

# 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
1
2 function App() {
3   return (
4     <Header title="my name is harkirat" />
5     <Header title="My name is raman" />
6   )
7 }
8
9 function Header({title}) {
10   return <div>
11     {title}
12   </div>
13 }
14
15 export default App
16
```



# 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 > ...
1
2 function App() {
3   return (
4     <>
5       <Header title="my name is harkirat" />
6       <Header title="My name is raman" />
7     </>
8   )
9 }
10
11 function Header({title}) {
12   return <div>
13     {title}
14   </div>
15 }
16
17 export default App
18
```



```
src > App.jsx > App
1
2 function App() {
3   return (
4     <div>
5       <Header title="my name is harkirat" />
6       <Header title="My name is raman" />
7     </div>
8   )
9 }
10
11 function Header({title}) {
12   return <div>
13     {title}
14   </div>
15 }
16
17 export default App
18
```

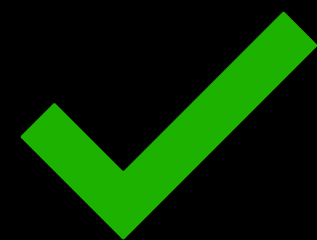


# React component return

Doesn't introduce an extra DOM element

```
src > App.jsx > ...
1
2 function App() {
3   return (
4     <>
5       <Header title="my name is harkirat" />
6       <Header title="My name is raman" />
7     </>
8   )
9 }
10
11 function Header({title}) {
12   return <div>
13     {title}
14   </div>
15 }
16
17 export default App
18
```

Slightly better



```
src > App.jsx > App
1
2 function App() {
3   return (
4     <div>
5       <Header title="my name is harkirat" />
6       <Header title="My name is raman" />
7     </div>
8   )
9 }
10
11 function Header({title}) {
12   return <div>
13     {title}
14   </div>
15 }
16
17 export default App
18
```



# Re-rendering in react

What exactly is a re-render?

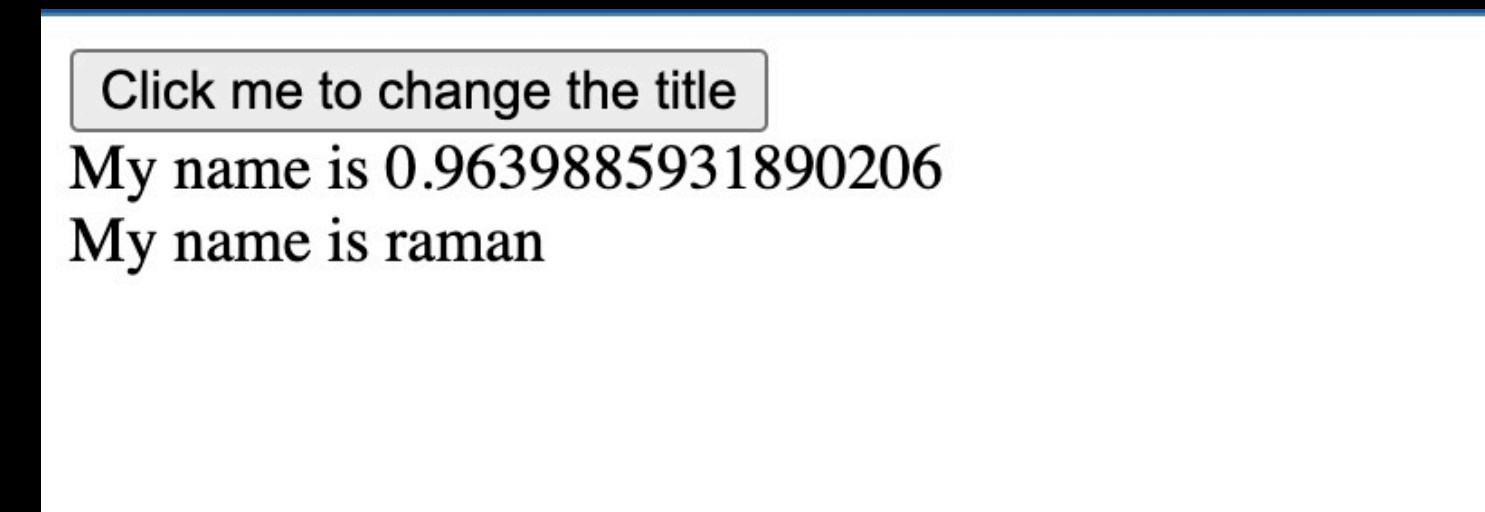
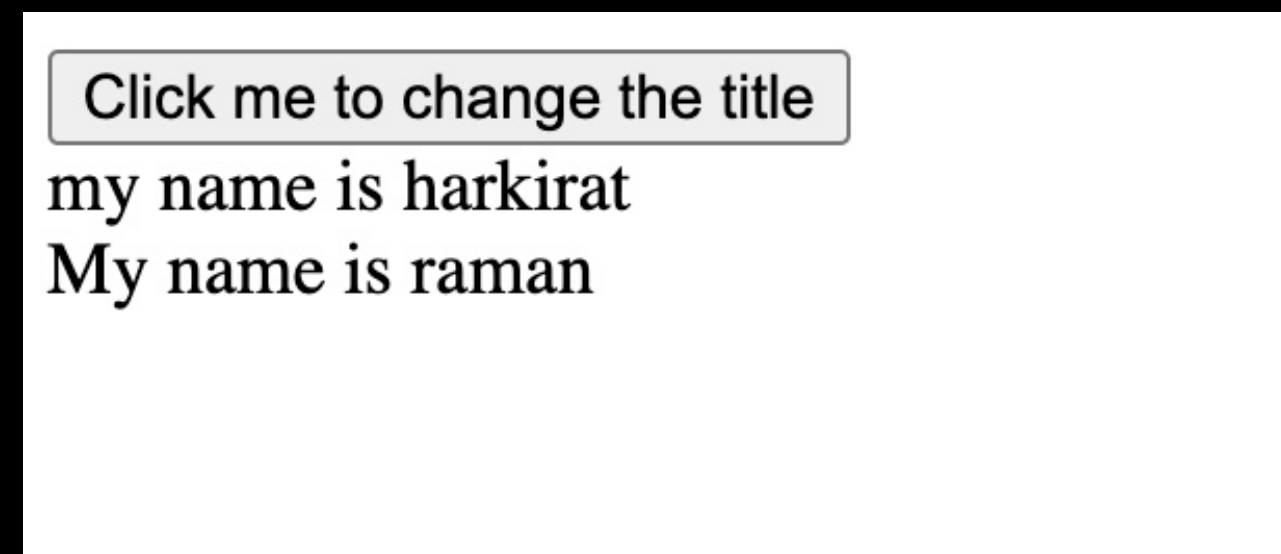
Please install react developer tools to visualise it

<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



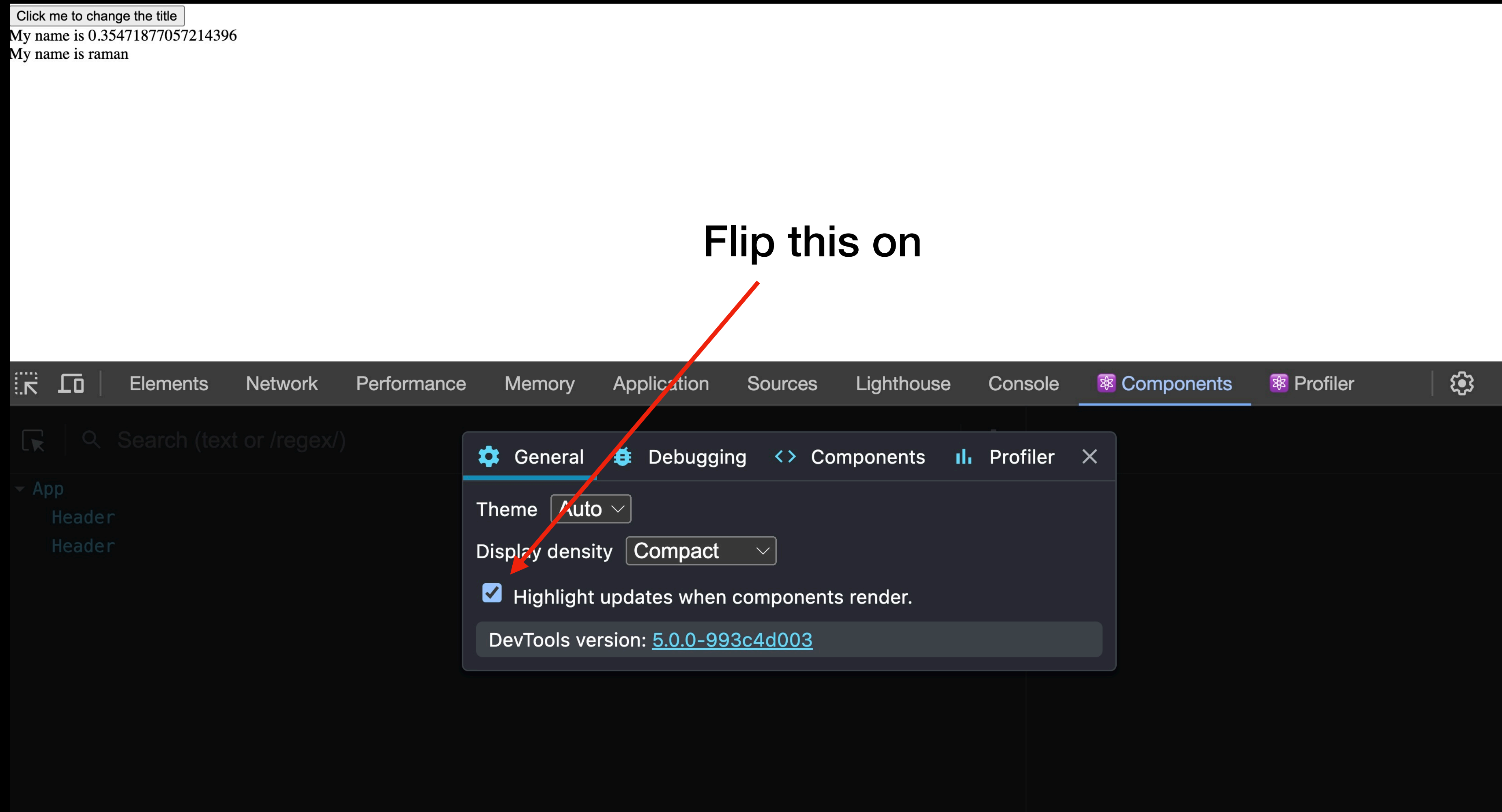
Hint (`Math.random()`) gives you a random number b/w 0-1)

<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





# 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 > ...
1  import { useState } from "react"
2
3  function App() {
4    const [firstTitle, setFirstTitle] = useState("my name is harkirat");
5
6    function changeTitle() {
7      setFirstTitle("My name is " + Math.random())
8    }
9
10   return (
11     <div>
12       <button onClick={changeTitle}>Click me to change the title</button>
13       <Header title={firstTitle} />
14       <Header title="My name is raman" />
15     </div>
16   )
17 }
18
19 function Header({title}) {
20   return <div>
21     {title}
22   </div>
23 }
24
25 export default App
26
```

# Re-rendering in react

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

```
src > App.jsx > ...
1  import { useState } from "react"
2
3  function App() {
4    const [firstTitle, setFirstTitle] = useState("my name is harkirat");
5
6    function changeTitle() {
7      setFirstTitle("My name is " + Math.random())
8    }
9
10   return (
11     <div>
12       <button onClick={changeTitle}>Click me to change the title</button>
13       <Header title={firstTitle} />
14       <Header title="My name is raman" />
15     </div>
16   )
17 }
18
19 function Header({title}) {
20   return <div>
21     {title}
22   </div>
23 }
24
25 export default App
26
```

Pushing the state down



```
src > App.jsx > ...
1  import { useState } from "react"
2
3  function App() {
4    return (
5      <div>
6        <HeaderWithButton />
7        <Header title="My name is raman" />
8      </div>
9    )
10 }
11
12 function HeaderWithButton() {
13   const [firstTitle, setFirstTitle] = useState("my name is harkirat");
14
15   function changeTitle() {
16     setFirstTitle("My name is " + Math.random())
17   }
18
19   return <>
20     <button onClick={changeTitle}>Click me to change the title</button>
21     <Header title={firstTitle} />
22   </>
23 }
24
25 function Header({title}) {
26   return <div>
27     {title}
28   </div>
29 }
30
31 export default App
32
```

<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 > ...
1  import { useState } from "react"
2
3  function App() {
4    const [firstTitle, setFirstTitle] = useState("my name is harkirat");
5
6    function changeTitle() {
7      setFirstTitle("My name is " + Math.random())
8    }
9
10   return (
11     <div>
12       <button onClick={changeTitle}>Click me to change the title</button>
13       <Header title={firstTitle} />
14       <Header title="My name is raman" />
15     </div>
16   )
17 }
18
19 function Header({title}) {
20   return <div>
21     {title}
22   </div>
23 }
24
25 export default App
26
```

Use React.memo

```
src > App.jsx > [x] default
1  import { useState } from "react"
2  import { memo } from 'react';
3
4  function App() {
5    const [firstTitle, setFirstTitle] = useState("my name is harkirat");
6
7    function changeTitle() {
8      setFirstTitle("My name is " + Math.random())
9    }
10
11   return (
12     <div>
13       <button onClick={changeTitle}>Click me to change the title</button>
14       <Header title={firstTitle} />
15       <br />
16       <Header title="My name is raman" />
17       <Header title="My name is raman" />
18       <Header title="My name is raman" />
19       <Header title="My name is raman" />
20     </div>
21   )
22 }
23
24 const Header = memo(function ({title}) {
25   return <div>
26     {title}
27   </div>
28 })
29
30 export default App
31
```

<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
4. A button in the top level App component to add a new TODO

# Keys in react

```
src > App.jsx > ...
 1  import { useState } from "react"
 2
 3  function App() {
 4    const [todos, setTodos] = useState([
 5      {
 6        title: "Go to gym",
 7        description: "Need to hit the gym from 7-9PM"
 8      }, {
 9        title: "Go to Clas",
10        description: "Need to go to the class from 4-7 PM"
11      }, {
12        title: "Eat foor",
13        description: "Need to eat food from 2-4 PM"
14      }
15    ])
16    return (
17      <div>
18        {todos.map(todo => <Todo title={todo.title} description={todo.description} />)}
19      </div>
20    )
21  }
22
23  function Todo({title, description}) {
24    return <div>
25      <h1>
26        {title}
27      </h1>
28      <h4>
29        {description}
30      </h4>
31    </div>
32  }
33
34  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

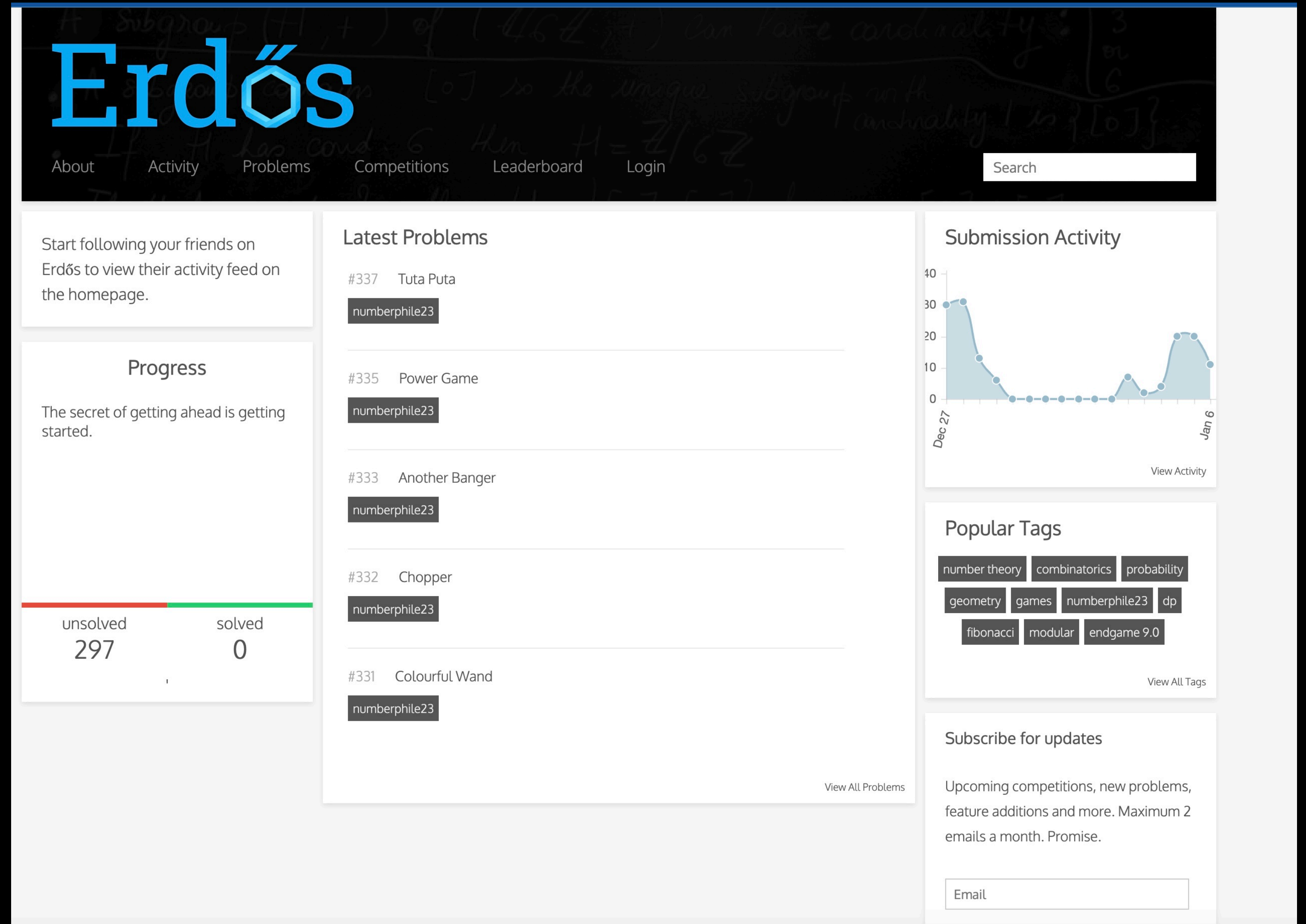
**Keys let react figure out if a  
TODO has been update,  
which has been delete,  
which has been added**

```
src > App.jsx > ...
1  import { useState } from "react"
2
3  let GLOBAL_ID = 4;
4
5  function App() {
6    const [todos, setTodos] = useState([
7      {
8        id: 1,
9        title: "Go to gym",
10       description: "Need to hit the gym from 7-9PM"
11     }, {
12       id: 2,
13       title: "Go to Clas",
14       description: "Need to go to the class from 4-7 PM"
15     }, {
16       id: 3,
17       title: "Eat foor",
18       description: "Need to eat food from 2-4 PM"
19     }
20   ])
21
22   function addTodo() {
23     setTodos([...todos, {
24       id: GLOBAL_ID++,
25       title: "new todo",
26       description: "new todo desc"
27     }])
28   }
29
30   return (
31     <div>
32       <button onClick={addTodo}>Add todo</button>
33       {todos.map((todo, index) => <Todo key={todo.id} title={todo.title} description={todo.description} />)}
34     </div>
35   )
36 }
37
38 function Todo({title, description}) {
39   return <div>
40     <h1>
41       {title}
42     </h1>
43     <h4>
44       {description}
45     </h4>
46   </div>
47 }
48
49 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



# 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

AboutActivityProblemsCompetitionsLeaderboardLogin

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 theorycombinatoricsprobability  
geometrygamesnumberphile23dp  
fibonacci  
modularendgame 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

```
1
2 function App() {
3
4   return (
5     <div style={{display: "flex"}}>
6       <Card>
7         hi there
8       </Card>
9       <Card>
10        <div>
11          hello from the 2nd card
12        </div>
13      </Card>
14    </div>
15  )
16 }
17
18 function Card({children}) {
19   return <div style={{
20     border: "1px solid black",
21     padding: 10,
22     margin: 10
23   }}>
24     {children}
25   </div>
26 }
27
28 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've 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.

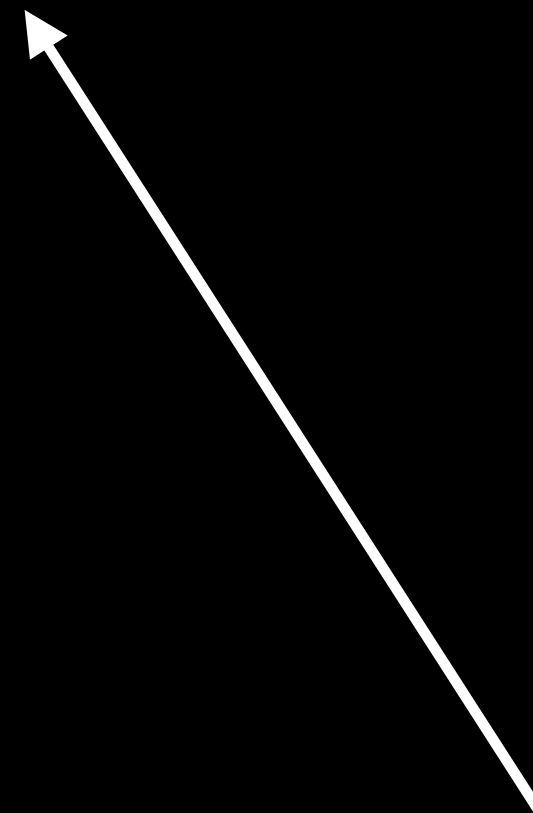
`useEffect,`  
`useMemo,`  
`useCallback,`  
`useRef,`  
`useReducer`  
`useContext`  
`useLayoutEffect`

# Hooks

Until now, we've 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

Logic to run

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

Dependency array

useEffect

# 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>

# 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
1  import { useState } from "react"
2  import { memo } from 'react';
3
4  function App() {
5    const [firstTitle, setFirstTitle] = useState("my name is harkirat");
6
7    function changeTitle() {
8      setFirstTitle("My name is " + Math.random())
9    }
10
11   function logFn() {
12     console.log("click on a todo happened")
13   }
14
15   return (
16     <div>
17       <button onClick={changeTitle}>Click me to change the title</button>
18       <Header title={firstTitle} />
19       <br />
20       <Header title="My name is raman" logFn={logFn} />
21       <Header title="My name is raman" logFn={logFn} />
22       <Header title="My name is raman" logFn={logFn} />
23       <Header title="My name is raman" logFn={logFn} />
24     </div>
25   )
26 }
27
28 const Header = memo(function ({title, logFn}) {
29   return <div onClick={logFn}>
30     {title}
31   </div>
32 })
33
34 export default App
```

useCallback

# 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

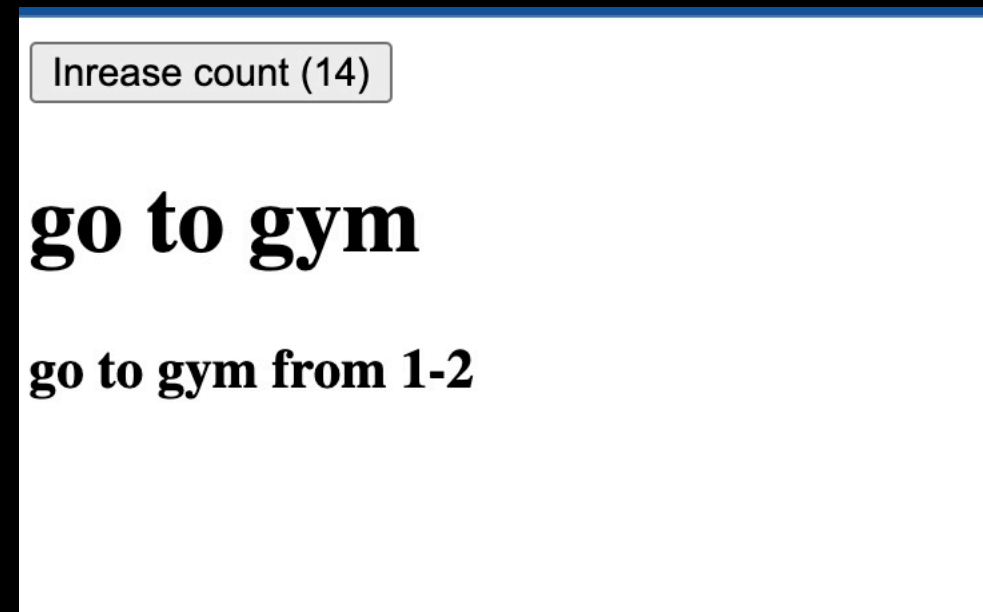
# Hooks

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
1  import { useState } from "react"
2  import { memo } from 'react';
3
4  function App() {
5    const [todos, setTodos] = useState([
6      {
7        id: 0,
8        title: "go to gym",
9        description: "go to gym from 1-2"
10     }, {
11       id: 1,
12       title: "eat food",
13       description: "Eat a lot of food"
14     }
15   ]);
16   const [counter, setCounter] = useState(0);
17
18   function increaseCount() {
19     setCounter(counter + 1);
20   }
21
22   const filteredTodos = todos.filter(x => x.id % 2 == 0);
23
24   return (
25     <div>
26       <button onClick={increaseCount}>Inrease count {counter}</button>
27       {filteredTodos.map(todo => <Todo title={todo.title} description={todo.description} />)}
28     </div>
29   )
30
31   const Todo = memo(function ({title, description}) {
32     return <div>
33       <h1>
34         {title}
35       </h1>
36       <h3>
37         {description}
38       </h3>
39     </div>
40   })
41
42   export default App
```



useMemo



# 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