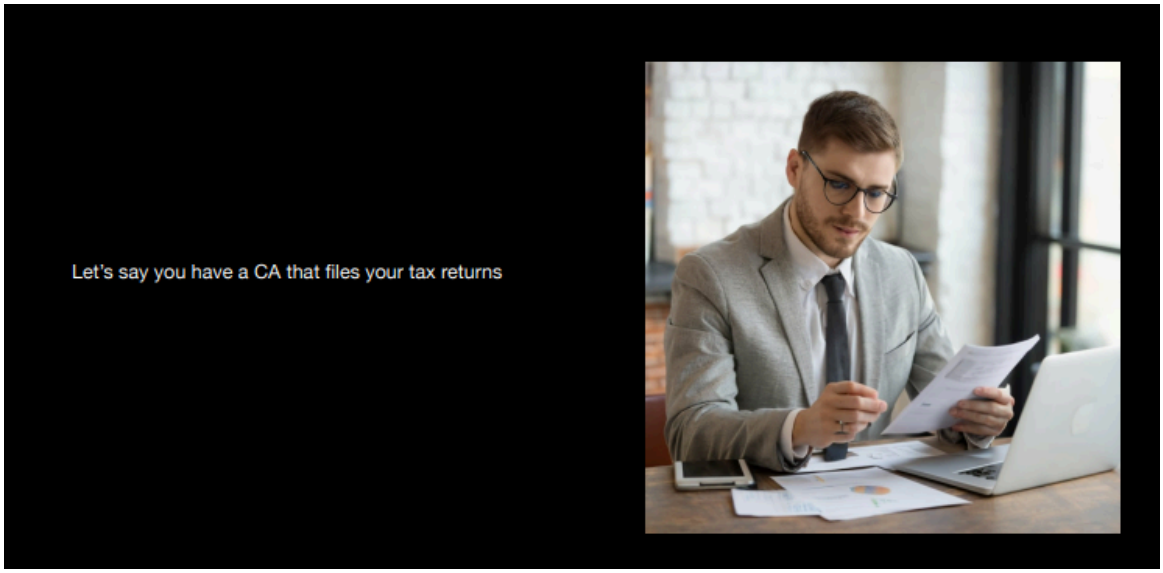
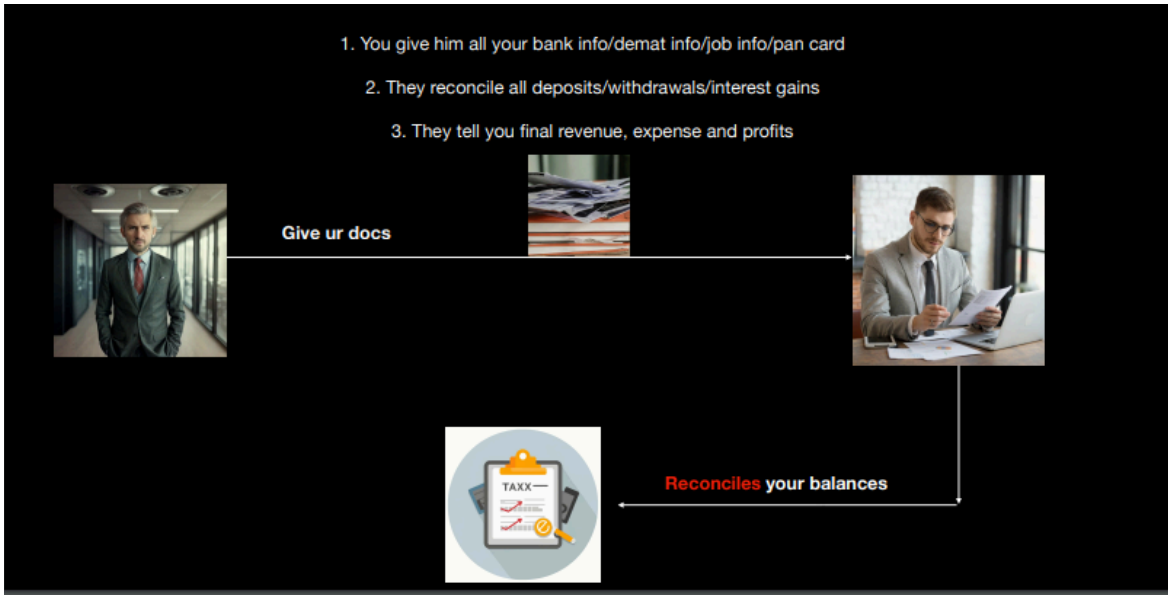
React Recap

(Reconciliation, state, useState, useEffect, useMemo, useCallback, useRef)

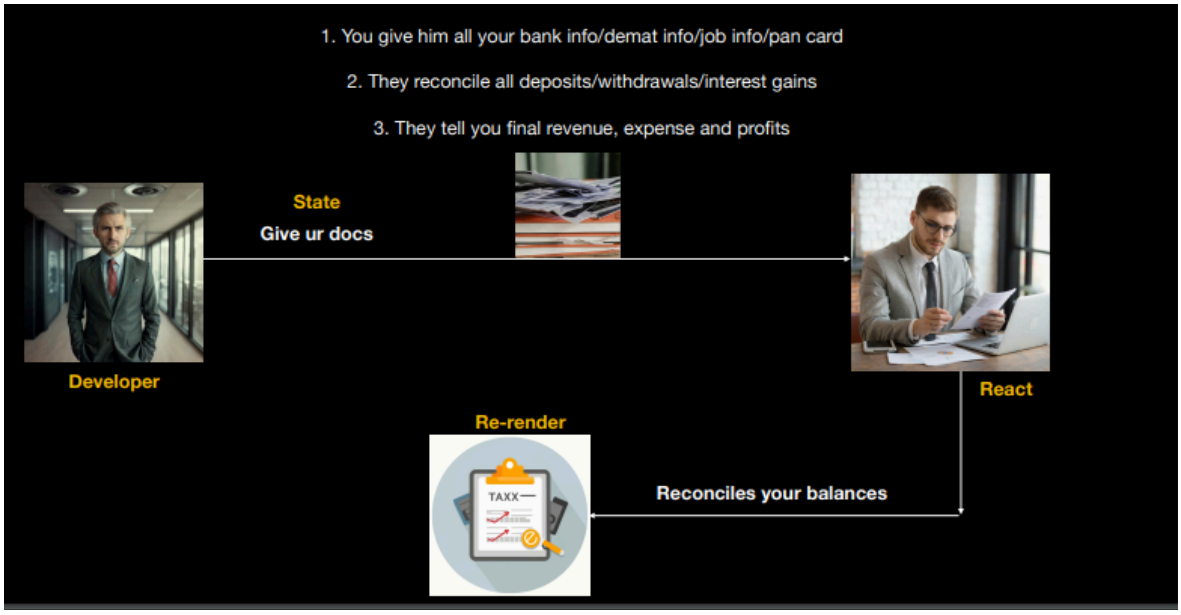
Reconciliation



How the process looks like

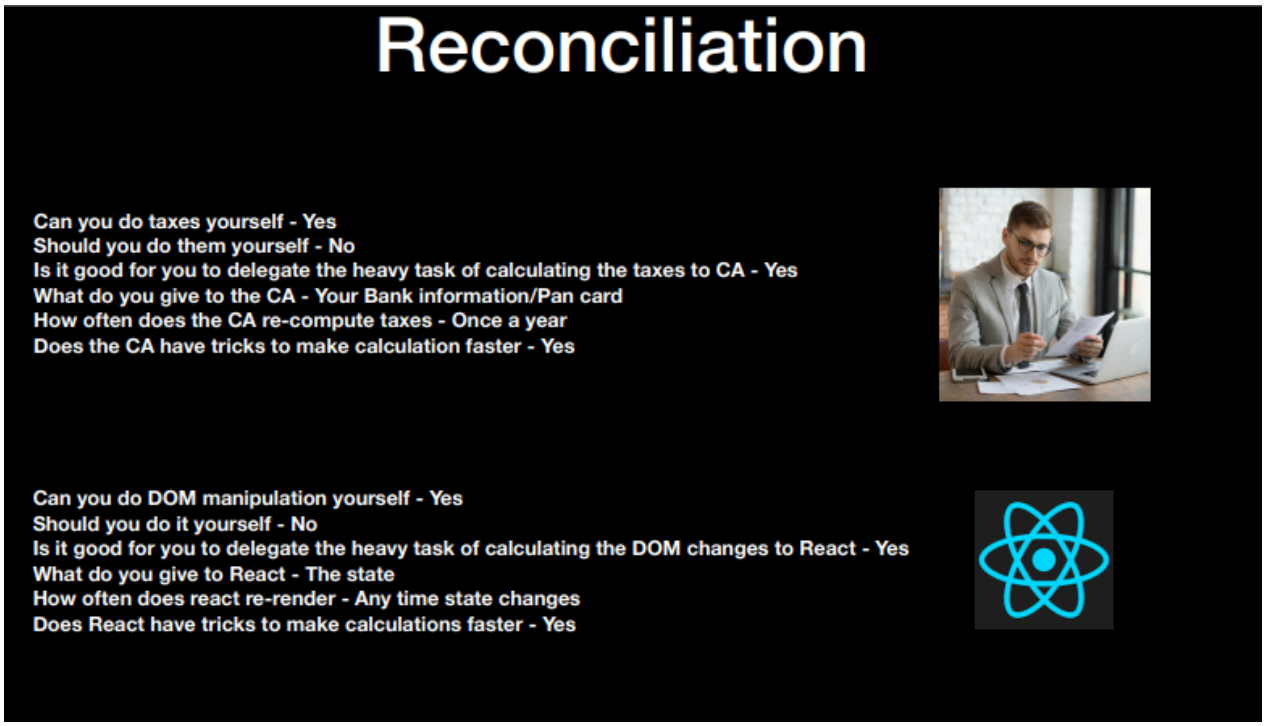


Lets see this example wrt React



It is good idea to delegate the tax etc calculation to the CA .

Reconciliation is the process of taking the current state finding the diff from existing state, reconciling how DOM should look like and then putting it on DOM



Basic React code:

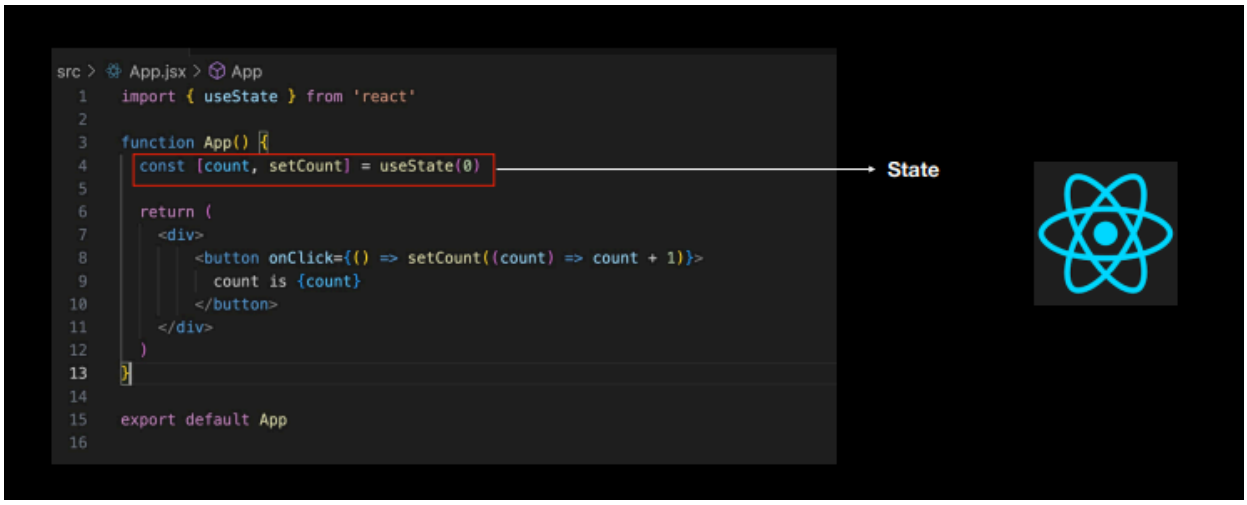
```jsx
import { useState } from "react";

function App(){
  const [count, setCount] = useState(0);
  return <div>
    <button onClick={()=>{
     setCount(count+1)
    }}>count is {count}</button>
  </div>
}

export default App;
```

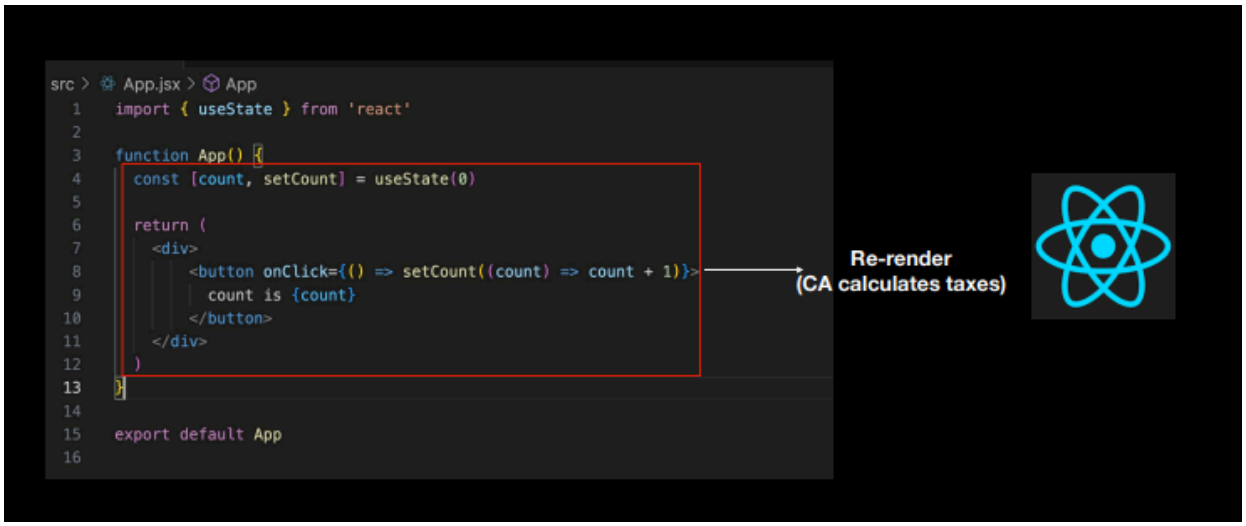
State is dynamic set of thing on our website.

useState is hook , setCount is function which let us update count then we can update the button content.

React understand what changes in DOM .

react-dom library has all the document.getElementById etc





How do we define re-rendering??

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?

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

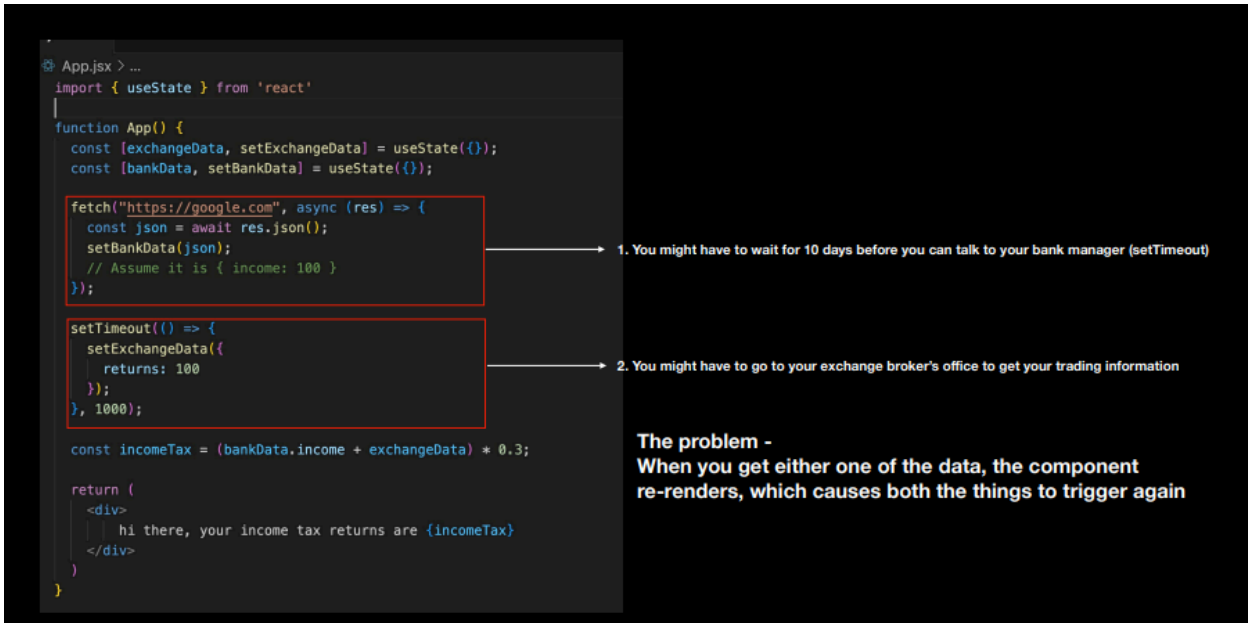
You will update your CA with this information as you get it

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

```
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.returns) * 0.3;

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

export default App
```

We have two state variables , our sources of income can be bank and maybe crypto.

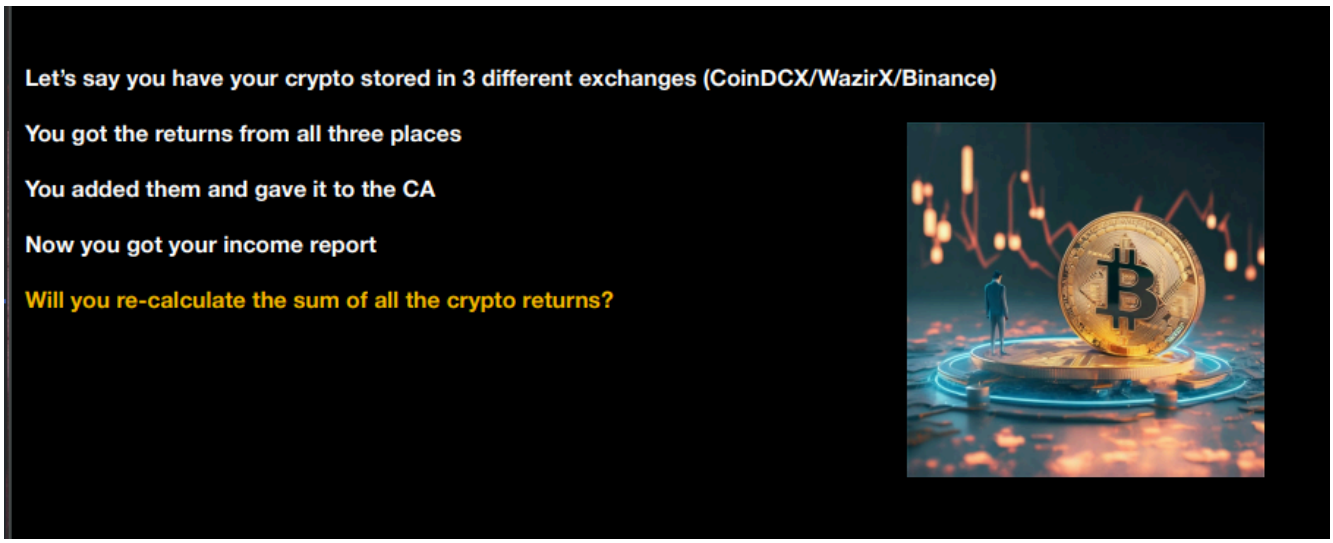


It also infinitely re-render the main reason is that we change the state inside the setTimeout which results in re-renders

This code only run once when the component mounts.

Or when certain changes occur in bank data.

useMemo()



Sum of crypto return do not depend on the other returns

```
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

Should you recompute cryptoReturns If only bankData has changed?

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

Will run again after 5 seconds.

Let's fix this :

```
  const cryptoReturns = useMemo(()=>{

   return exchange1Data.returns + exchange2Data.returns;
  },[exchange1Data,exchange2Data])
```

useCallback()

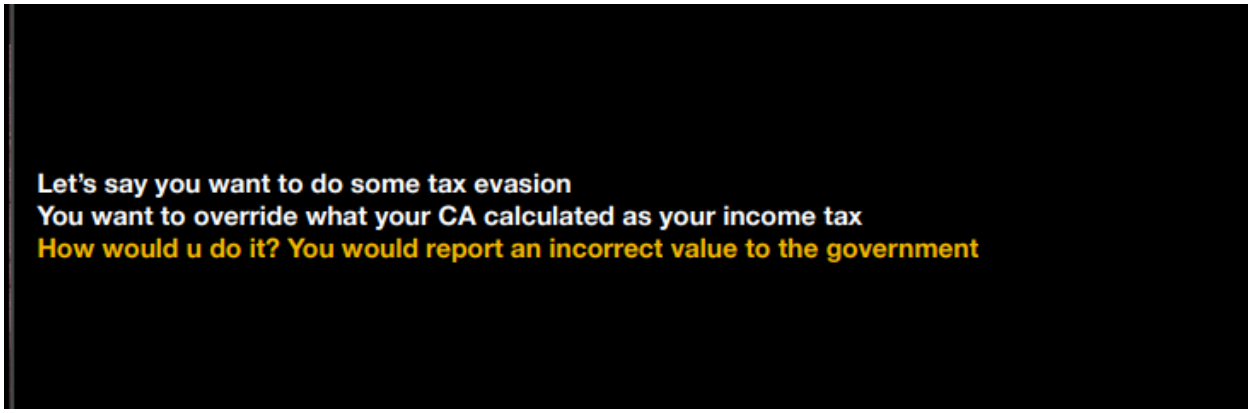
If you ever want to memoize a function we will use useCallback

```
  // useCallback is not about minimizing the amount of code that is run
  // useCallback is about not rendering a child component, if the function
hasnt/doesn't need to change across renders.

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

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

```
function a(){
  console.log("hi");
}
```

```
function b(){
  console.log("hi");
}
a==b; //false
```

useRef()




```
import { useCallback, useEffect, useRef, useState } 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
```

Our income tax return is initially 20k then it will be 10 after 5 seconds.

We have overwritten React.