

React

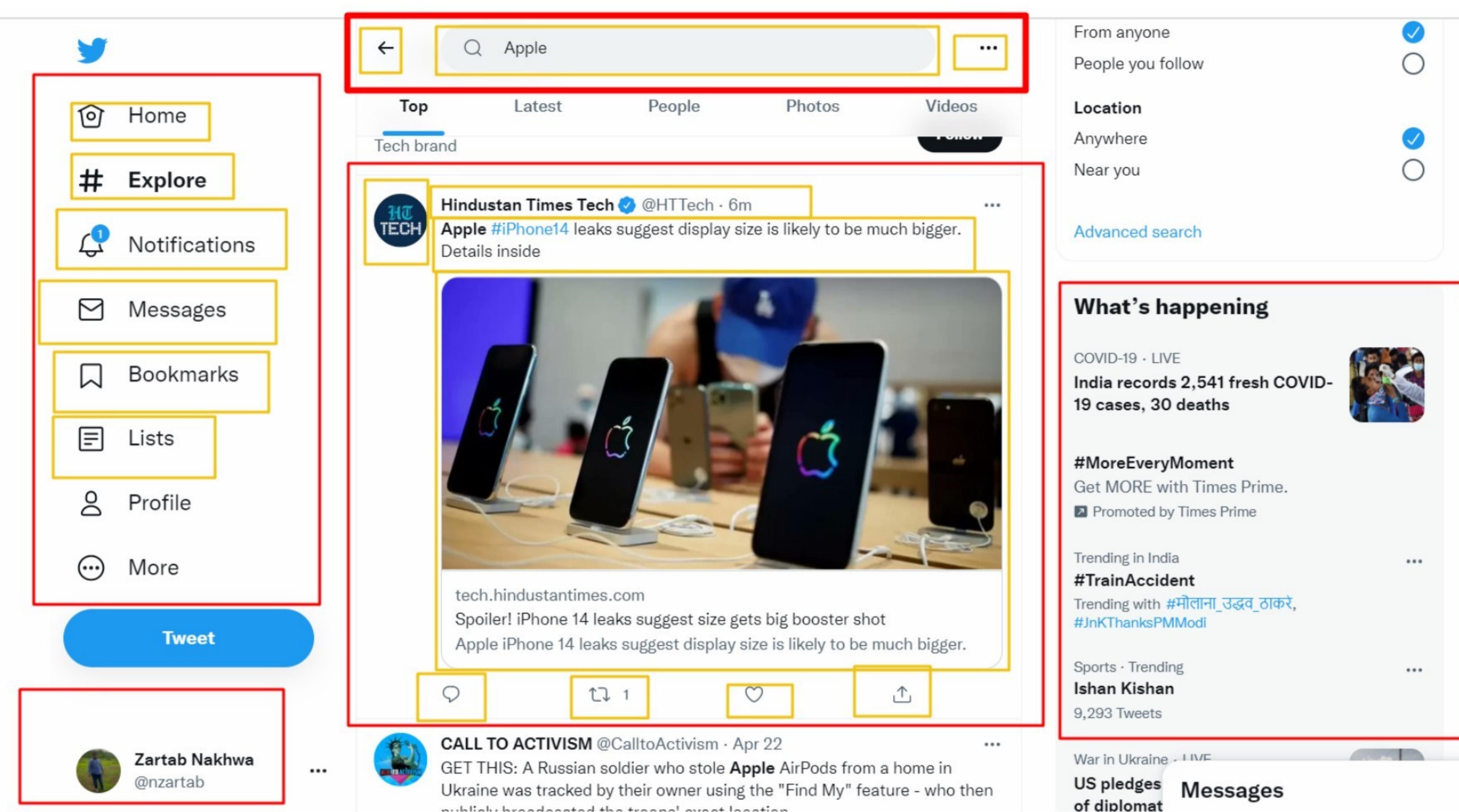
UI Library

React was

developed at
Facebook

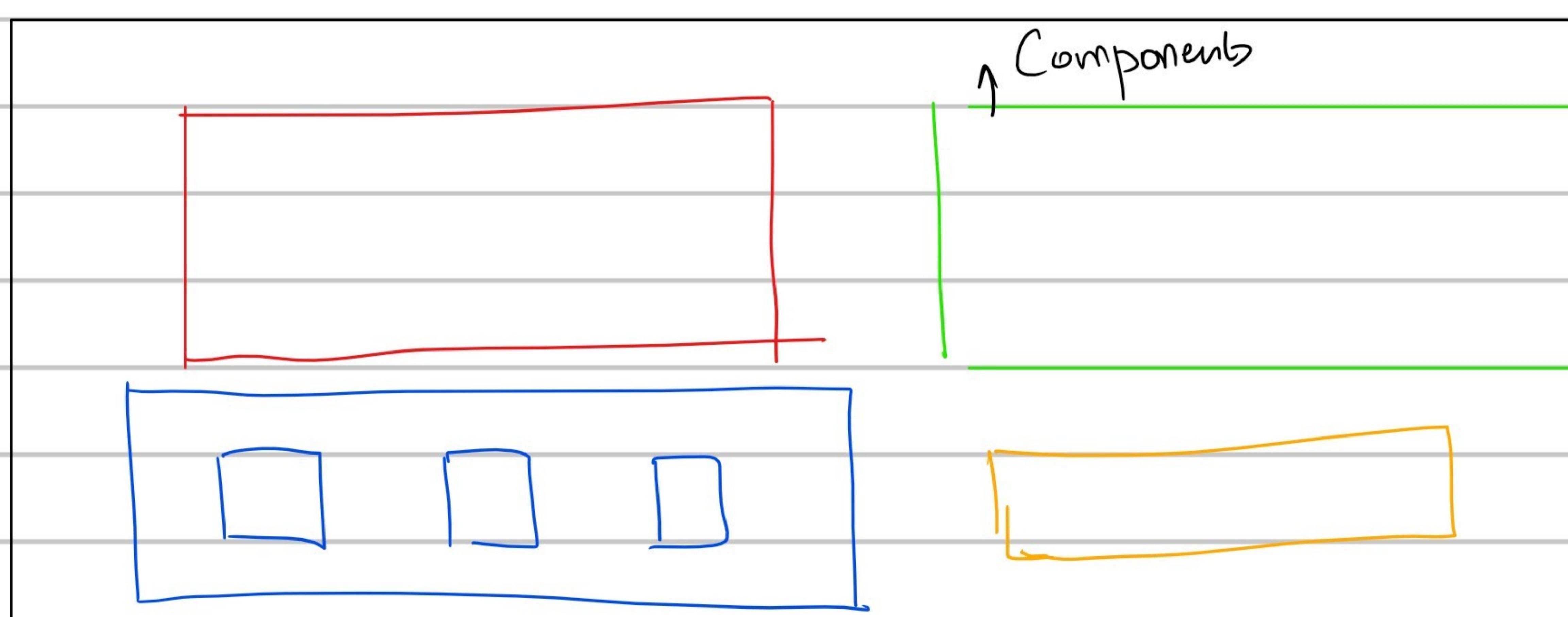
JavaScript based front end
→ UI library

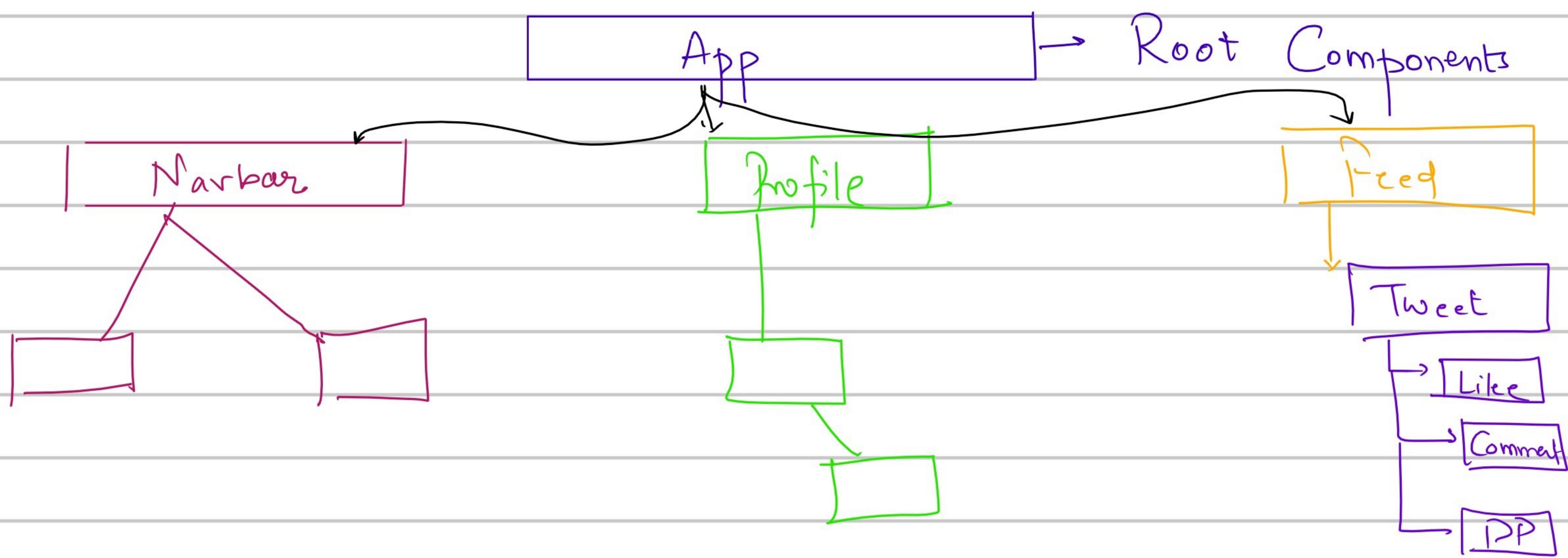
React is made up of Components !!



Component

piece of UI; which has its own STATE & LIFECYCLE





~~SPA~~

ES - 6 Refresher

- * let v/s const v/s var
- * map & filter
- * objects
- * object destructuring
- * this keyword
- * spread operator
- * arrow functions
- * class & inheritance
- * Modules ~~&~~ Named & Default Exports

* let vs var vs const

Var

Scope → var declarations are globally scoped or if used inside a function anywhere
→ update + redeclare
functionally scoped

let

Scope → block scope → It will only be available in the block it is declared
→ update + no redeclaration

const → Block scoped → maintains a constant value

→ no update + no redeclaration

* Arrow Functions *

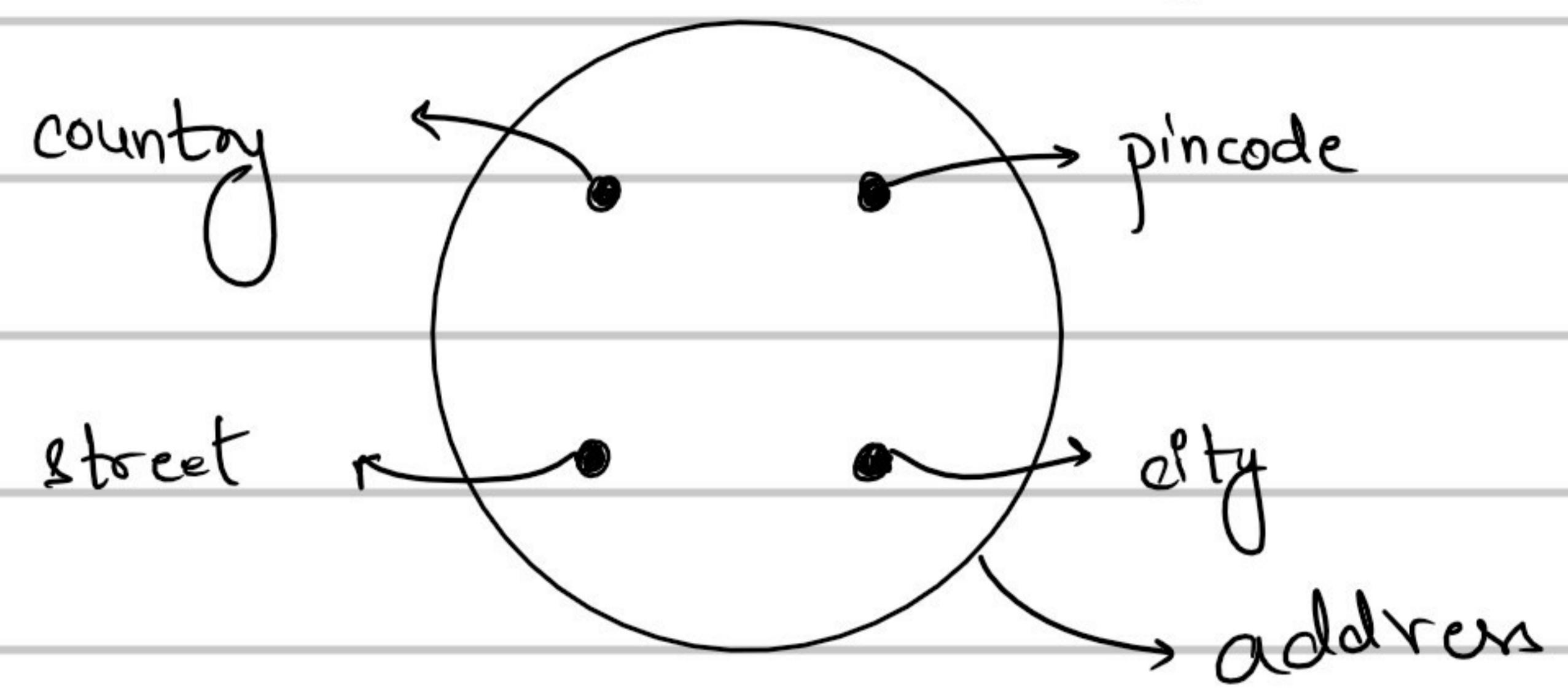
} cleaner way of creating traditional JS functions

let someVariable = (arg1, arg2, ... argN) => {
 statements
}

Arrow functions are either assigned to a variable or they are used as an argument to be passed to some other function

Object Destructuring

```
const {street, city} = address
```



* Spread Operator *

...
...

object

array.

Spreads the

elements of source array/object

Named & Default Exports

Named
Export

export

Person.js

class Person {

}

export function m1() {

}

export function m2() {

}

export let someValue = 50

Default Export

export default taxRate = 18

tax.js

Home-programming

export default

export

class

export

function

export

variable

trading.js

```
import taxRate from 'tax.js'  
import {Person, m1, m2} from 'person.js'
```

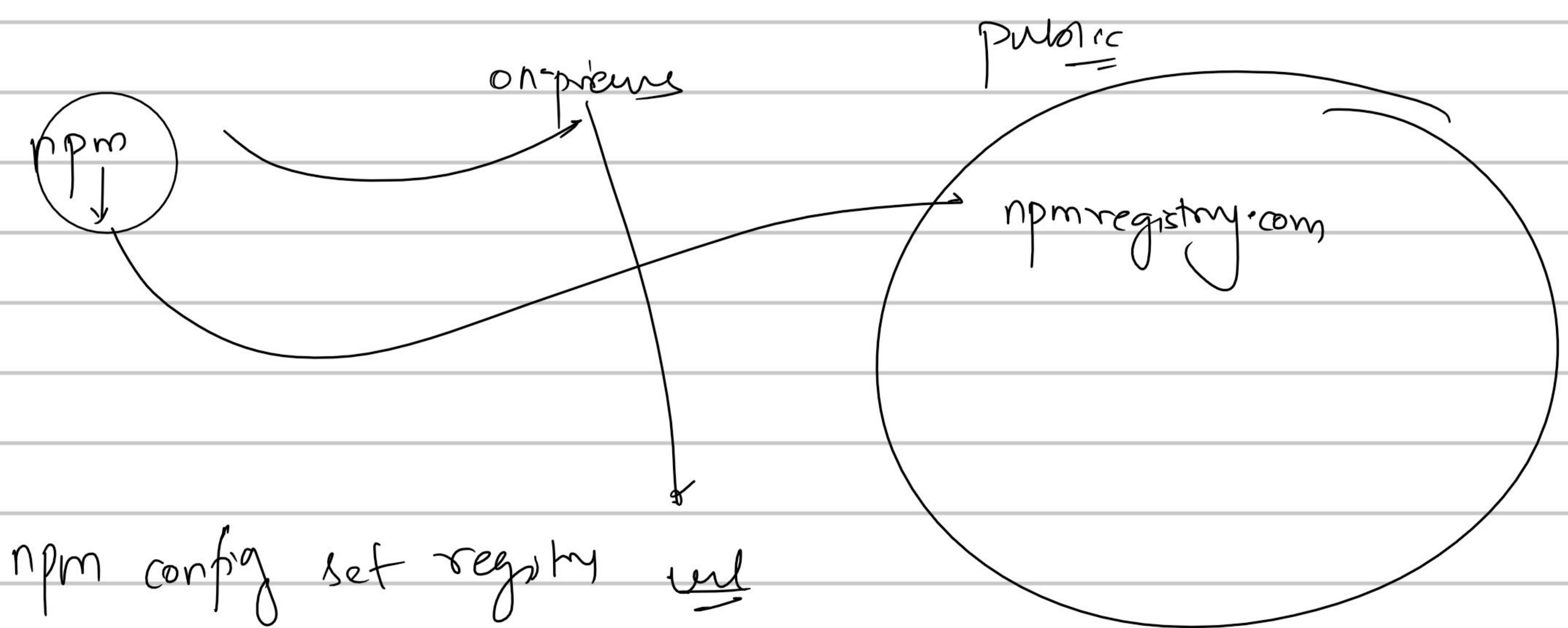
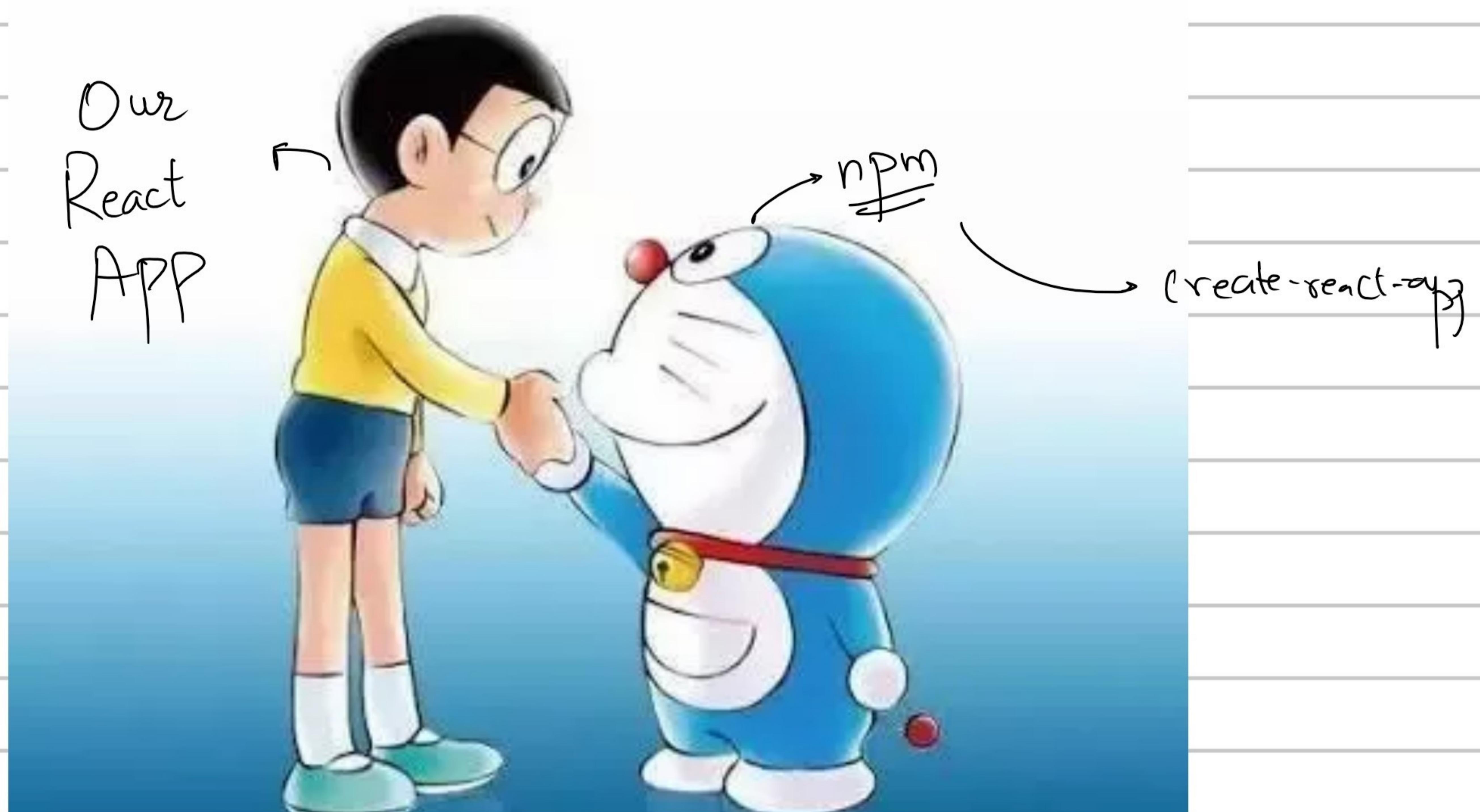
```
import React, {useState, Component} from 'react'; import {..., ...} from '...'
```

Named Export → import {..., ...} from '...'

Default Export → import ... from '...'

Setting Up React Environment

```
npm i -g create-react-app
```



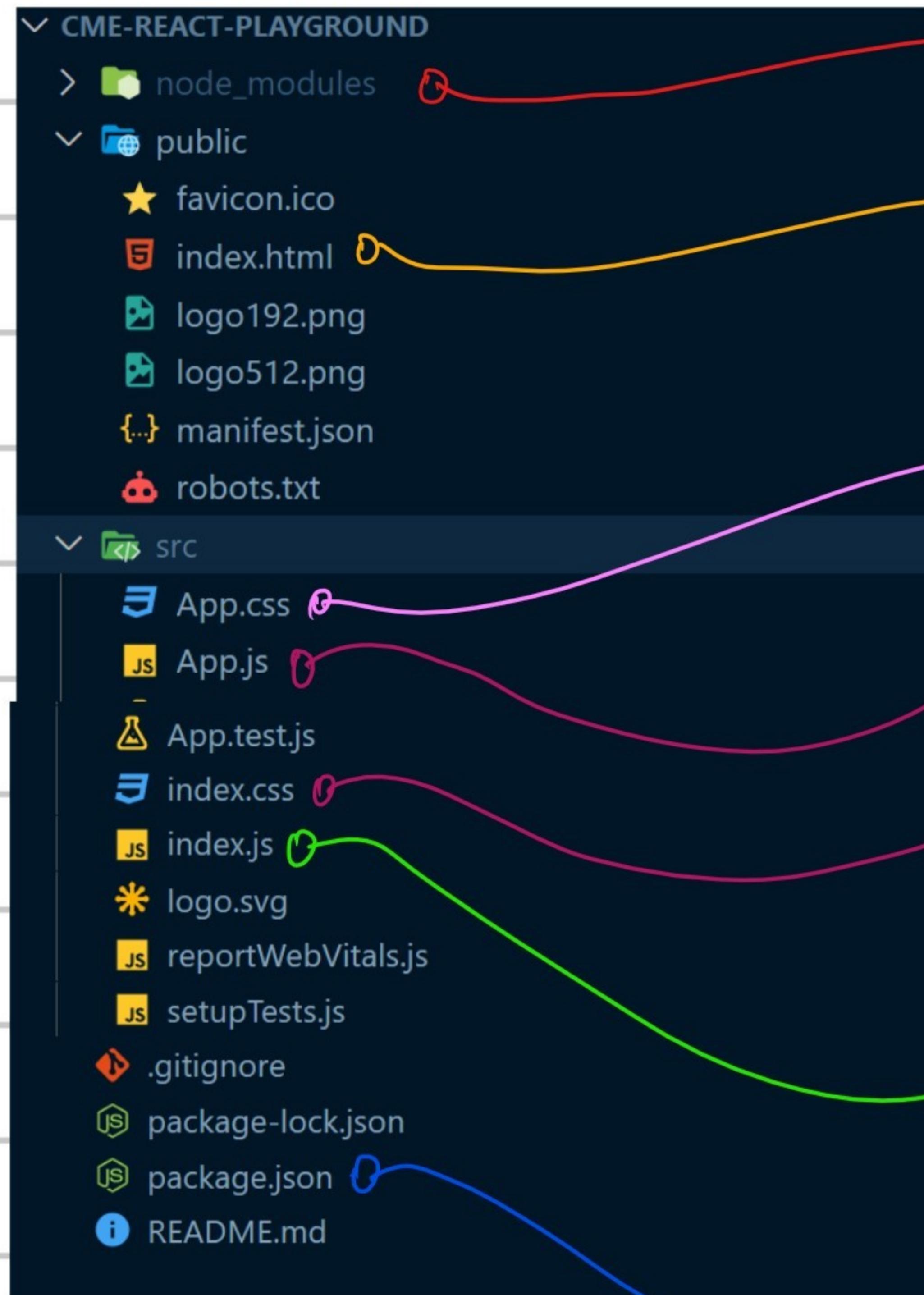
To create a React Application

```
npx create-react-app app-name
```

To start the app

```
cd app-name  
npm start
```

Structure of React App



contain all the required & imported library packages.

renders the data given by index.js

Styles App.js

Root Component → React Component

Styles index.html

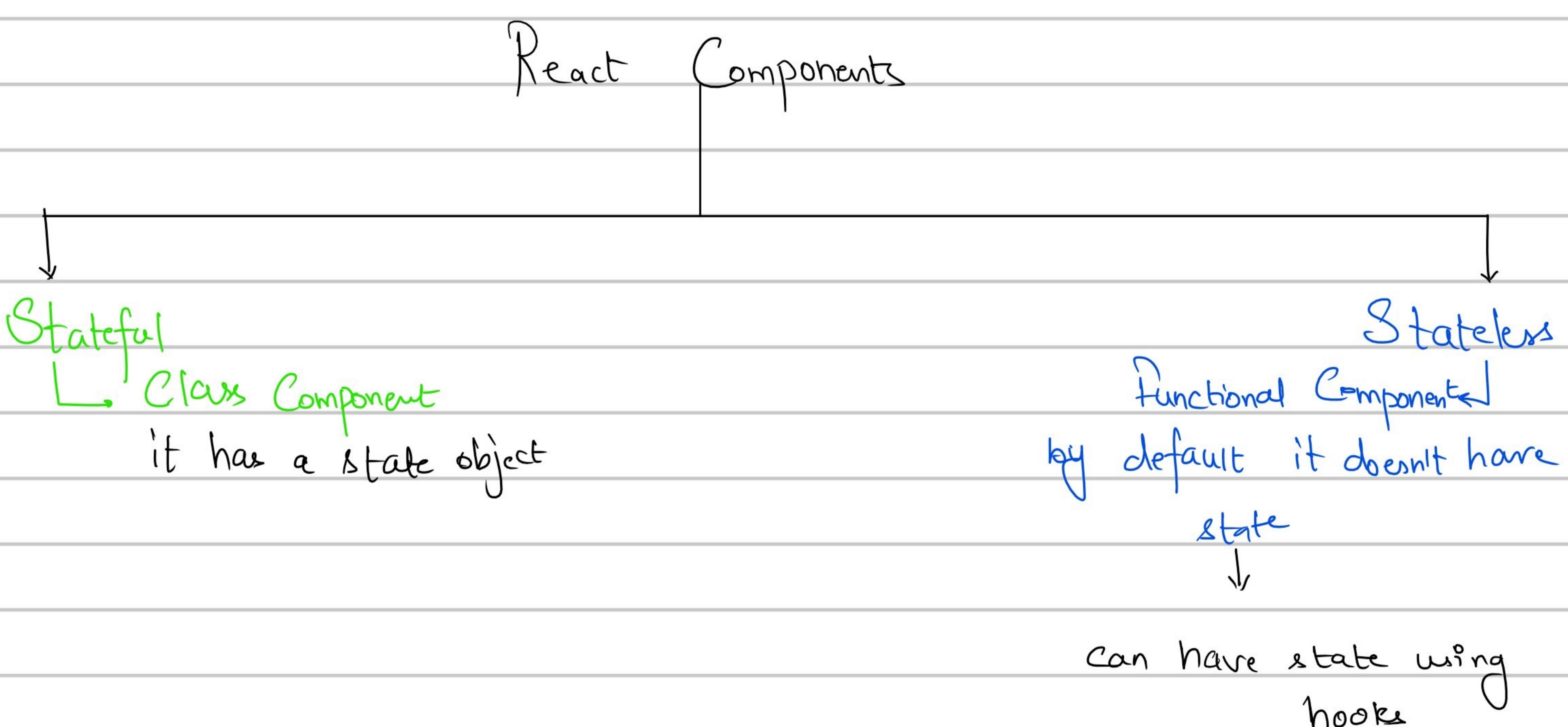
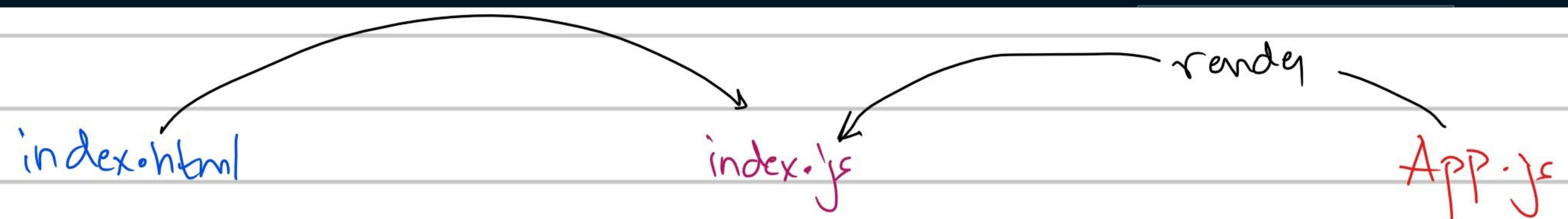
Main JS file which loads the Root Component

Configuration Files

flow

The screenshot shows a code editor with three tabs open:

- index.html**: Contains basic HTML structure including a title, body, and a note about webfonts.
- index.js**: Contains JavaScript code for rendering a React component. It imports React, ReactDOM, and App from './App'. It creates a root element using ReactDOM.createRoot and renders the App component within a StrictMode.
- App.js**: Contains a functional component named App that returns a header with a logo, a title, and a link to 'Learn React'.



React uses JSX for implementation

Syntax extension
to
JavaScript

JSX → JavaScript XML

Browser → JS

const element = <h1> Hello {name} </h1>
JS → HTML

```
let name = 'Zartab'  
const el = <h1> Hello {name} </h1>
```

React → Declarative

<button onClick={displayName} />

displayName() {

JS → Imperative

hide the complexity

```
let btn = document.getElementById('btn');
```

}

```
btn.addEventListener('click', displayName)
```

Basic React Component

```
class Tweet extends Component {
```

state = {

→ State goes here

}

render() {

→ UI Implementation

→ returns a React

Element

}

}

render()

class SomeComponent extends Component {

 render() {

 return (

 <div>

 <h1>HW</h1>

 <h2> HW</h2> ✘

 </div>)

→ JSX → return a single component

* DOM *

Dom → Document Object Model

- represents the UI of the application
- Dom is a tree like data structure

→ Everytime application state is updated, Dom is updated.

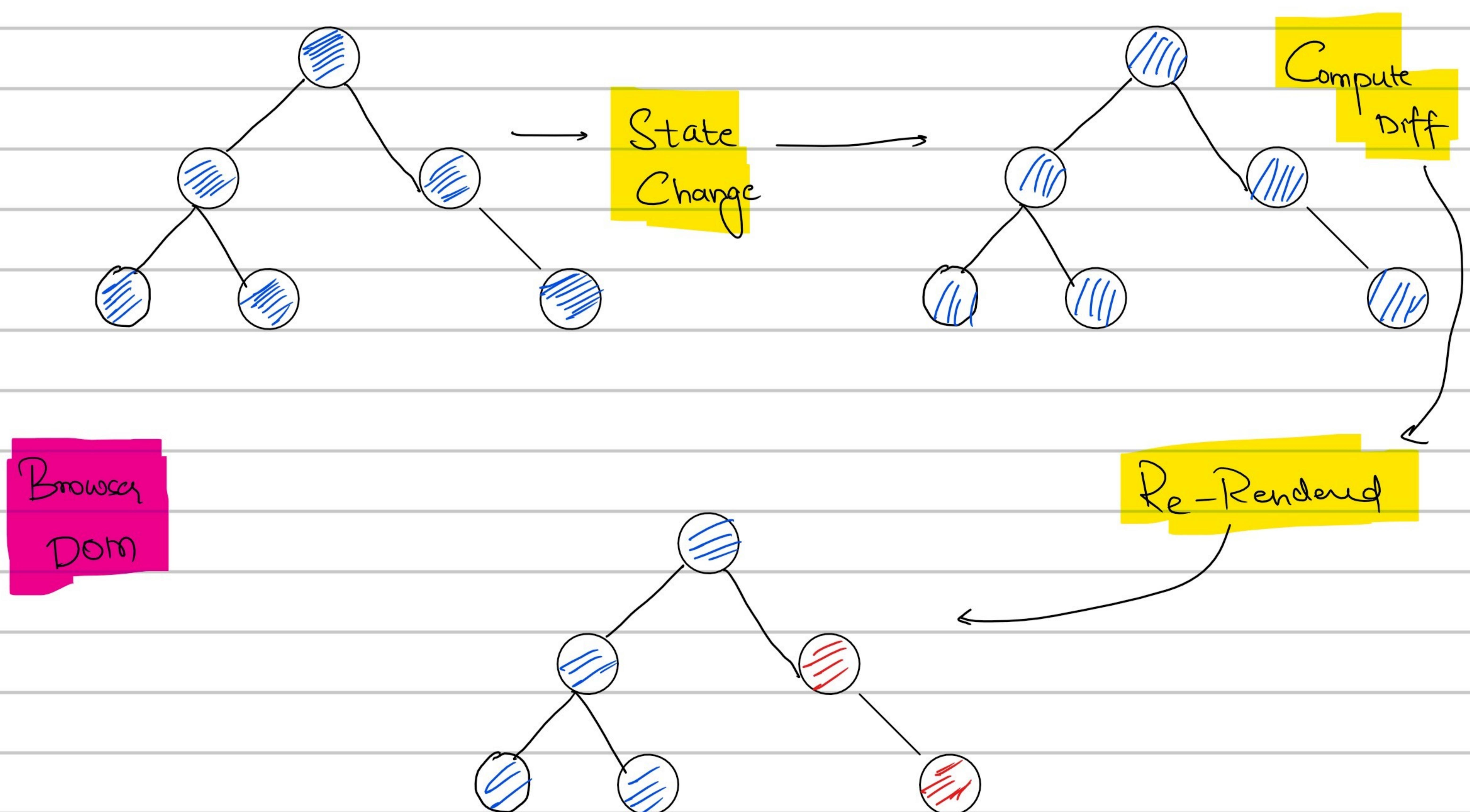
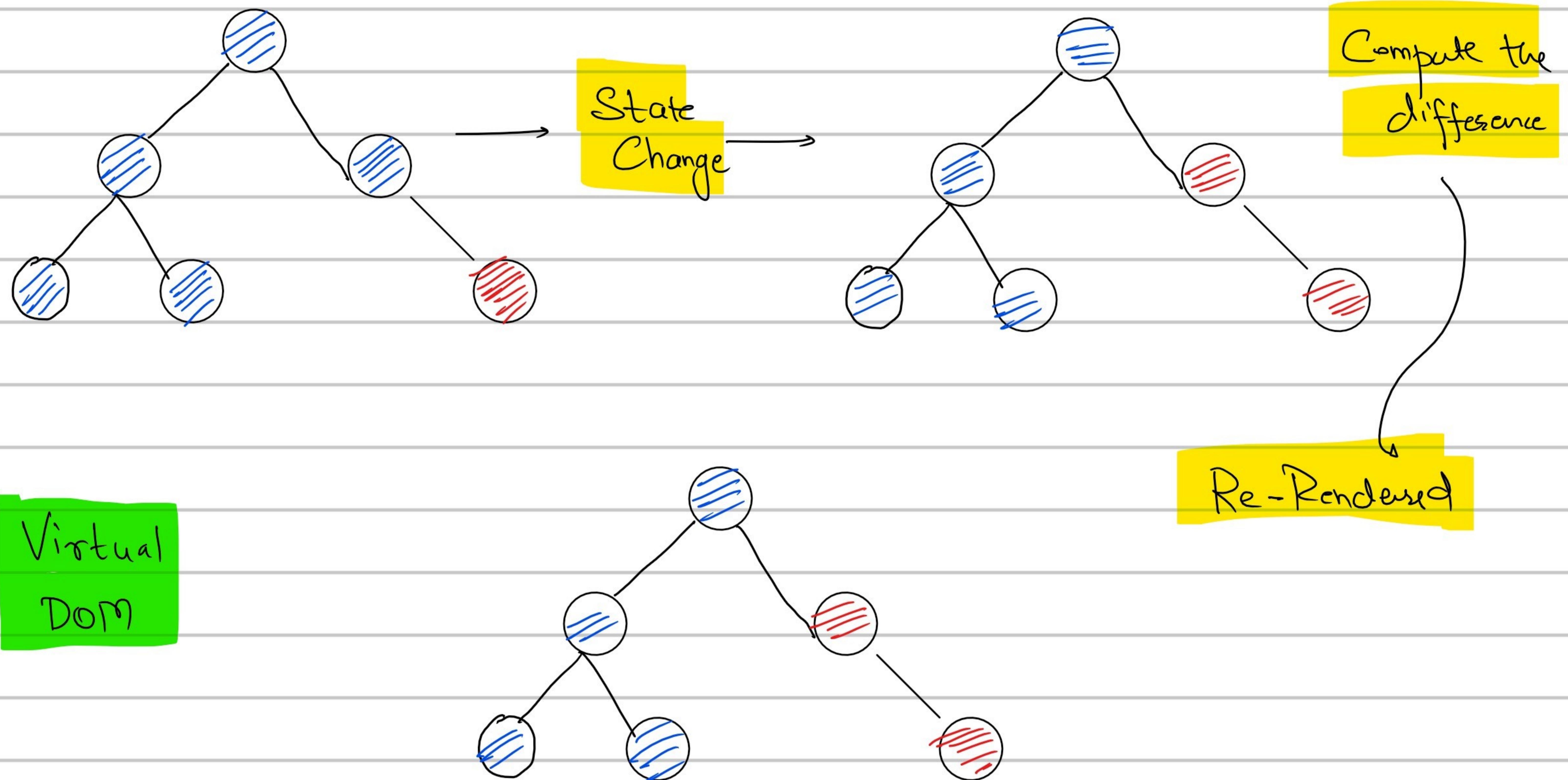
re-rendering of page & its child components are ~~so~~ time-consuming op

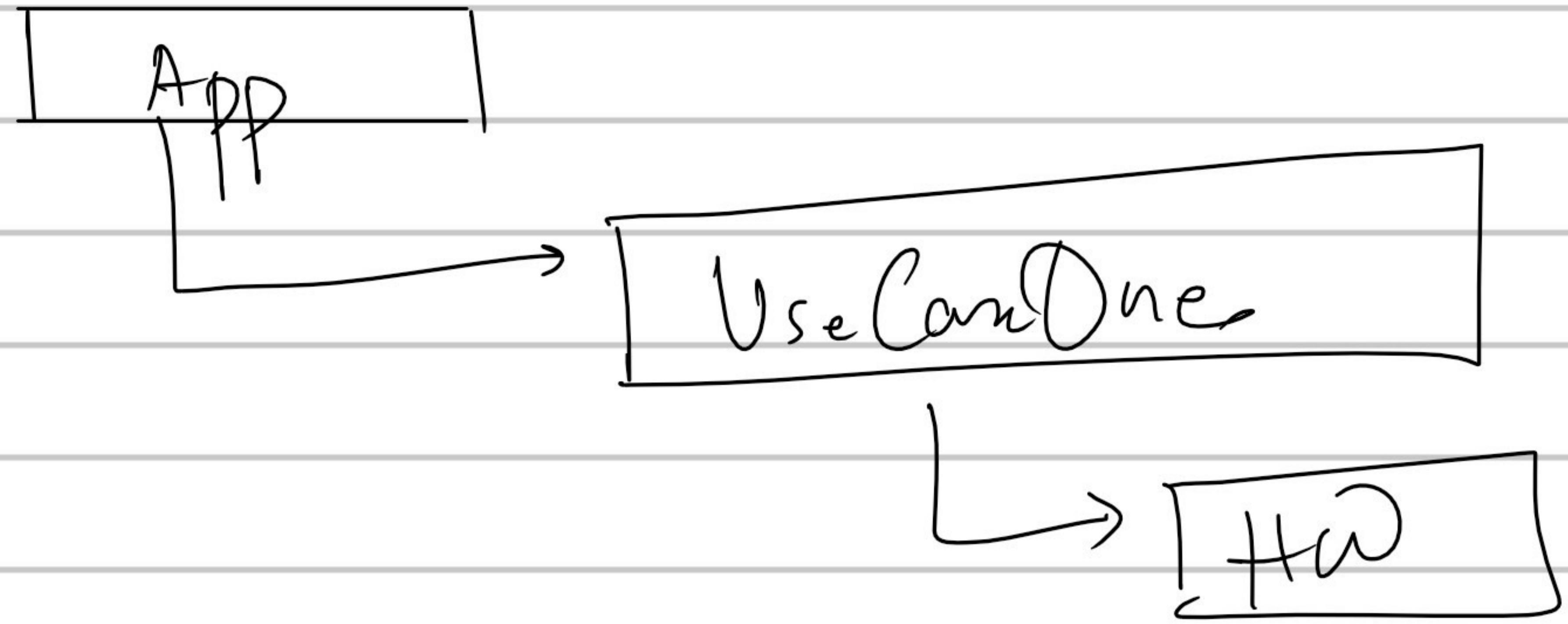
Virtual Dom

- * When new elements are added to the UI, a virtual dom which is represented as a tree is created.
- * Each element is a node in the ~~tree~~ tree. If state of this element changes, new virtual ~~tree~~ dom tree is created

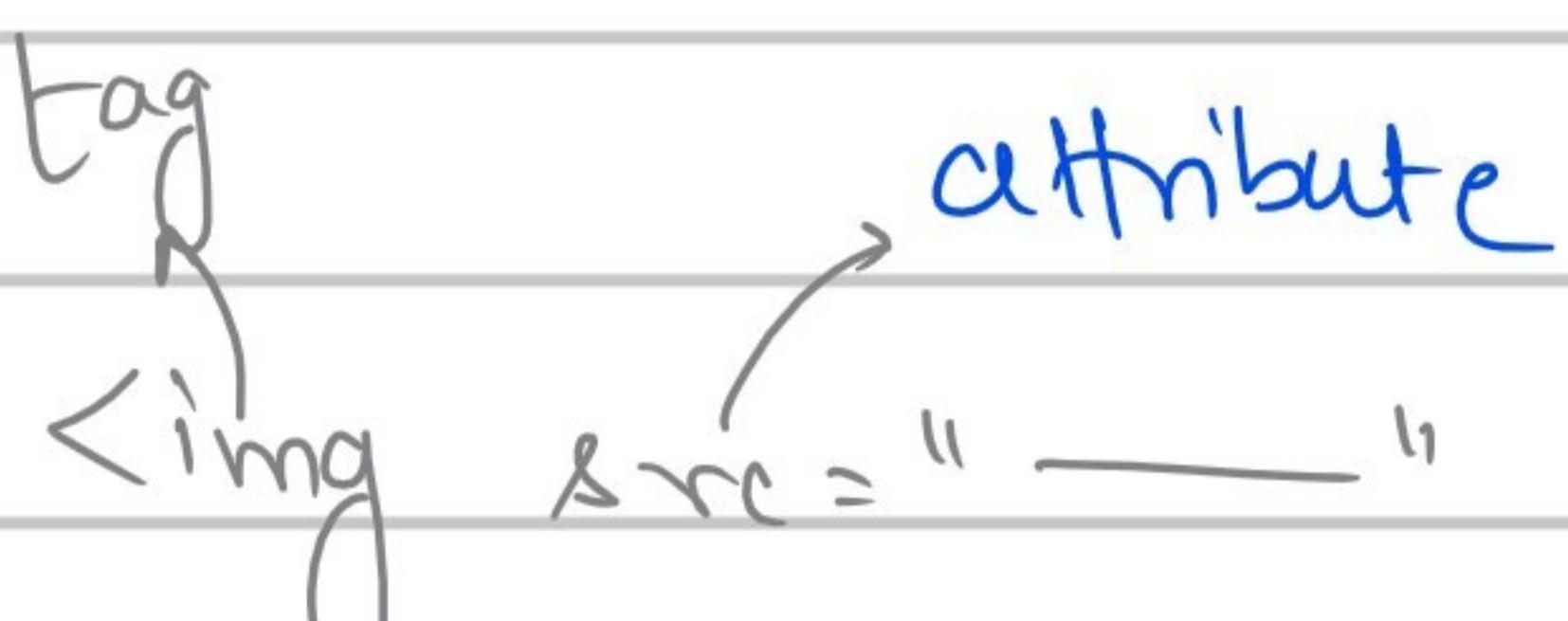
The tree is compared with previous virtual dom tree

Difffed / Difffing





* Props *

- Normal HTML tag have attribute  ``
 - React Components have props
- `<NameComponent name="Zantab" />`
- prop → equivalent to attribute
 there can be any number of props &
 they can be of any name

| | |
|--|---|
| <code><NameComponent name="Zantab" /></code> | Props are captured ① Class Components → available in an object called as props |
| | ② Function Components → available as an argument |

* State *

State is a plain JS object used by React to represent the information about a component's current situation.

class SomeComponent extends Component {

 state = {

³
 ^y

 render() {

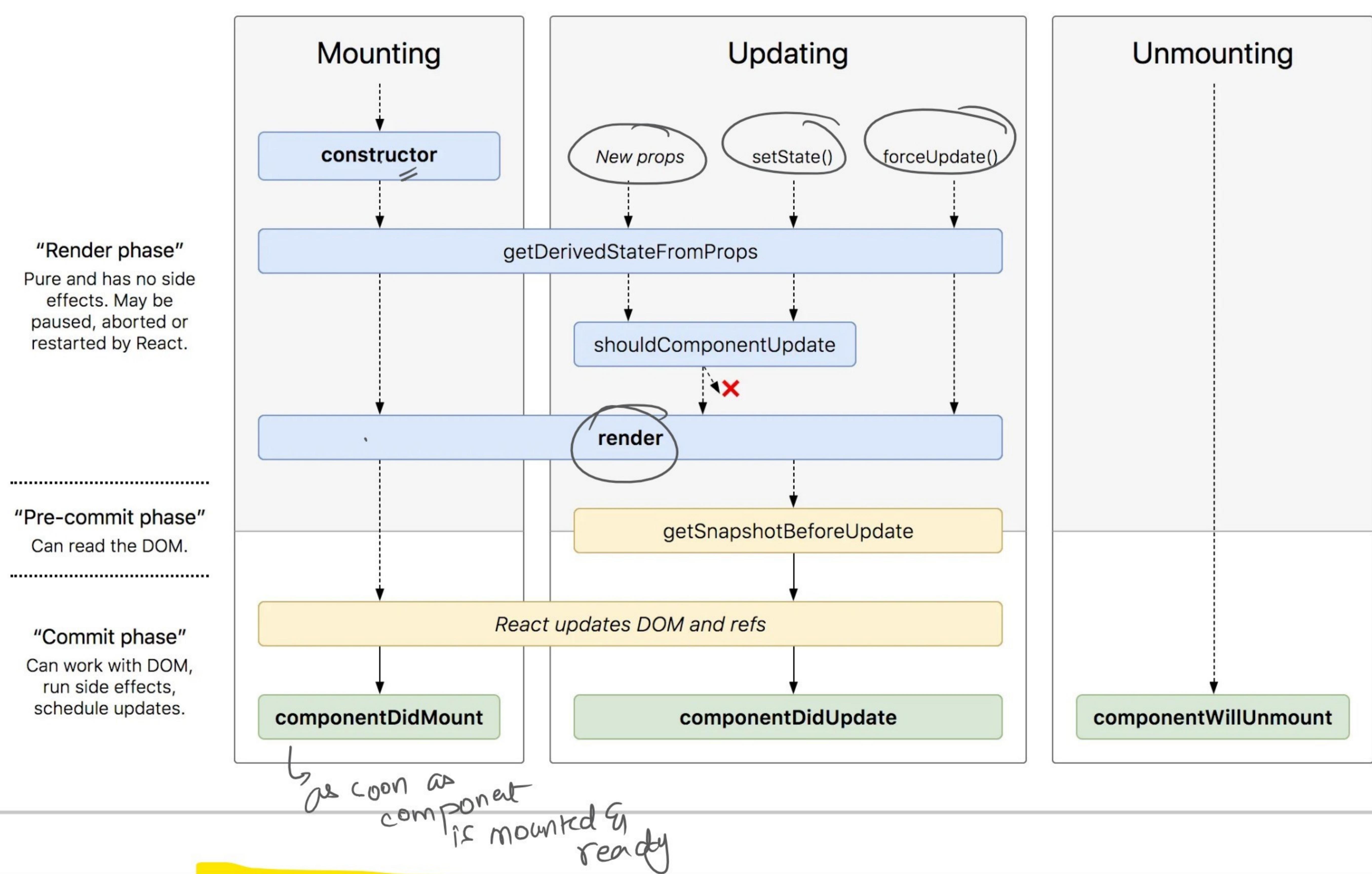
³
 ^y

State is like props,
but it is private &
fully controlled by the
component

When we need to change the state of a React class Component.

this.setState({ key: value })

React version 16.4 Language en-US



Mounting → Birth of Component

Update → Growth of Component

Unmount → Death of Component

* Hooks & State *

Special type of function which allows us to break big class components into smaller components.

In previous React versions, if a component needed state, it has to use a class component

useState → hook

import {useState} from 'react'

const [variable, setVariable] = useState(defaultValue);

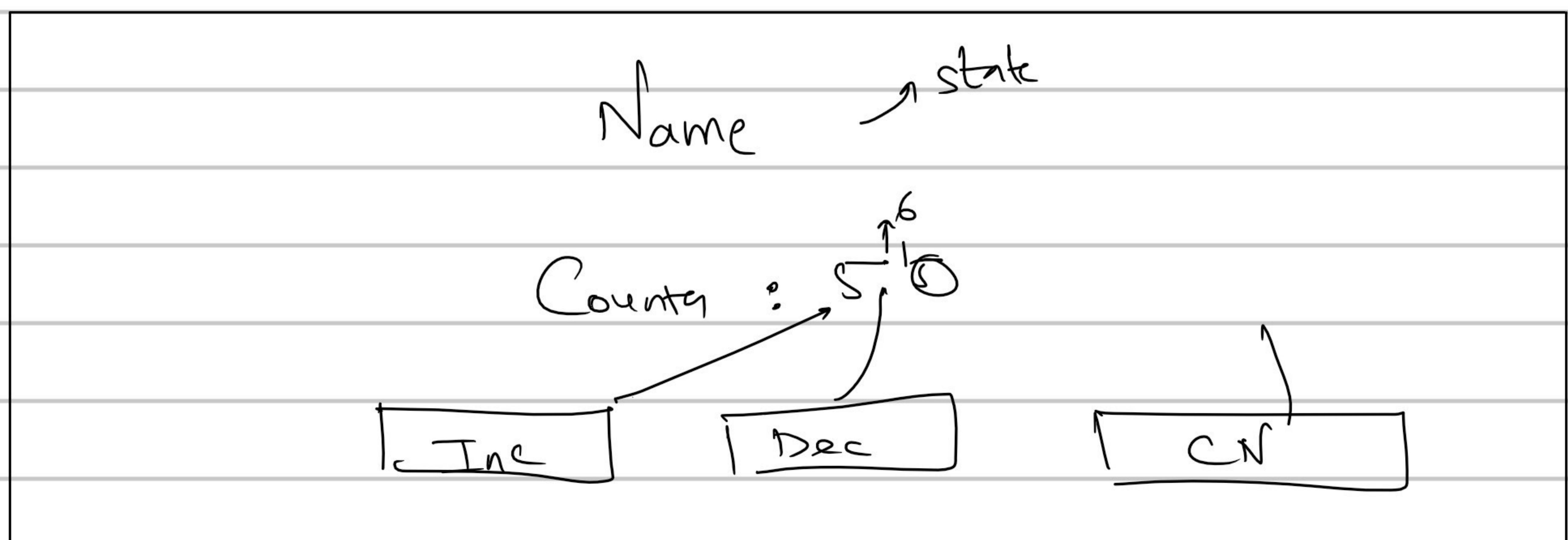
state function to update the state hook for creating state in function component default value for state variable

const [productCount, setProductCount] = useState(0);

const [fruit, setFruit] = useState('Apple');

const [colors, setColors] = useState([])

const [address, setAddress] = useState({country: 'India'});

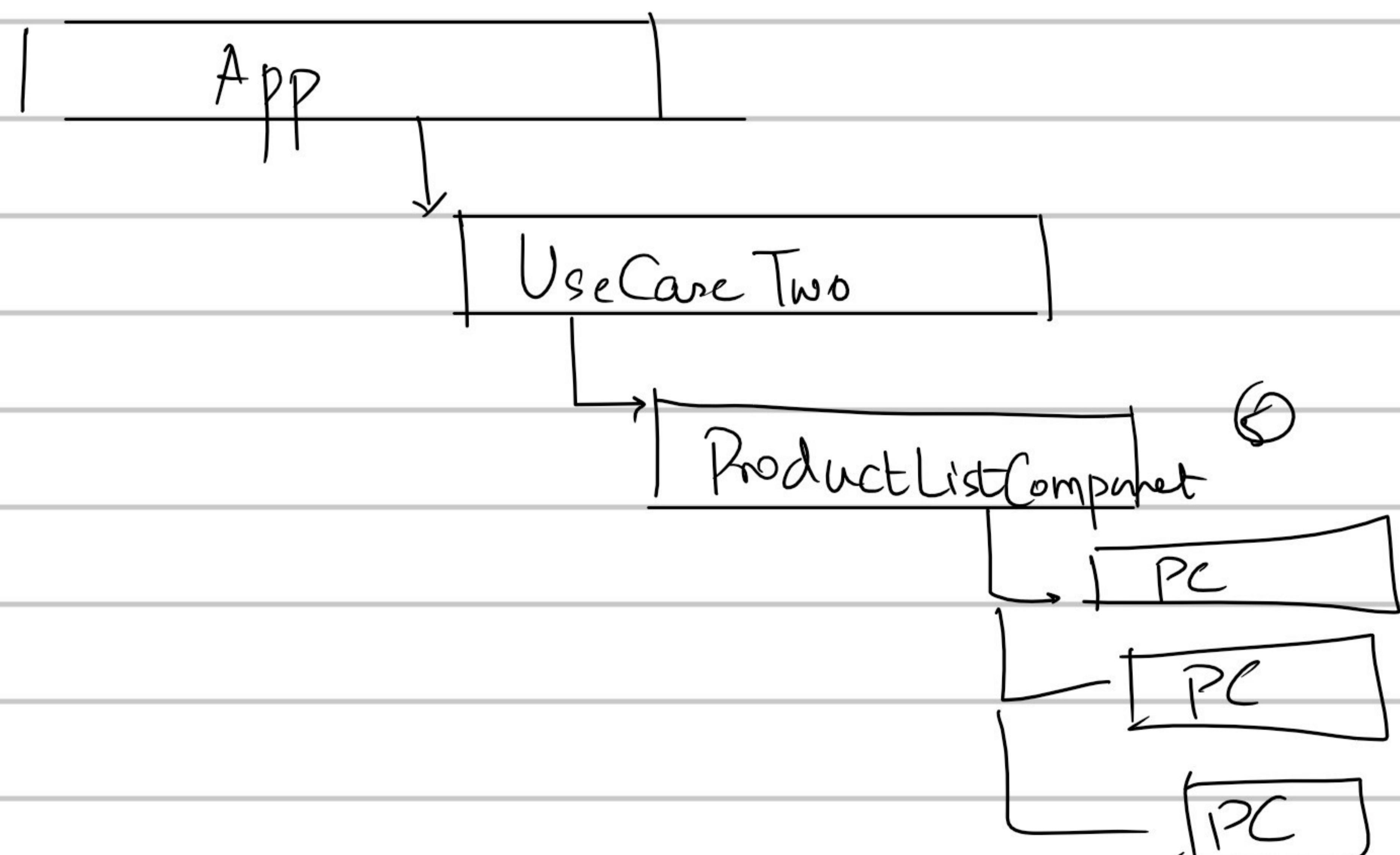


Whenever project need some dependency

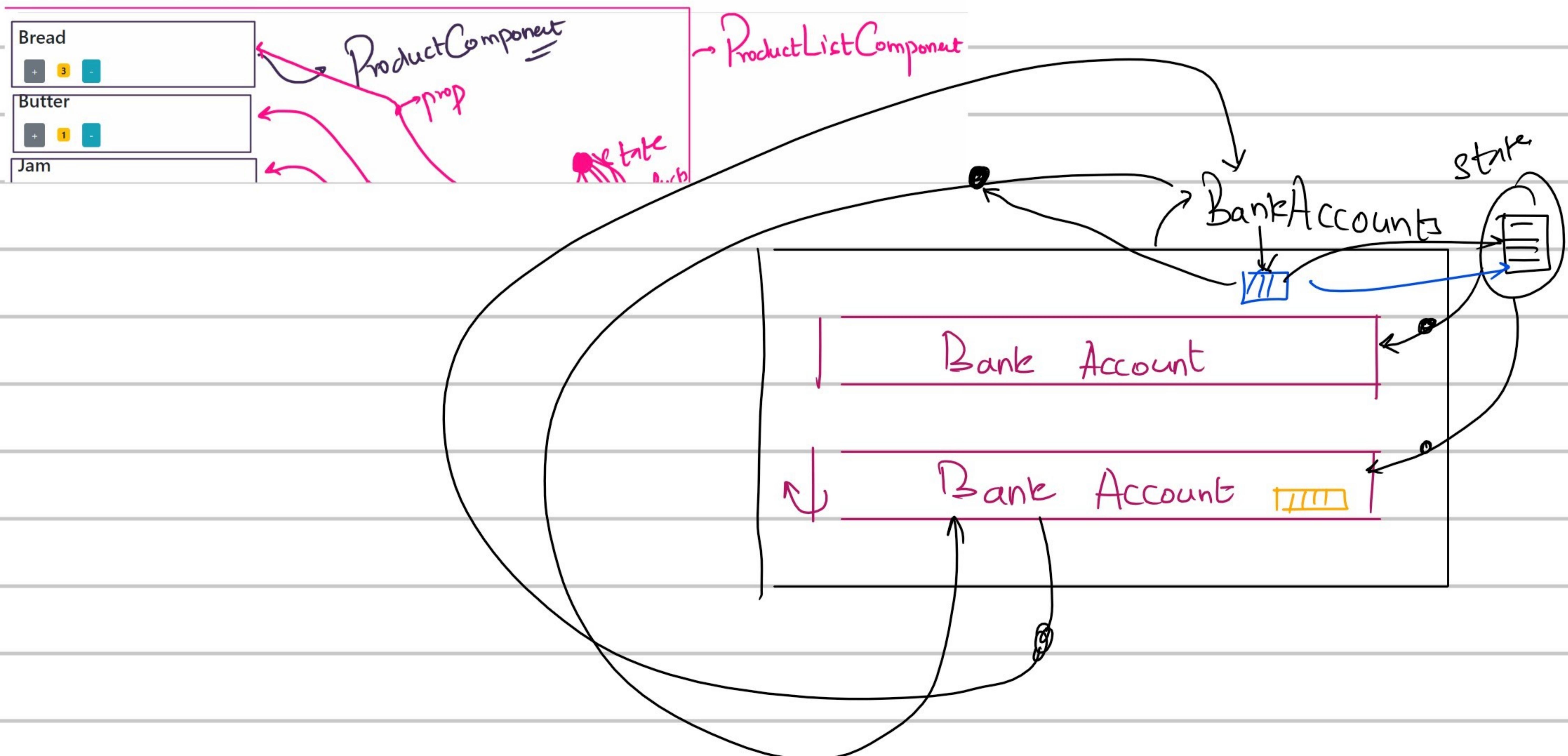
Go inside the project folder

npm i dependency@ version

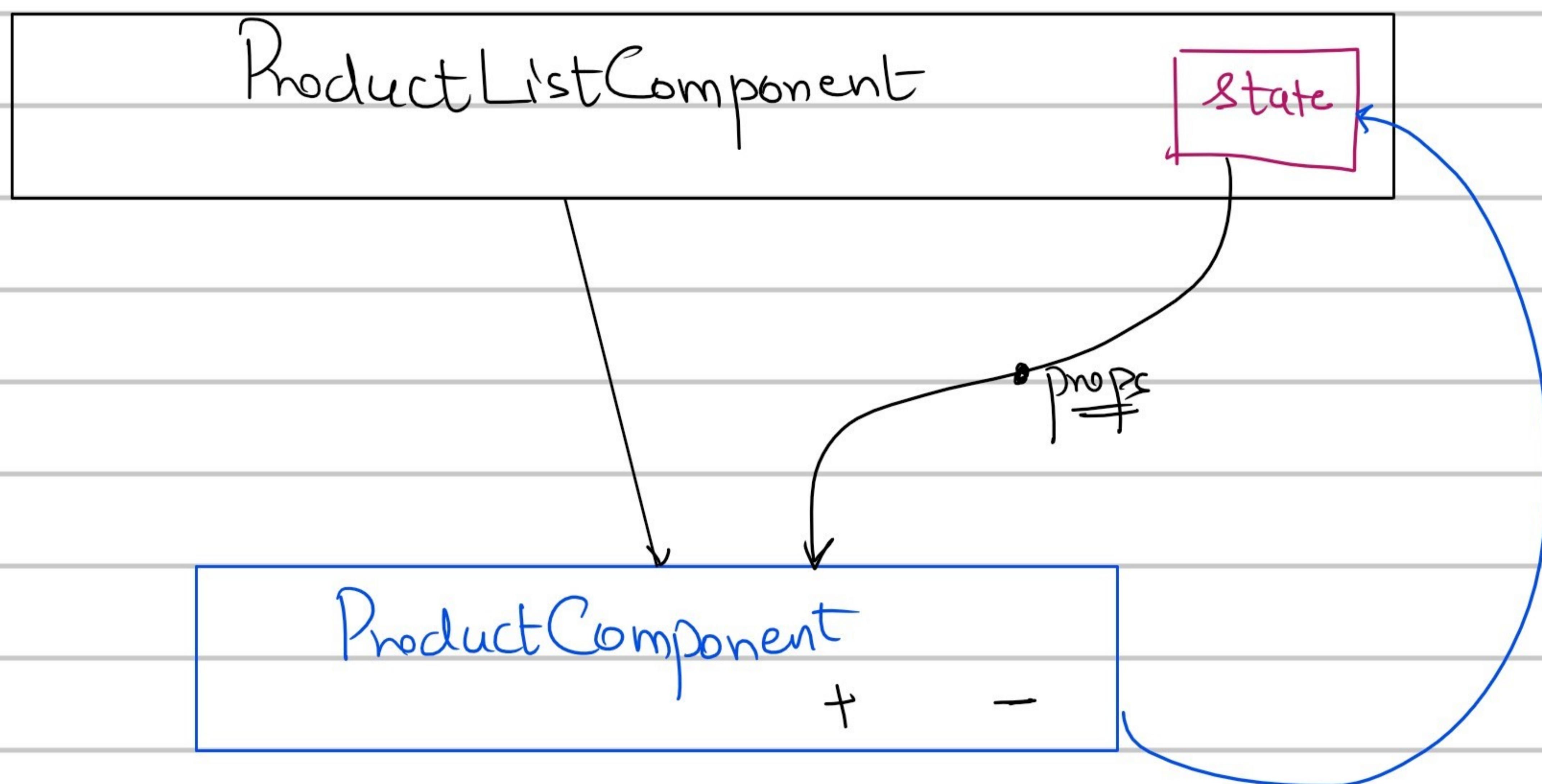
npm i bootstrap@4.1.1



Hello CME React Mumbai



State should be modified
by the component to
which it belongs !!



- * PLC is responsible for updating the state
- * But the action to update the state will come from PC
- * PLC will be required to pass a reference of a function to PC via props; which PC can invoke when it requires to make the changes

<button onClick = {somefunction() } >

{ somefunction }

() → Execute straight away

→ Execute when event occur
clicked

<button onClick = {someOtherFunc(x) } >

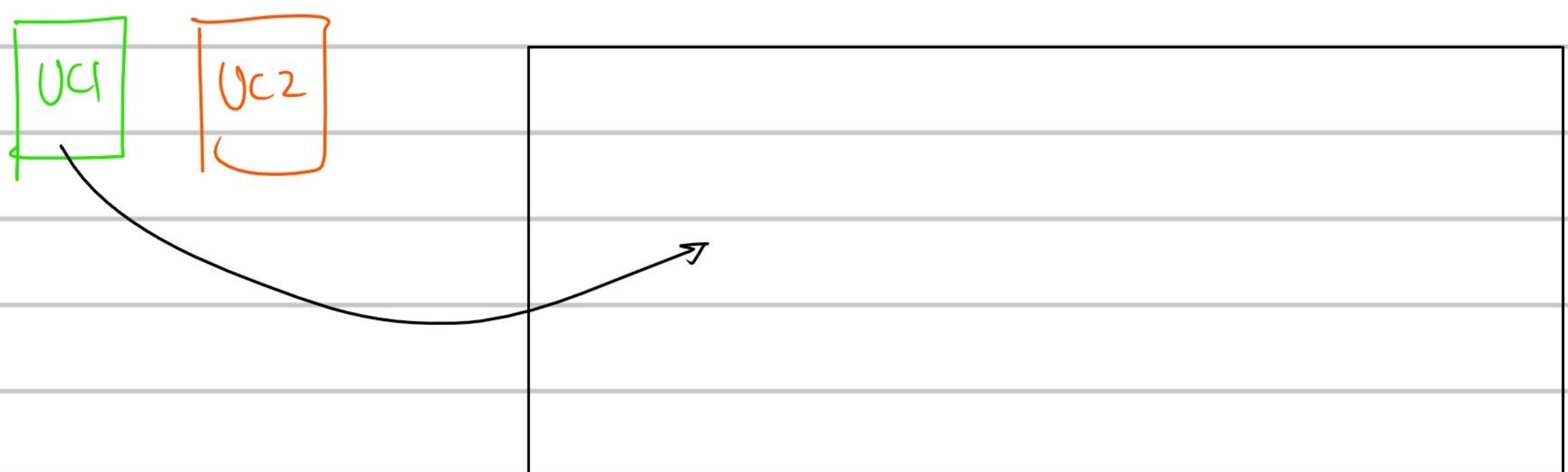
<button onClick = {() => someOtherFunc(x) } >

<button onClick = { test } >

function test() {
 someOtherFunc(x);
}

React Routing

React App → SPA → Single Page Application.



npm i react-router-dom

* BrowserRouter *

hold all the
browser
history

<BrowserRouter>

App

</BrowserRouter>

* Route *

<Route>

Component of react-router-dom

render the component when URL
matches

* Link *

→ declarative, accessible navigation around project.

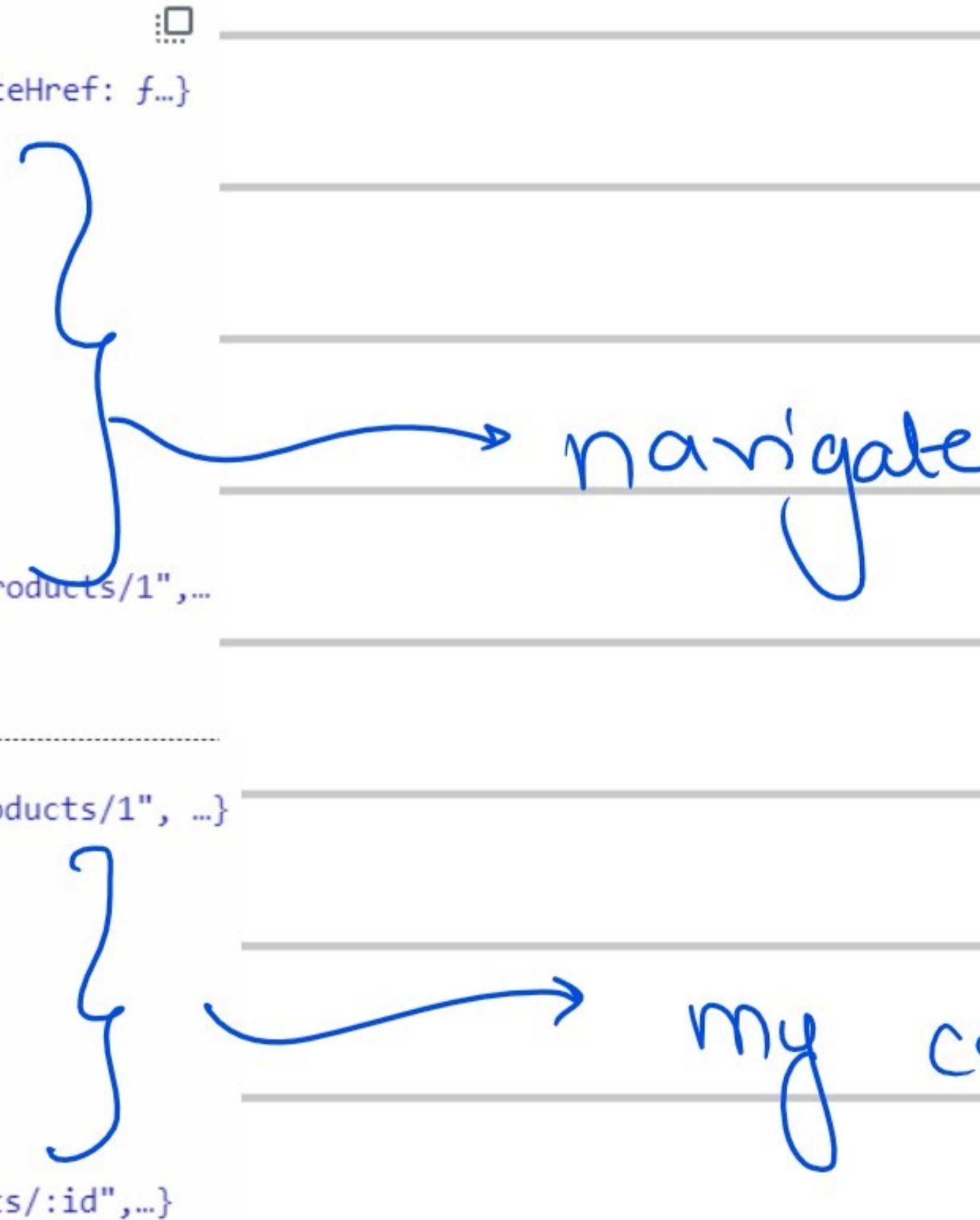
 Products

→ <Link to="/products"> Product </Link>

* Route Parameters *

```
props
  - history: {action: "PUSH", block: f block() {}, createHref: f...
    action: "PUSH"
    block: f block() {}
    createHref: f createHref() {}
    go: f go() {}
    goBack: f goBack() {}
    goForward: f goForward() {}
    length: 50
    listen: f listen() {}
    location: {hash: "", key: "qtfwcb", pathname: "/products/1",...
    /push: f push() {}
    /replace: f replace() {}
    new entry: ""
  - location: {hash: "", key: "qtfwcb", pathname: "/products/1", ...}
    hash: ""
    key: "qtfwcb"
    pathname: "/products/1"
    search: ""
    state: undefined
    new entry: ""
  - match: {isExact: true, params: {}, path: "/products/:id",...}
    isExact: true
    params: {id: "1"}
    path: "/products/:id"
    url: "/products/1"
    new entry: ""
  staticContext: undefined
  new entry: ""
```

When a component
is enclosed <Route>



my component's current location

route parameters

`push` → user can go backward & forward

`replace` → link is set to new → you cannot go back

useEffect Hook

lets you perform side effects
in functional component

Class

Function

componentDidMount()

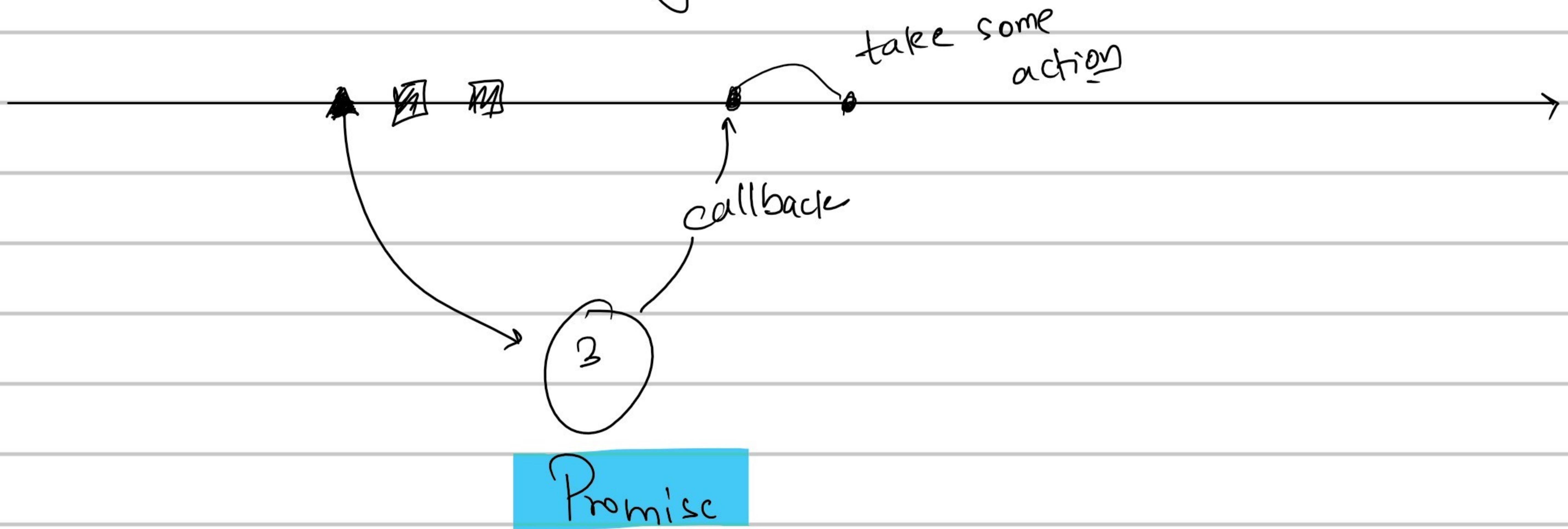
componentDidUpdate()

componentWillUnmount()

useEffect(function)

useEffect (function, dependency)

* Async Operations *



holds the eventual result of an asynchronous operation

Pending

Async Task

Resolved / Fulfilled

Rejected

HTTP API

