

# React Recap

Reconcillation, state, useState, useEffect, useMemo, useCallback,  
useRef

# Reconciliation

Let's say you have a CA that files your tax returns



# Reconciliation

1. You give him all your bank info/demat info/job info/pan card
2. They reconcile all deposits/withdrawals/interest gains
3. They tell you final revenue, expense and profits



Give ur docs



Reconciles your balances





# Reconciliation

1. You give him all your bank info/demat info/job info/pan card
2. They reconcile all deposits/withdrawals/interest gains
3. They tell you final revenue, expense and profits



**Developer**

**State**  
Give ur docs



**React**

Reconciles your balances



**Re-render**

# Reconciliation

**Can you do taxes yourself - Yes**

**Should you do them yourself - No**

**Is it good for you to delegate the heavy task of calculating the taxes to CA - Yes**

**What do you give to the CA - Your Bank information/Pan card**

**How often does the CA re-compute taxes - Once a year**

**Does the CA have tricks to make calculation faster - Yes**



**Can you do DOM manipulation yourself - Yes**

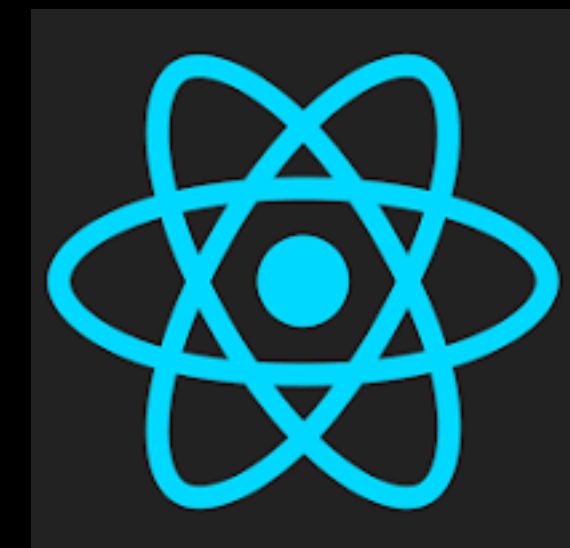
**Should you do it yourself - No**

**Is it good for you to delegate the heavy task of calculating the DOM changes to React - Yes**

**What do you give to React - The state**

**How often does react re-render - Any time state changes**

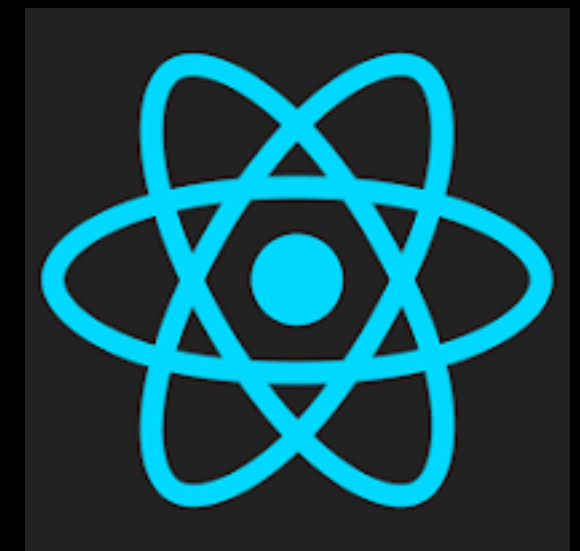
**Does React have tricks to make calculations faster - Yes**



# Reconciliation

## Basic react code

```
src > App.jsx > App
1  import { useState } from 'react'
2
3  function App() {
4    const [count, setCount] = useState(0)
5
6    return (
7      <div>
8        <button onClick={() => setCount((count) => count + 1)}>
9          count is {count}
10       </button>
11     </div>
12   )
13 }
14
15 export default App
16
```

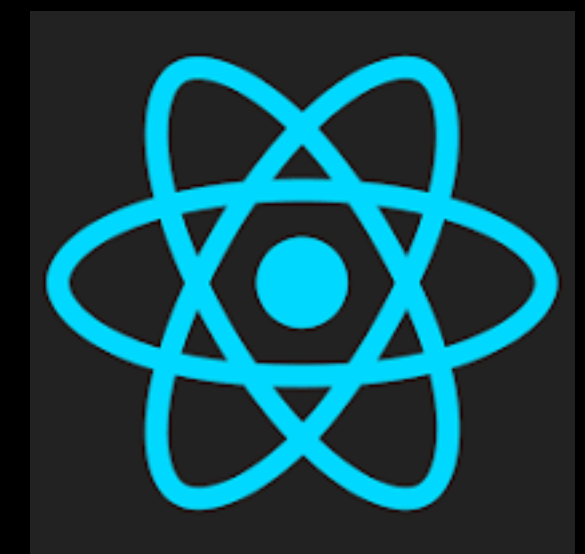




# Reconciliation

```
src > App.jsx > App
1  import { useState } from 'react'
2
3  function App() {
4    const [count, setCount] = useState(0)
5
6    return (
7      <div>
8        <button onClick={() => setCount((count) => count + 1)}>
9          count is {count}
10        </button>
11      </div>
12    )
13  }
14
15  export default App
16
```

State



# Reconciliation

src > App.jsx > App

```
1  import { useState } from 'react'
```

```
2
```

```
3  function App() {
```

```
4    const [count, setCount] = useState(0)
```

```
5
```

```
6    return (
```

```
7      <div>
```

```
8        <button onClick={() => setCount((count) => count + 1)}>
```

```
9          count is {count}
```

```
10        </button>
```

```
11      </div>
```

```
12    )
```

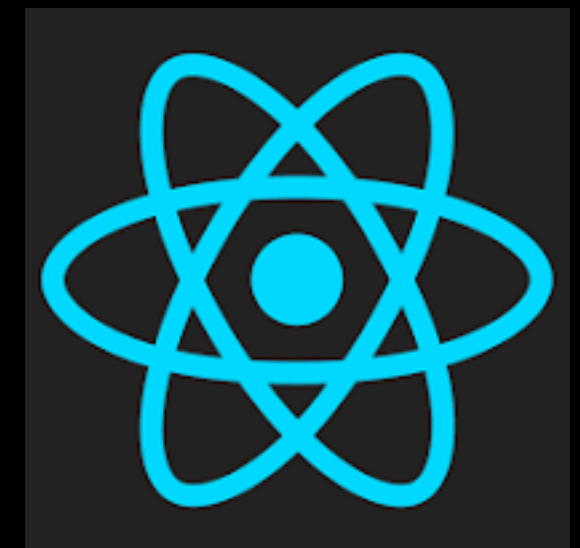
```
13  }
```

```
14
```

```
15  export default App
```

```
16
```

State update  
(Giving bank details to CA)





# Reconciliation

src > App.jsx > App

```
1 import { useState } from 'react'
```

```
2
```

```
3 function App() {
```

```
4   const [count, setCount] = useState(0)
```

```
5
```

```
6   return (
```

```
7     <div>
```

```
8       <button onClick={() => setCount((count) => count + 1)}>
```

```
9         count is {count}
```

```
10       </button>
```

```
11     </div>
```

```
12   )
```

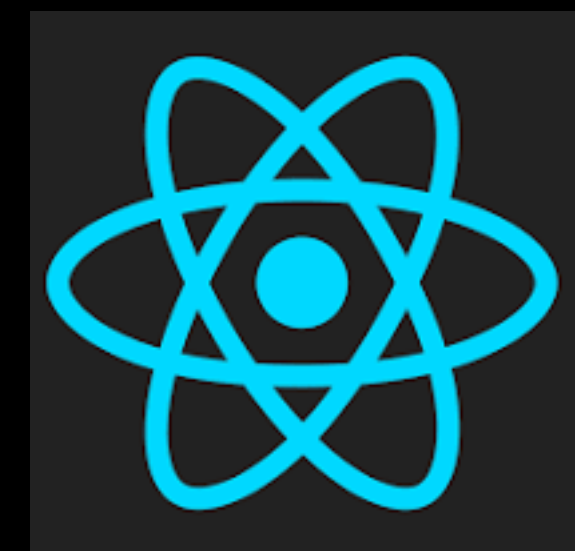
```
13 }
```

```
14
```

```
15 export default App
```

```
16
```

Re-render  
(CA calculates taxes)



# How do you define a re-render?

```
src > App.jsx > App
1  import { useState } from 'react'
2
3  function App() {
4    const [count, setCount] = useState(0)
5
6    return (
7      <div>
8        <button onClick={() => setCount((count) => count + 1)}>
9          count is {count}
10        </button>
11      </div>
12    )
13  }
14
15  export default App
16
```

How do you define a re-render?



# How do you define a re-render?

```
src > App.jsx > App
1  import { useState } from 'react'
2
3  function App() {
4    const [count, setCount] = useState(0)
5
6    return (
7      <div>
8        <button onClick={() => setCount((count) => count + 1)}>
9          count is {count}
10        </button>
11      </div>
12    )
13  }
14
15  export default App
16
```

**This function running  
means this component  
re-rendered**

**(Try putting a log inside)**

# React Recap

Reconciliation, state, useState, useEffect, useMemo, useCallback, use

# React Recap

Reconciliation, state, useState, useEffect, useMemo, useCallback, use



# useEffect

How do you get all the docs  
needed to submit to your CA?

More importantly, **how many times do you  
get the same docs?**



# useEffect

**When you're filing taxes, you might need to get your data from various sources before you give it to your CA (side effects)**

- 1. You might have to wait for 10 days before you can talk to your bank manager (setTimeout)**
- 2. You might have to go to your exchange broker's office to get your trading information**

**You will update your CA with this information as you get it**

**Q: Will you, in any case, do any of these more than once in a single year?**



# useEffect

App.jsx > ...

```
import { useState } from 'react'
```

```
function App() {
```

```
  const [exchangeData, setExchangeData] = useState({});
```

```
  const [bankData, setBankData] = useState({});
```

```
  fetch("https://google.com", async (res) => {  
    const json = await res.json();  
    setBankData(json);  
    // Assume it is { income: 100 }  
  });
```

1. You might have to wait for 10 days before you can talk to your bank manager (setTimeout)

```
  setTimeout(() => {  
    setExchangeData({  
      returns: 100  
    });  
  }, 1000);
```

2. You might have to go to your exchange broker's office to get your trading information

```
  const incomeTax = (bankData.income + exchangeData) * 0.3;
```

```
  return (
```

```
    <div>
```

```
    | hi there, your income tax returns are {incomeTax}
```

```
    </div>
```

```
  )
```

```
}
```



# useEffect

App.jsx > ...

```
import { useState } from 'react'
```

```
function App() {
```

```
  const [exchangeData, setExchangeData] = useState({});
```

```
  const [bankData, setBankData] = useState({});
```

```
  fetch("https://google.com", async (res) => {  
    const json = await res.json();  
    setBankData(json);  
    // Assume it is { income: 100 }  
  });
```

```
  setTimeout(() => {  
    setExchangeData({  
      returns: 100  
    });  
  }, 1000);
```

```
  const incomeTax = (bankData.income + exchangeData) * 0.3;
```

```
  return (
```

```
    <div>
```

```
    | hi there, your income tax returns are {incomeTax}
```

```
    </div>
```

```
  )
```

```
}
```

1. You might have to wait for 10 days before you can talk to your bank manager (setTimeout)

2. You might have to go to your exchange broker's office to get your trading information

**The problem -**

**When you get either one of the data, the component re-renders, which causes both the things to trigger again**

# useEffect

```
App.jsx
src > App.jsx > App
1  import { useEffect, useState } from 'react'
2
3  function App() {
4    const [exchangeData, setExchangeData] = useState({});
5    const [bankData, setBankData] = useState({});
6
7    useEffect(() => {
8      fetch("https://google.com", async (res) => {
9        const json = await res.json();
10       setBankData(json);
11       // Assume it is { income: 100 }
12     });
13   }, [])
14
15   useEffect(() => {
16     setTimeout(() => {
17       setExchangeData({
18         returns: 100
19       });
20     }, 1000);
21   }, [])
22
23   const incomeTax = (bankData.income + exchangeData) * 0.3;
24
25   return (
26     <div>
27       hi there, your income tax returns are {incomeTax}
28     </div>
29   )
30 }
31
```

1. You might have to wait for 10 days before you can talk to your bank manager (setTimeout)

2. You might have to go to your exchange broker's office to get your trading information

**The Solution -  
The useEffect hook**

# useEffect

```
App.jsx
src > App.jsx > App
1  import { useEffect, useState } from 'react'
2
3  function App() {
4    const [exchangeData, setExchangeData] = useState({});
5    const [bankData, setBankData] = useState({});
6
7    useEffect(() => {
8      fetch("https://google.com", async (res) => {
9        const json = await res.json();
10       setBankData(json);
11       // Assume it is { income: 100 }
12     });
13   }, []);
14
15   useEffect(() => {
16     setTimeout(() => {
17       setExchangeData({
18         returns: 100
19       });
20     }, 1000);
21   }, []);
22
23   const incomeTax = (bankData.income + exchangeData) * 0.3;
24
25   return (
26     <div>
27       hi there, your income tax returns are {incomeTax}
28     </div>
29   )
30 }
31
```

useEffect expects 2 inputs

What logic to run

On what state variable change should it run



# React Recap

Reconcillation, state, useState, useEffect, useMemo, useCallback,  
useRef

# useMemo

Let's say you have your crypto stored in 3 different exchanges (CoinDCX/WazirX/Binance)

You got the returns from all three places

You added them and gave it to the CA

Now you got your income report

**Will you re-calculate the sum of all the crypto returns?**



```

App.jsx > App > incomeTax
import { useEffect, useState } from 'react'

function App() {
  const [exchange1Data, setExchange1Data] = useState({});
  const [exchange2Data, setExchange2Data] = useState({});
  const [bankData, setBankData] = useState({});

  useEffect(() => {
    // Some operation to get the data
    setExchange1Data({
      returns: 100
    });
  }, [])

  useEffect(() => {
    // Some operation to get the data
    setExchange2Data({
      returns: 100
    });
  }, [])

  useEffect(() => {
    // Some operation to get the data
    setTimeout(() => {
      setBankData({
        income: 100
      });
    })
  }, [])

  const cryptoReturns = exchange1Data.returns + exchange2Data.returns;
  const incomeTax = (cryptoReturns + bankData.income) * 0.3

  return (
    <div>
      | hi there, your income tax returns are {incomeTax}
    </div>
  )
}

export default App

```

# useMemo

## The code

1. Gets data from exchange 1
2. Gets data from exchange 2
3. Gets income details from bank
4. Renders returns on screen

<https://gist.github.com/hkirat/65b98174c838eae46e0af0112806d25e>



# useMemo

```
App.jsx > App > incomeTax
import { useEffect, useState } from 'react'

function App() {
  const [exchange1Data, setExchange1Data] = useState({});
  const [exchange2Data, setExchange2Data] = useState({});
  const [bankData, setBankData] = useState({});

  useEffect(() => {
    // Some operation to get the data
    setExchange1Data({
      returns: 100
    });
  }, [])

  useEffect(() => {
    // Some operation to get the data
    setExchange2Data({
      returns: 100
    });
  }, [])

  useEffect(() => {
    // Some operation to get the data
    setTimeout(() => {
      setBankData({
        income: 100
      });
    })
  }, [])

  const cryptoReturns = exchange1Data.returns + exchange2Data.returns;
  const incomeTax = (cryptoReturns + bankData.income) * 0.3;

  return (
    <div>
      hi there, your income tax returns are {incomeTax}
    </div>
  )
}

export default App
```

**Should you recompute cryptoReturns  
If only bankData has changed in a render?**

<https://gist.github.com/hkirat/4bf35818f07e69b881185989ca9c8857>



# useMemo

# App.css # index.css App.jsx • main.jsx

src > App.jsx > App > cryptoReturns > useMemo() callback

```
1 import { useEffect, useMemo, useState } from 'react'
2
3 function App() {
4   const [exchange1Data, setExchange1Data] = useState({});
5   const [exchange2Data, setExchange2Data] = useState({});
6   const [bankData, setBankData] = useState({});
7
8   useEffect(() => {
9     // Some operation to get the data
10    setExchange1Data({
11      returns: 100
12    });
13  }, [])
14
15  useEffect(() => {
16    // Some operation to get the data
17    setExchange2Data({
18      returns: 100
19    });
20  }, [])
21
22  useEffect(() => {
23    // Some operation to get the data
24    setTimeout(() => {
25      setBankData({
26        income: 100
27      });
28    }, 5000)
29  }, [])
30
31  // useEffect, useMemo
32
33  const cryptoReturns = useMemo(() => {
34    return exchange1Data.returns + exchange2Data.returns;
35  }, [exchange1Data, exchange2Data]);
36
37
38  const incomeTax = (cryptoReturns + bankData.income) * 0.3
39
40  return (
41    <div>
```

```
const cryptoReturns = exchange1Data.returns + exchange2Data.returns;
```



```
const cryptoReturns = useMemo(() => {
  return exchange1Data.returns + exchange2Data.returns;
}, [exchange1Data, exchange2Data]);
```

# React Recap

Reconciliation, state, useState, useEffect, useMemo, useCallback, use

# useCallback

If you ever want to memoize a function, we use useCallback

```
App.jsx
src > App.jsx > default
1  import { useCallback, useEffect, useState } from 'react'
2
3  function App() {
4    const [exchange1Data, setExchange1Data] = useState({});
5    const [exchange2Data, setExchange2Data] = useState({});
6    const [bankData, setBankData] = useState({});
7
8    useEffect(() => {
9      // Some operation to get the data
10     setExchange1Data({
11       returns: 100
12     });
13   }, [])
14
15   useEffect(() => {
16     // Some operation to get the data
17     setExchange2Data({
18       returns: 100
19     });
20   }, [])
21
22   useEffect(() => {
23     // Some operation to get the data
24     setTimeout(() => {
25       setBankData({
26         income: 100
27       });
28     })
29   }, [exchange1Data, exchange2Data])
30
31   const calculateCryptoReturns = function() {
32     return exchange1Data.returns + exchange2Data.returns;
33   }
34
35   const incomeTax = (calculateCryptoReturns() + bankData.income) * 0.3
36
37   return (
38     <div>
39       | hi there, your income tax returns are {incomeTax}
40     </div>
41   )
```



# useCallback

If you ever want to memoize a function, we use useCallback

```
App.jsx
src > App.jsx > default
1  import { useCallback, useEffect, useState } from 'react'
2
3  function App() {
4    const [exchange1Data, setExchange1Data] = useState({});
5    const [exchange2Data, setExchange2Data] = useState({});
6    const [bankData, setBankData] = useState({});
7
8    useEffect(() => {
9      // Some operation to get the data
10     setExchange1Data({
11       returns: 100
12     });
13   }, [])
14
15   useEffect(() => {
16     // Some operation to get the data
17     setExchange2Data({
18       returns: 100
19     });
20   }, [])
21
22   useEffect(() => {
23     // Some operation to get the data
24     setTimeout(() => {
25       setBankData({
26         income: 100
27       });
28     })
29   }, [exchange1Data, exchange2Data])
30
31   const calculateCryptoReturns = function() {
32     return exchange1Data.returns + exchange2Data.returns;
33   }
34
35   const incomeTax = (calculateCryptoReturns() + bankData.income) * 0.3
36
37   return (
38     <div>
39       | hi there, your income tax returns are {incomeTax}
40     </div>
41   )
```



# React Recap

Reconcillation, state, useState, useEffect, useMemo, useCallback,  
useRef

# useRef

Let's say you want to do some tax evasion

You want to override what your CA calculated as your income tax

**How would u do it? You would report an incorrect value to the government**

# useRef

```
src > App.jsx > ...
1  import { useEffect, useRef } from 'react'
2
3  function App() {
4    const divRef = useRef();
5
6    useEffect(() => {
7      setTimeout(() => {
8        divRef.current.innerHTML = "10"
9      }, 5000);
10   }, [])
11
12   const incomeTax = 20000;
13
14   return (
15     <div>
16       | hi there, your income tax returns are <div ref={divRef}>{incomeTax}</div>
17     </div>
18   )
19 }
20
21 export default App
22
```

<https://gist.github.com/hkirat/2d3ddf08541d70d92d7f911b91fbd1b2>

# useRef

**This isn't a great use case for useRef**

**But you will find one good one in the assignment from this week**

<https://github.com/100xdevs-cohort-2/assignments/blob/master/week-6/3-use-ref/src/components/Assignment1.jsx>

<https://github.com/100xdevs-cohort-2/assignments/blob/master/week-6/3-use-ref/src/components/Assignment2.jsx>