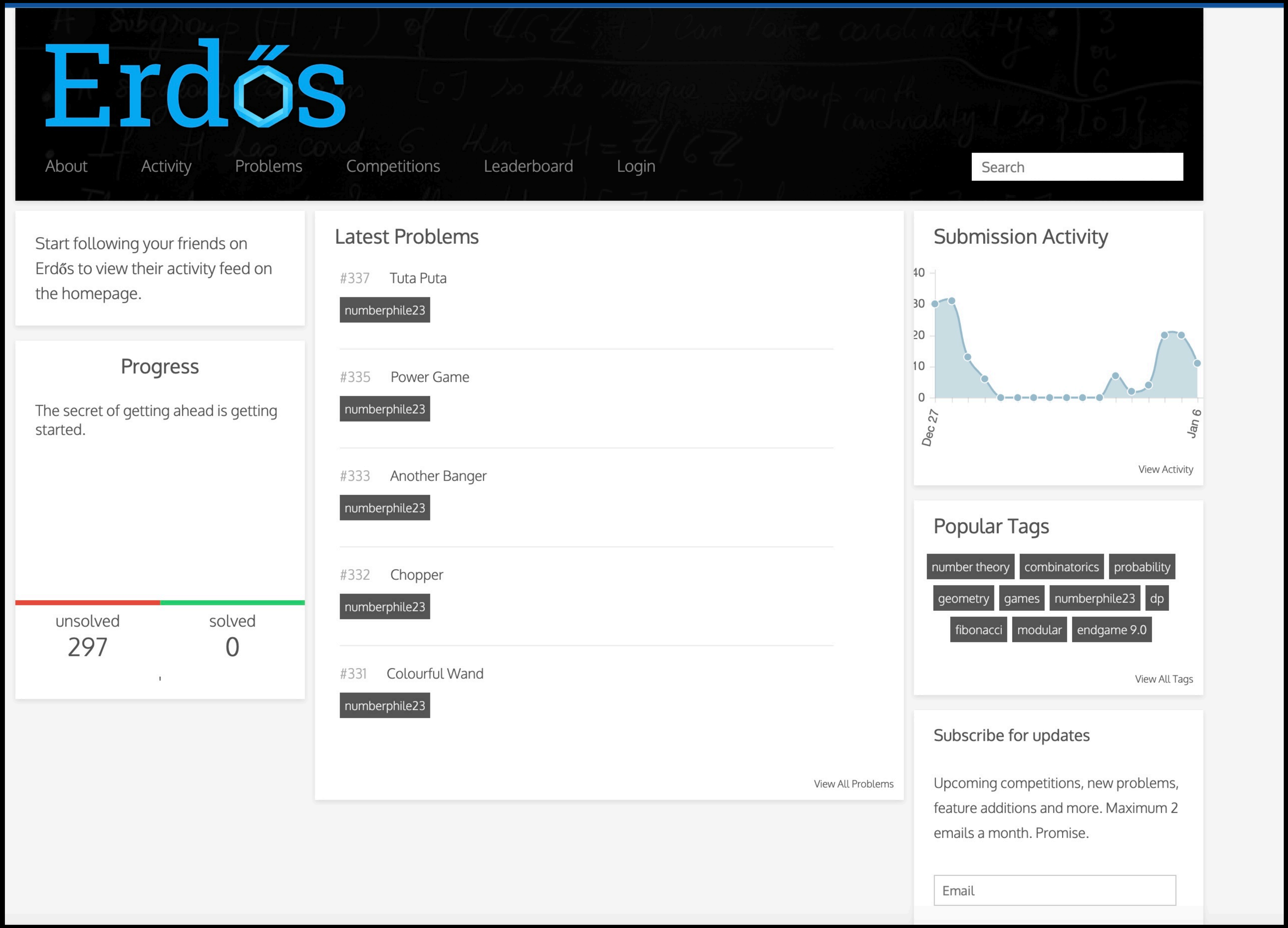


```
import { useState } from "react"

let GLOBAL_ID = 4;

function App() {
  const [todos, setTodos] = useState([{
    id: 1,
    title: "Go to gym",
    description: "Need to hit the gym from 7-9PM"
  }, {
    id: 2,
    title: "Go to Clas",
    description: "Need to go to the class from 4-7 PM"
  }, {
    id: 3,
    title: "Eat foor",
    description: "Need to eat food from 2-4 PM"
  }])

  function addTodo() {
    setTodos([...todos, {
      id: GLOBAL_ID++,
      title: "new todo",
      description: "new todo desc"
    }])
  }

  return (
    <div>
      <button onClick={addTodo}>Add todo</button>
      {todos.map((todo, index) => <Todo key={todo.id} title={todo.title} description={todo.description} />)}
    </div>
  )
}

function Todo({title, description}) {
  return <div>
    <h1>
      {title}
    </h1>
    <h4>
      {description}
    </h4>
  </div>
}

export default App
```

https://gist.github.com/hkirat/081e96d737029a1baff9cac2e3391d66

Wrapper components

**Lets say you want to build this,
You will notice a lot of cards on the
right look the same**



Erdős

About Activity Problems Competitions Leaderboard Login

Search

Start following your friends on Erdős to view their activity feed on the homepage.

Progress

The secret of getting ahead is getting started.

unsolved 297

solved 0

Latest Problems

#337 Tuta Puta
numberphile23

#335 Power Game
numberphile23

#333 Another Banger
numberphile23

#332 Chopper
numberphile23

#331 Colourful Wand
numberphile23

View All Problems

Submission Activity

View Activity

Popular Tags

number theory combinatorics probability geometry games numberphile23 dp fibonacci modular endgame 9.0

View All Tags

Subscribe for updates

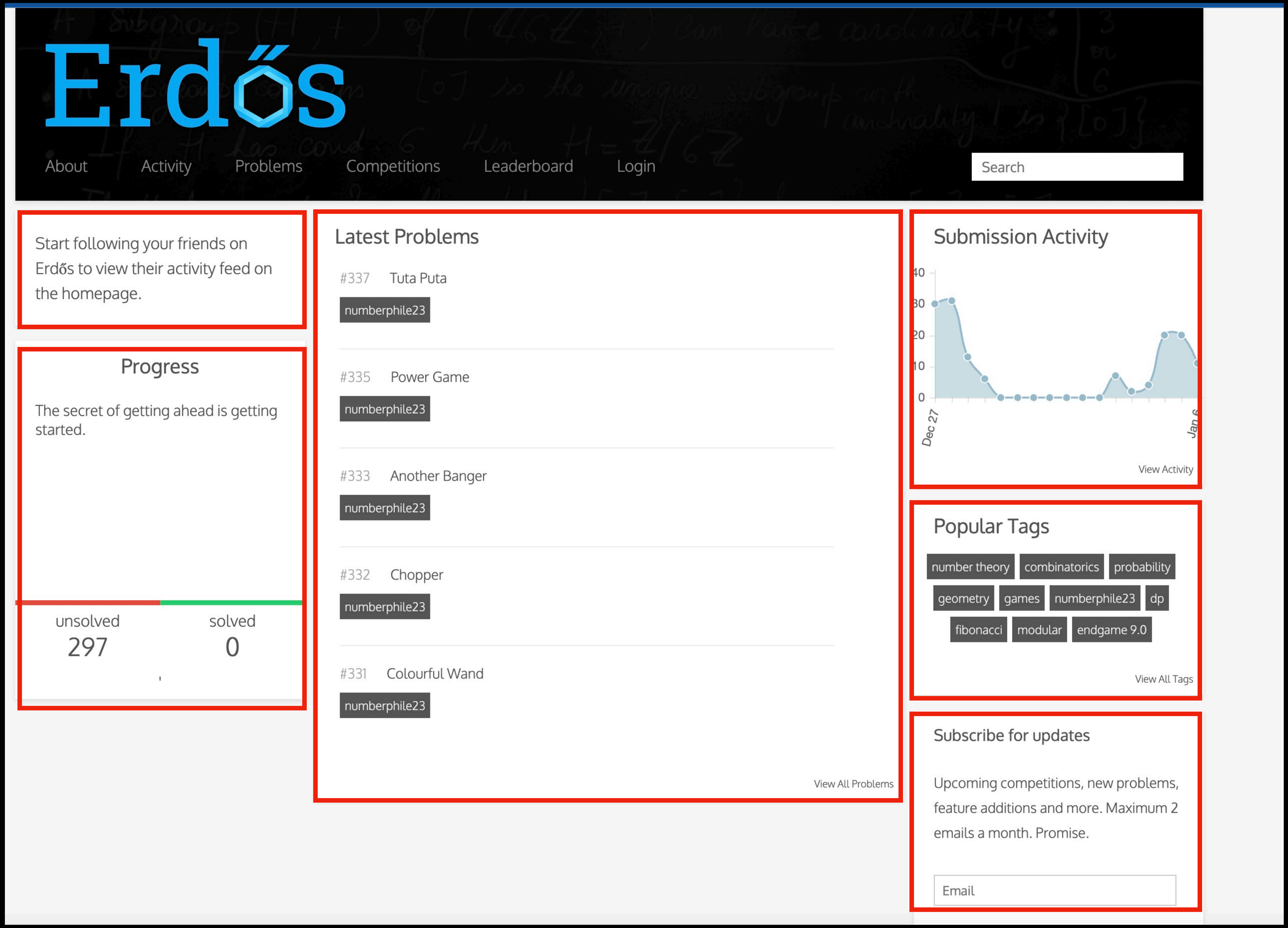
Upcoming competitions, new problems, feature additions and more. Maximum 2 emails a month. Promise.

Email

Wrapper components

Lets say you want to build this,
You will notice a lot of cards on the right look the same

You can create a wrapper Card component that takes the inner React component as an input



Erdős

About Activity Problems Competitions Leaderboard Login

Search

Start following your friends on Erdős to view their activity feed on the homepage.

Progress

The secret of getting ahead is getting started.

unsolved 297 solved 0

Latest Problems

#337 Tuta Puta
numberphile23

#335 Power Game
numberphile23

#333 Another Banger
numberphile23

#332 Chopper
numberphile23

#331 Colourful Wand
numberphile23

View All Problems

Submission Activity

View Activity

Popular Tags

number theory combinatorics probability geometry games numberphile23 dp fibonacci modular endgame 9.0

View All Tags

Subscribe for updates

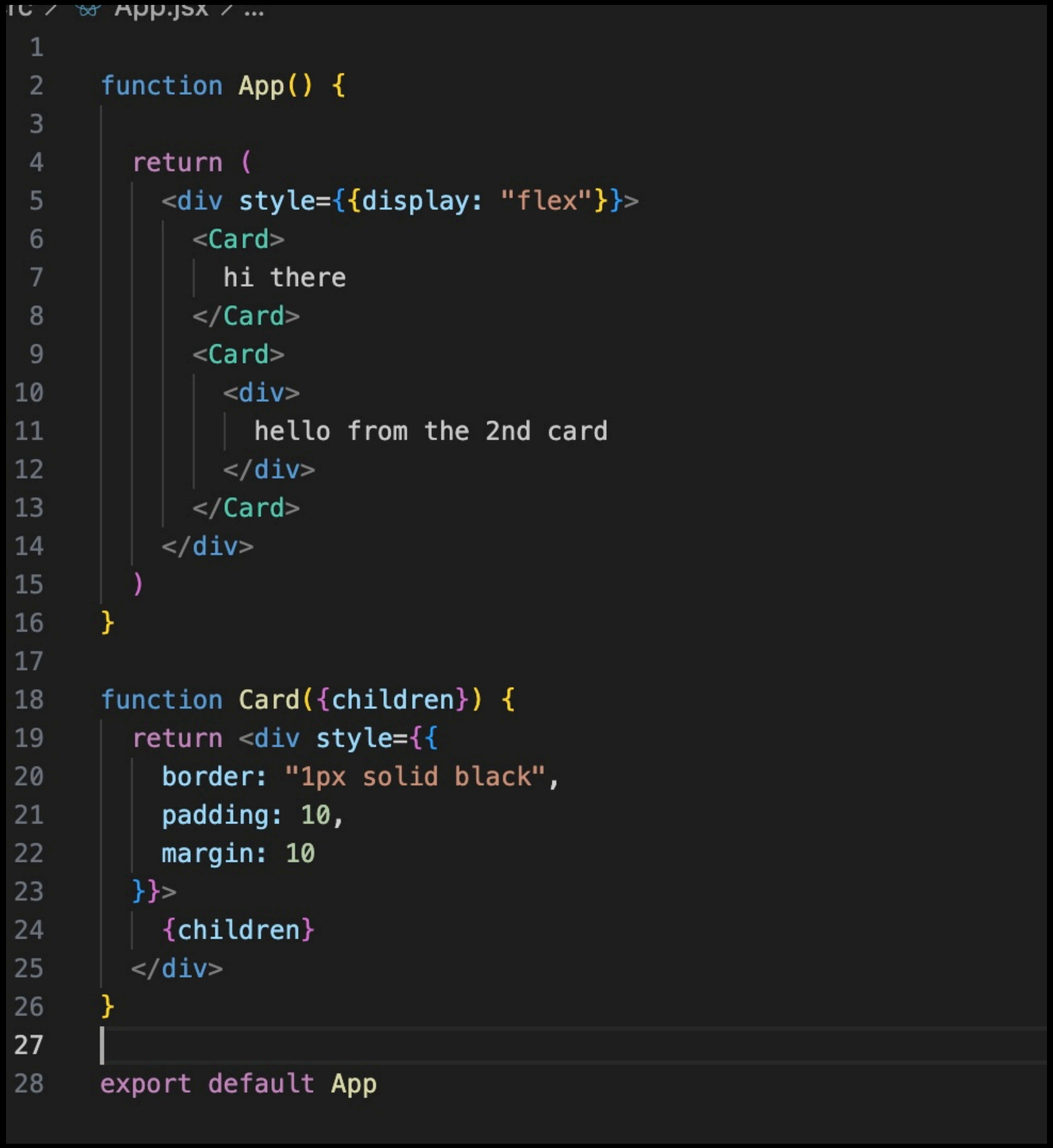
Upcoming competitions, new problems, feature additions and more. Maximum 2 emails a month. Promise.

Email

Wrapper components

Lets say you want to build this,
You will notice a lot of cards on the
right look the same

You can create a wrapper Card component
that takes the inner React component as an input



```jsx

function App() {

  return (
    <div style={{display: "flex"}}>
      <Card>
        hi there
      </Card>
      <Card>
        <div>
          hello from the 2nd card
        </div>
      </Card>
    </div>
  )
}

function Card({children}) {
  return <div style={{
    border: "1px solid black",
    padding: 10,
    margin: 10
  }}>
    {children}
  </div>
}

export default App
```

https://gist.github.com/hkirat/40e5e43e2afdb779e3b71946cf2d67ef

Checkpoint

React returns ✓, re-rendering ✓, key ✓, Wrapper components ✓,
useEffect, useMemo, useCallback, useRef,
Prop Drilling

Hooks

Until now, we're discussed useState

These functions that start with use are called hooks

Hooks in React are functions that allow you to "hook into" React state and lifecycle features from function components.

useEffect,
useMemo,
useCallback,
useRef,
useReducer
useContext
useLayoutEffect

Hooks

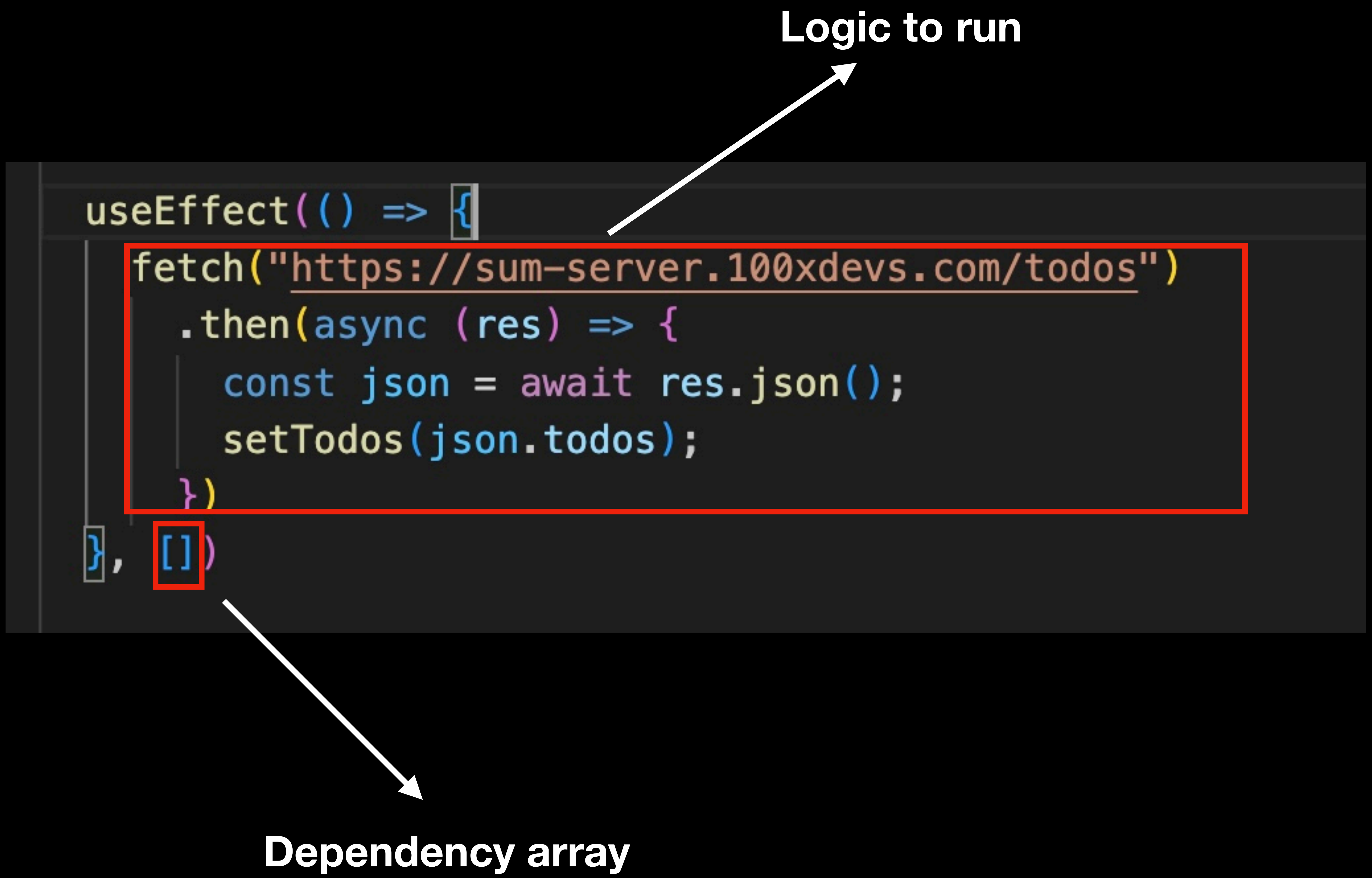
Until now, we’re discussed useState

These functions that start with use are called hooks

Hooks in React are functions that allow you to "hook into" React state and lifecycle features from function components.

useEffect

Hooks



```
useEffect(() => {
  fetch("https://sum-server.100xdevs.com/todos")
    .then(async (res) => {
      const json = await res.json();
      setTodos(json.todos);
    })
}, [])
```

The way to get code from backend is use fetch calls

Hooks

Create an app that polls the sum server
Gets the current set of TODOs
Renders it on screen

https://sum-server.100xdevs.com/todos

Solution
https://gist.github.com/hkirat/e10da900663a6f7a155c8505daae894f

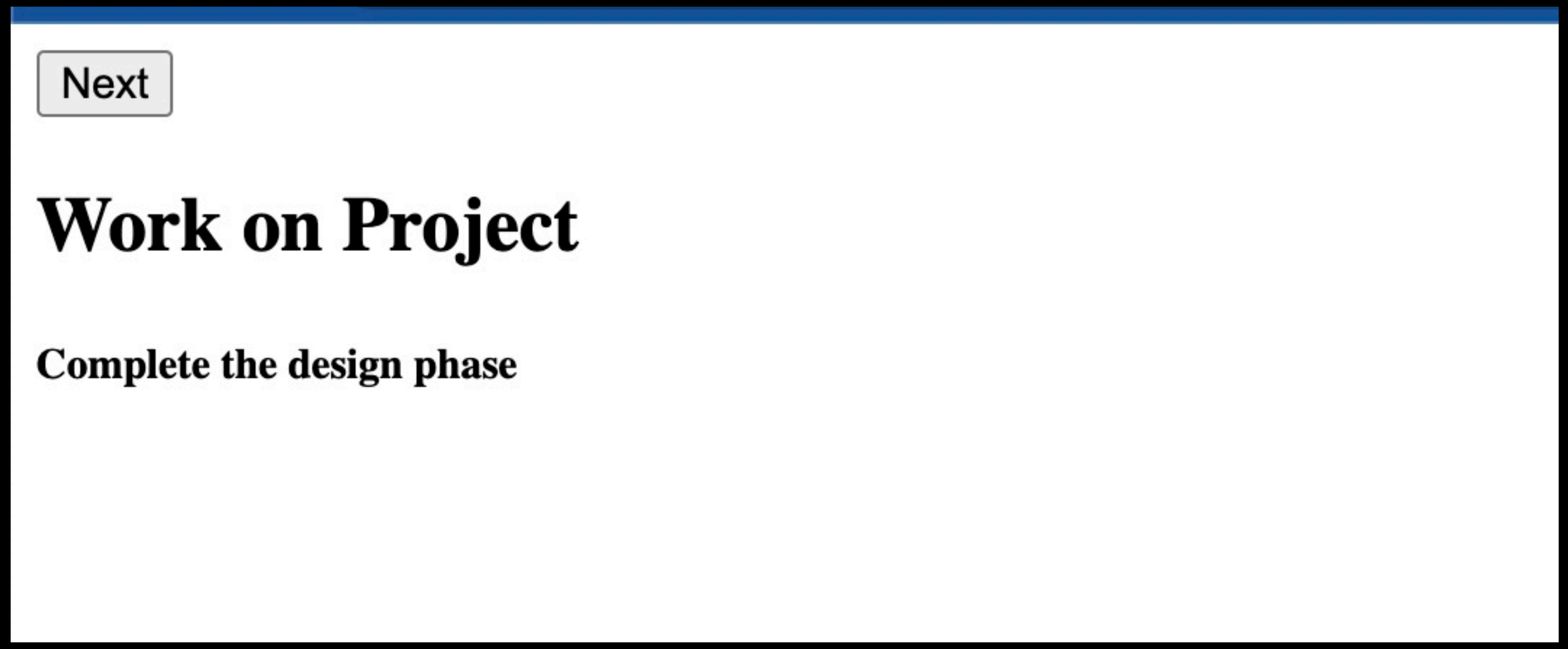
useEffect

Hooks

Assignment #2
Create a component that takes a todo id as input
And renders it by fetching it from the server
The parent component should have a button, clicking on which the next todo is fetched

https://sum-server.100xdevs.com/todo?id=1

Solution
https://gist.github.com/hkirat/0bfe829110da1ef01dd6a5593d115dba



Work on Project

Complete the design phase

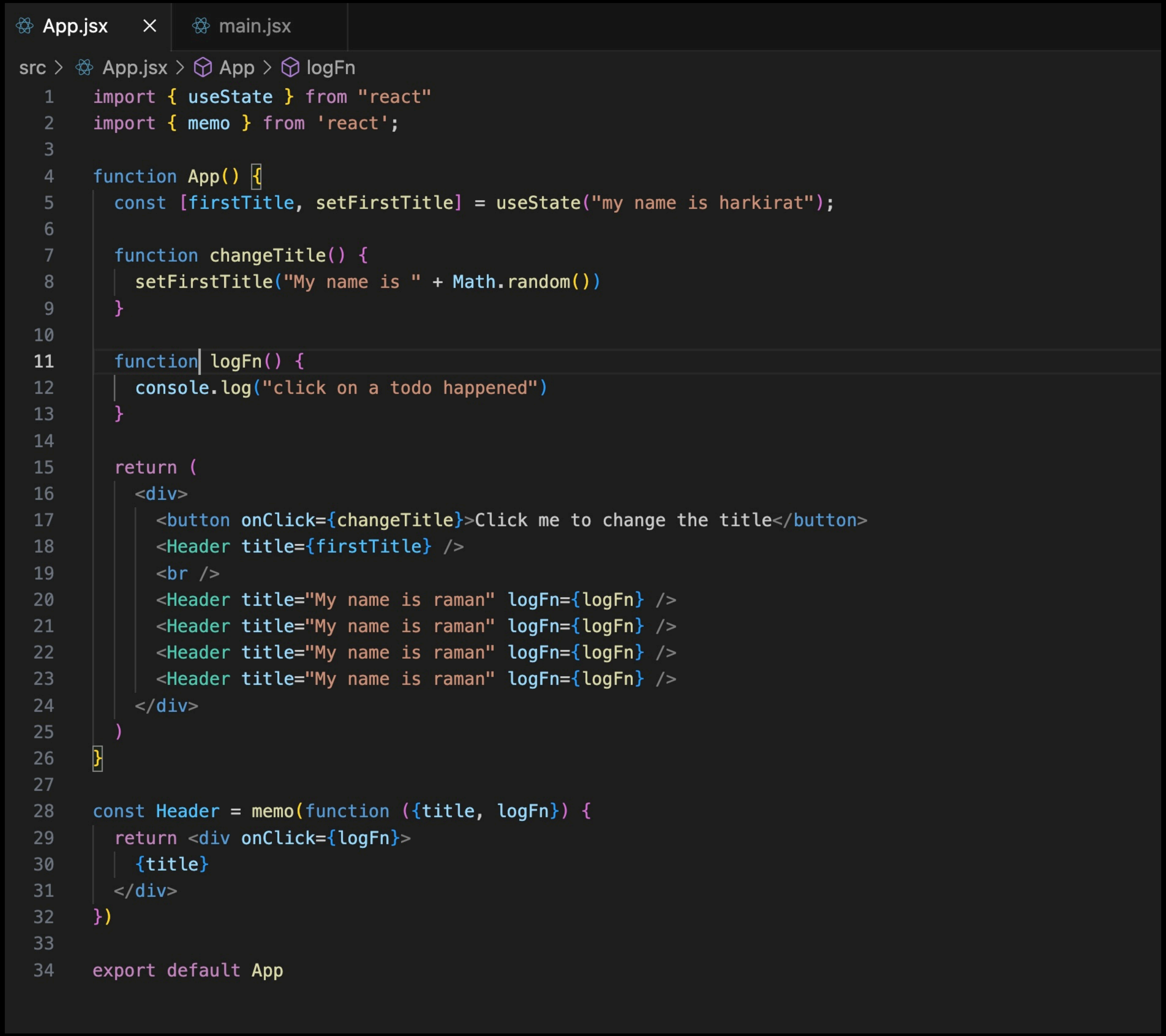
Hooks

useCallback is used to memorize a callback function. This is useful when you have a function that you pass down to child components and you don't want to re-create the function on every render, which could lead to unnecessary re-renders of the child components.

useCallback

Hooks

What's the issue in this code?



App.jsx × main.jsx

src > App.jsx > App > logFn

```jsx
import { useState } from "react"
import { memo } from 'react';

function App() {
  const [firstTitle, setFirstTitle] = useState("my name is harkirat");

  function changeTitle() {
    setFirstTitle("My name is " + Math.random())
  }

  function logFn() {
    console.log("click on a todo happened")
  }

  return (
    <div>
      <button onClick={changeTitle}>Click me to change the title</button>
      <Header title={firstTitle} />
      <br />
      <Header title="My name is raman" logFn={logFn} />
      <Header title="My name is raman" logFn={logFn} />
      <Header title="My name is raman" logFn={logFn} />
      <Header title="My name is raman" logFn={logFn} />
    </div>
  )
}

const Header = memo(function ({title, logFn}) {
  return <div onClick={logFn}>
    {title}
  </div>
})

export default App
```

useCallback

https://gist.github.com/hkirat/eb04182b0dd9793d425018475d2f7e54

Hooks

useMemo is used to memorize a value. This is useful when you have a computationally expensive calculation that you don't want to re-run on every render unless specific dependencies change.

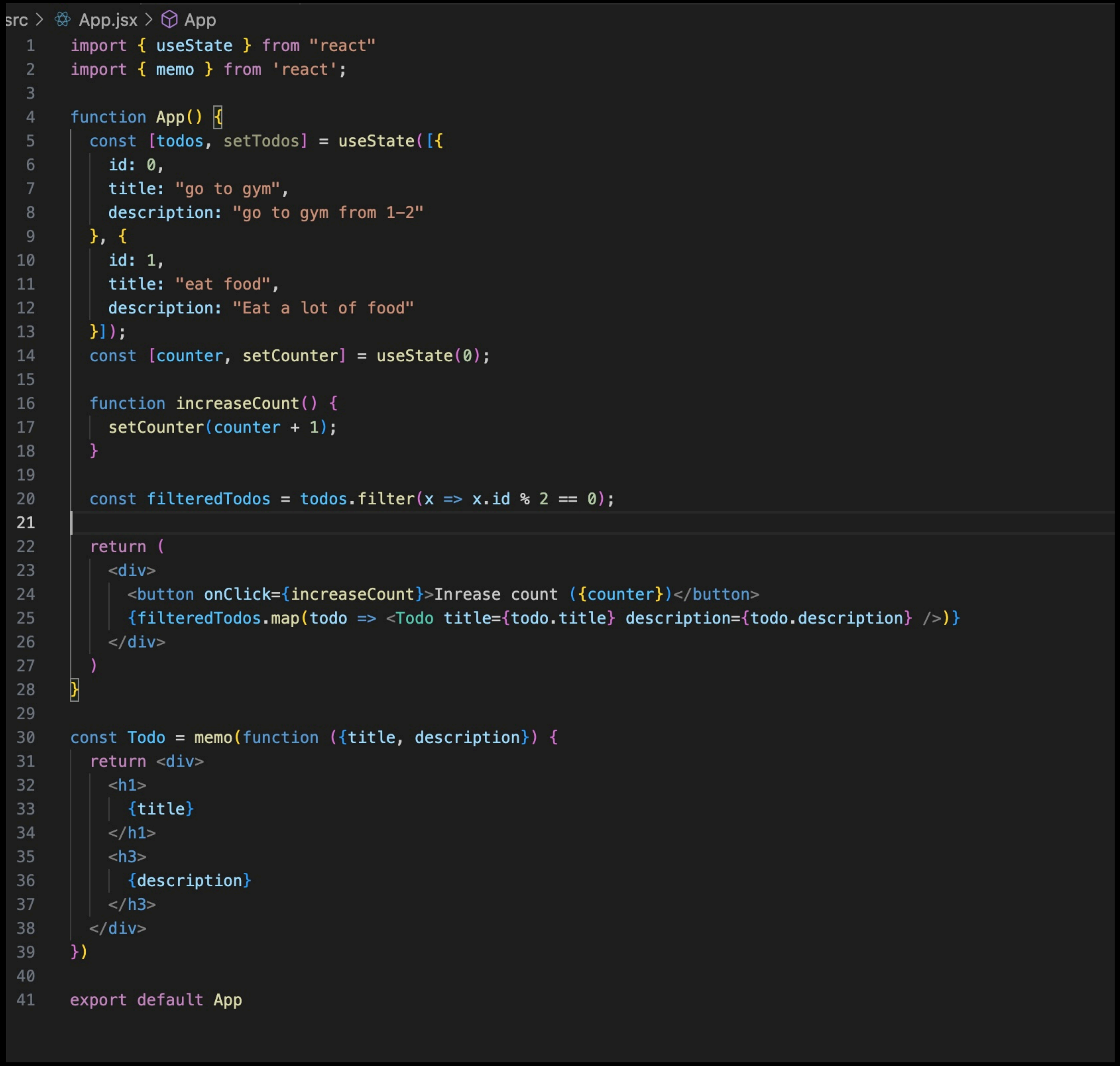
useMemo

Hooks

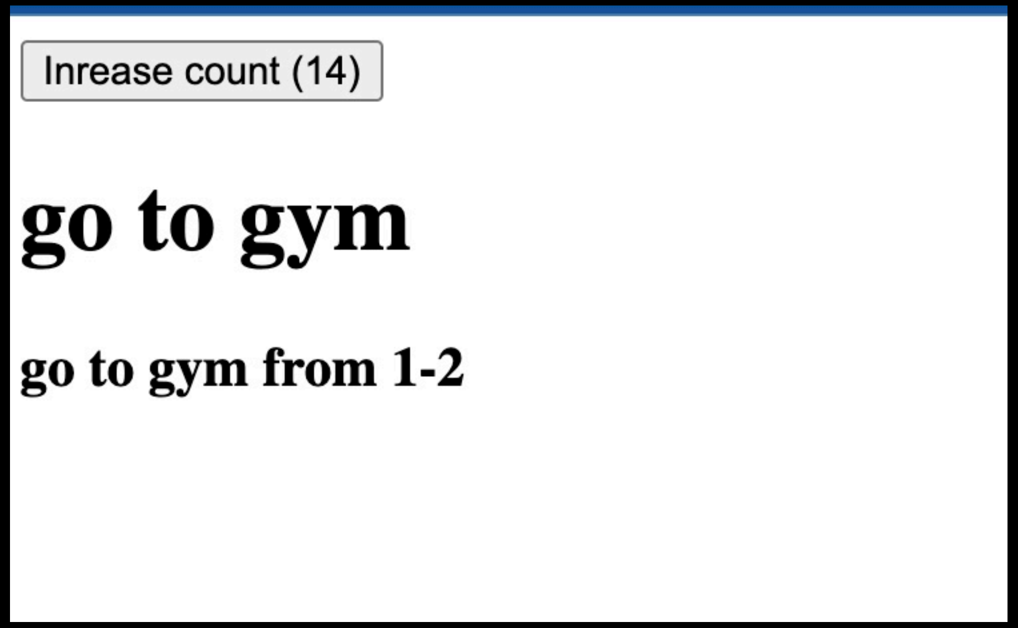
Assignment - Create an app that does two things -

1. Renders a list of all todos with even id
2. Lets a user increase a counter variable

Here's some boilerplate



```
src > App.jsx > App
import { useState } from "react"
import { memo } from 'react';

function App() {
  const [todos, setTodos] = useState([{
    id: 0,
    title: "go to gym",
    description: "go to gym from 1-2"
  }, {
    id: 1,
    title: "eat food",
    description: "Eat a lot of food"
  }]);
  const [counter, setCounter] = useState(0);

  function increaseCount() {
    setCounter(counter + 1);
  }

  const filteredTodos = todos.filter(x => x.id % 2 == 0);

  return (
    <div>
      <button onClick={increaseCount}>Inrease count ({counter})</button>
      {filteredTodos.map(todo => <Todo title={todo.title} description={todo.description} />)}
    </div>
  )
}

const Todo = memo(function ({title, description}) {
  return <div>
    <h1>
      {title}
    </h1>
    <h3>
      {description}
    </h3>
  </div>
})

export default App
```



Inrease count (14)

go to gym

go to gym from 1-2

useMemo

https://gist.github.com/hkirat/d66dee87cf1f03fc3509940efa782334

Hooks

useRef is a hook in React that is used to persist values across renders without causing a re-render of the component. It's often used for accessing DOM elements directly, storing a mutable reference to a value, or keeping track of a previous state/value. Here are a few real-world examples:

useRef

Hooks

useRef is a hook in React that is used to persist values across renders without causing a re-render of the component. It's often used for accessing DOM elements directly, storing a mutable reference to a value, or keeping track of a previous state/value. Here are a few real-world examples:

Assignment - Create a component which renders an input box and auto focusses on the input box when it renders

useRef