React Recap

Reconcilliation, 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

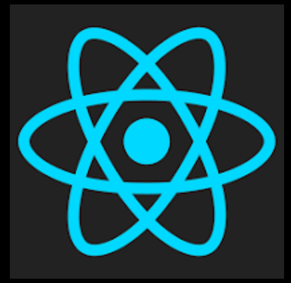
Re-render

Reconciles your balances

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

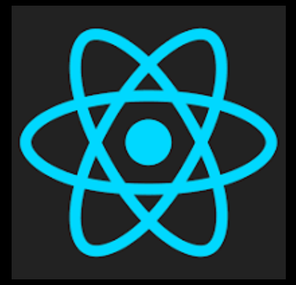


Reconciliation

Basic react code

src > App.jsx > App

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

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

  return (
    <div>
      <button onClick={() => setCount((count) => count + 1)}>
        count is {count}
      </button>
    </div>
  )
}

export default App

```

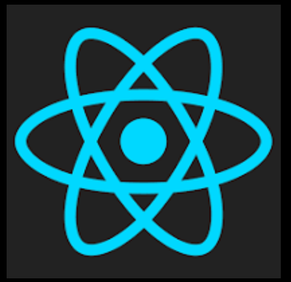


Reconciliation

src > App.jsx > App

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

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

  return (
    <div>
      <button onClick={() => setCount((count) => count + 1)}>
        count is {count}
      </button>
    </div>
  )
}

export default App

```

State

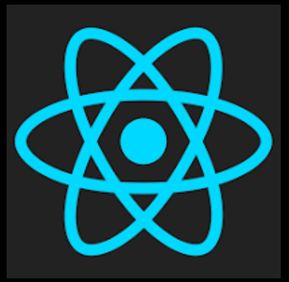


Reconciliation

src > App.jsx > App

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

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

  return (
    <div>
      <button onClick={() => setCount((count) => count + 1)}>
        count is {count}
      </button>
    </div>
  )
}

export default App

```

State update
(Giving bank details to CA)

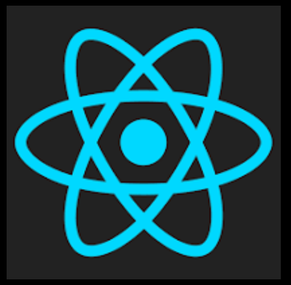


Reconciliation

src > App.jsx > App

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

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

  return (
    <div>
      <button onClick={() => setCount((count) => count + 1)}>
        count is {count}
      </button>
    </div>
  )
}

export default App

```

Re-render
(CA calculates taxes)



How do you define a re-render?

src > App.jsx > App

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

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

  return (
    <div>
      <button onClick={() => setCount((count) => count + 1)}>
        count is {count}
      </button>
    </div>
  )
}

export default App

```

How do you define a re-render?

How do you define a re-render?

src > App.jsx > App

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

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

  return (
    <div>
      <button onClick={() => setCount((count) => count + 1)}>
        count is {count}
      </button>
    </div>
  )
}

export default App

```

This function running means this component re-rendered

(Try putting a log inside)

React Recap

Reconcilliation, state, useState, useEffect, useMemo, useCallback, use

React Recap

Reconcilliation, 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 > ...

```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

https://gist.github.com/hkirat/81084e0922b1a73d1a8fc29d96013131

useEffect

App.jsx > ...

```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

```jsx
import { useEffect, useState } from 'react'

function App() {
  const [exchangeData, setExchangeData] = useState({});
  const [bankData, setBankData] = useState({});

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

  useEffect(() => {
    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 Solution -
The useEffect hook

useEffect

App.jsx

src > App.jsx > App

```jsx
import { useEffect, useState } from 'react'

function App() {
  const [exchangeData, setExchangeData] = useState({});
  const [bankData, setBankData] = useState({});

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

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

  }, [])
  const incomeTax = (bankData.income + exchangeData) * 0.3;

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

```

useEffect expects 2 inputs

What logic to run

On what state variable change should it run

React Recap

Reconcilliation, 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?

useMemo

App.jsx > App > incomeTax

```jsx
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
```

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

```jsx
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

```jsx
import { useEffect, useMemo, 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
      });
    }, 5000)
  }, [])

  // useEffect,  useMemo

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


  const incomeTax = (cryptoReturns + bankData.income) * 0.3

  return (
    <div>
```

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



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

React Recap

Reconcilliation, state, useState, useEffect, useMemo, useCallback, use

useCallback

App.jsx

src > App.jsx > default

```jsx
import { useCallback, 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
      });
    })
  }, [exchange1Data, exchange2Data])

  const calcuateCryptoReturns = function() {
    return exchange1Data.returns + exchange2Data.returns;
  }

  const incomeTax = (calcuateCryptoReturns() + bankData.income) * 0.3

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

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

useCallback

App.jsx

src > App.jsx > default

```jsx
import { useCallback, 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
      });
    })
  }, [exchange1Data, exchange2Data])

  const calcuateCryptoReturns = function() {
    return exchange1Data.returns + exchange2Data.returns;
  }

  const incomeTax = (calcuateCryptoReturns() + bankData.income) * 0.3

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

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

React Recap

Reconcilliation, 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 > ...

```jsx
import { useEffect, useRef } from 'react'

function App() {
  const divRef = useRef();

  useEffect(() => {
    setTimeout(() => {
      divRef.current.innerHTML = "10"
    }, 5000);
  }, [])

  const incomeTax = 20000;

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

export default App

```

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