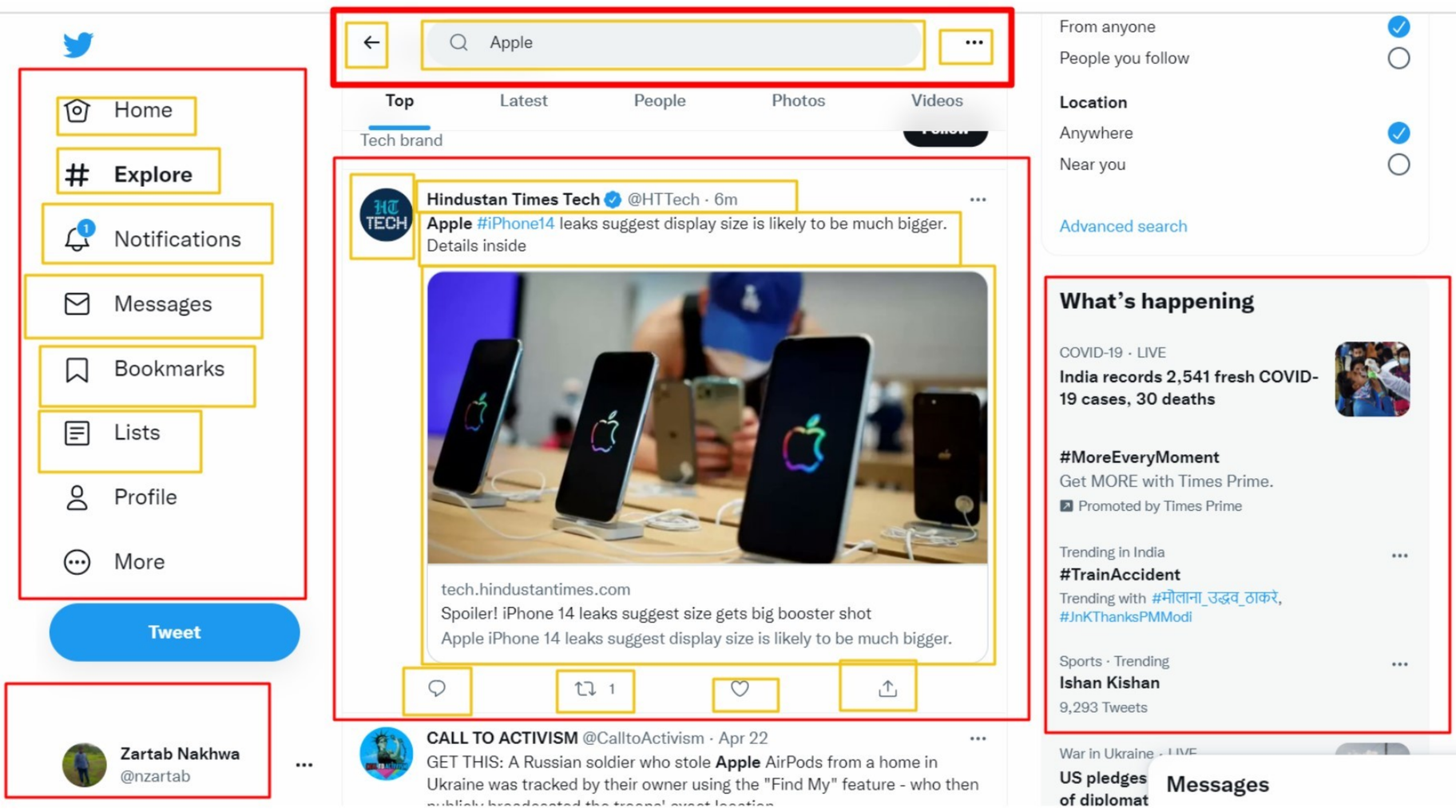
React was developed at Facebook

React → UI Library

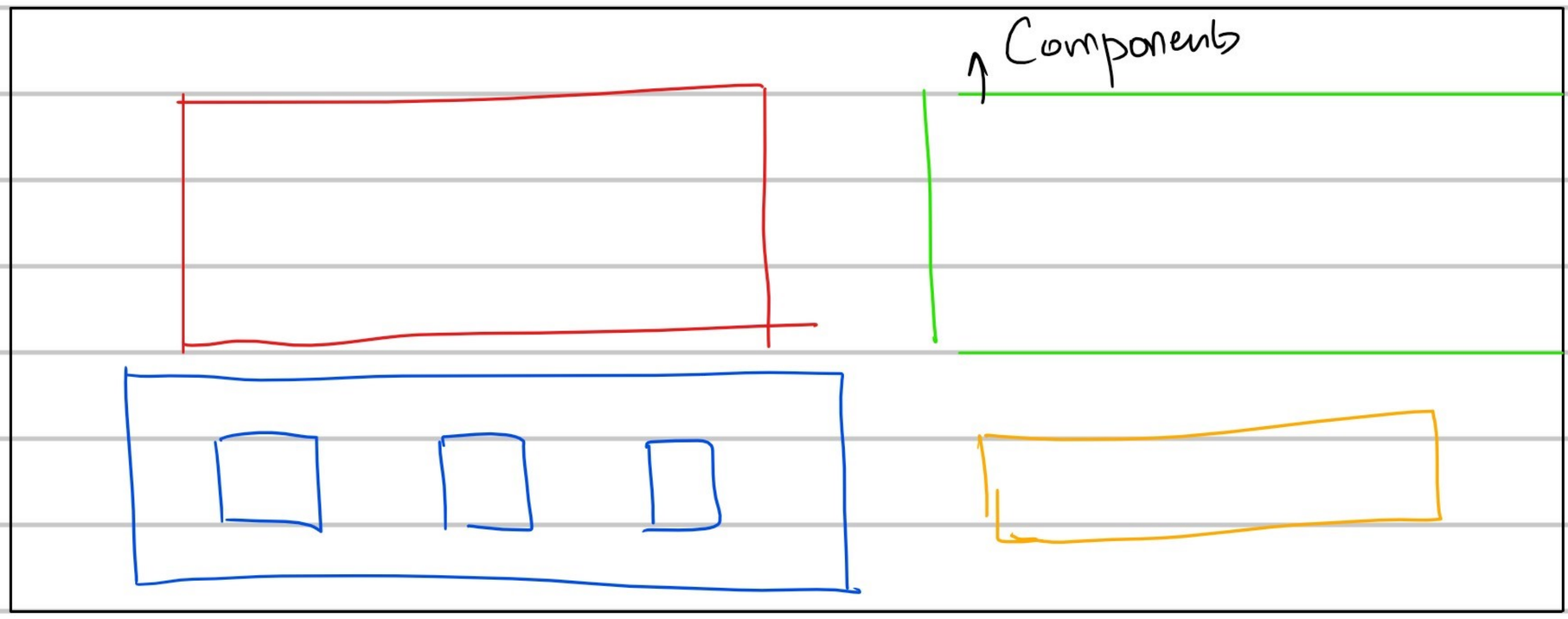
→ Javascript based front end → UI library

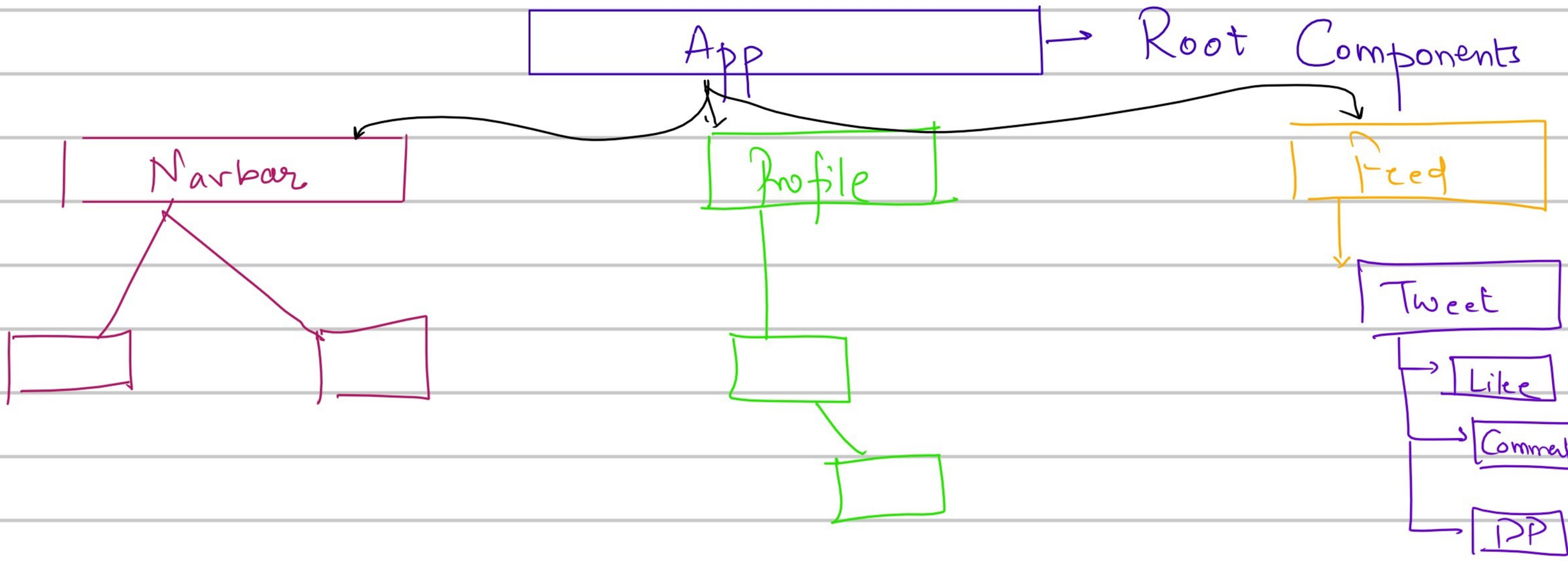
React is made up of Components !!



Component

piece of UI; which has its own | STATE & LIFECYCLE |





SPA

ES-6 Refresher

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

* let v/s var v/s const

var

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

→ update + redeclare

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) => {
                        statement(s)
                    }
```

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

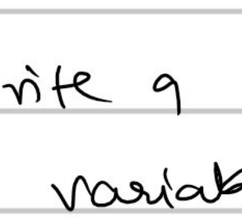
array.filter

{ numbers = [1, 2, 3, 5, 5]

Higher Order
↓
which takes another function as an input

```
function evenfilter(number){
    if number%2 ==0{
        return true.
    }
} else {
    return false;
}
```

array.map → each element from any



resultant away

String Interpolation

when we need to write a string without breaking for variables.

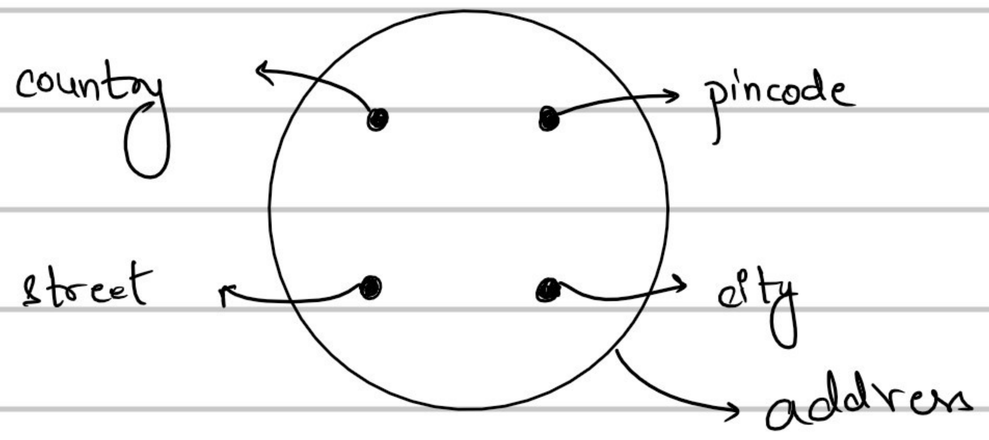
color = "Green"

tag = "
" + color + "
"

tag = `<li> ${color} </li>`

const {street} = address

Object Destructuring



* Spread Operator *

... → object / array.

→ Spreads the elements of source array/object

Named & Default Exports

person.js

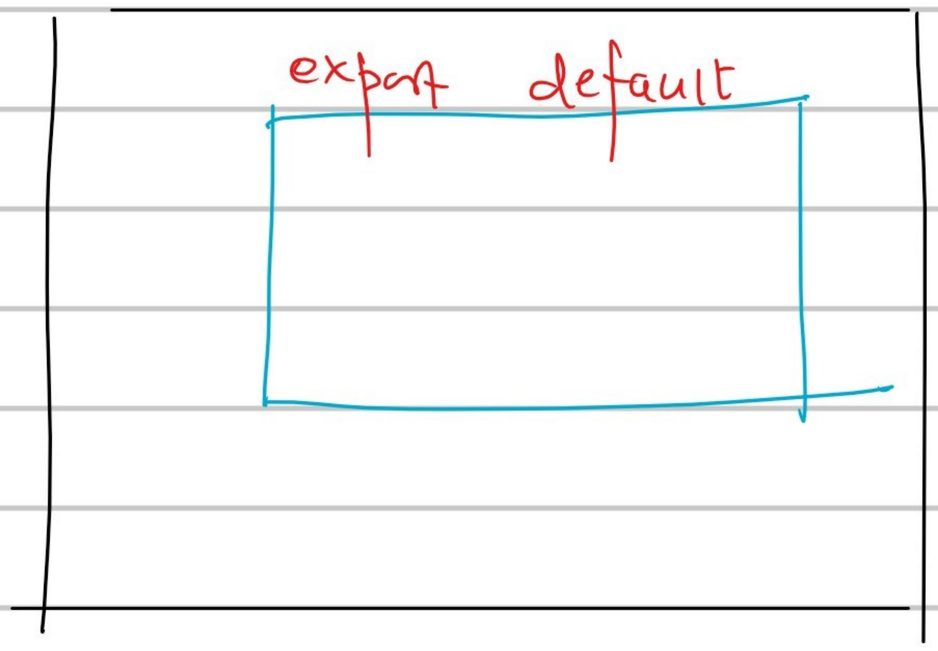
Named Exports

```
export class Person {

}

export function m1() {

}

export function m2() {

}

export let someValue = 10
```

tax.js

Default Export

```
export default taxRate = 18
```



Some-programs.js

export class

export function

export variable

training.js

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

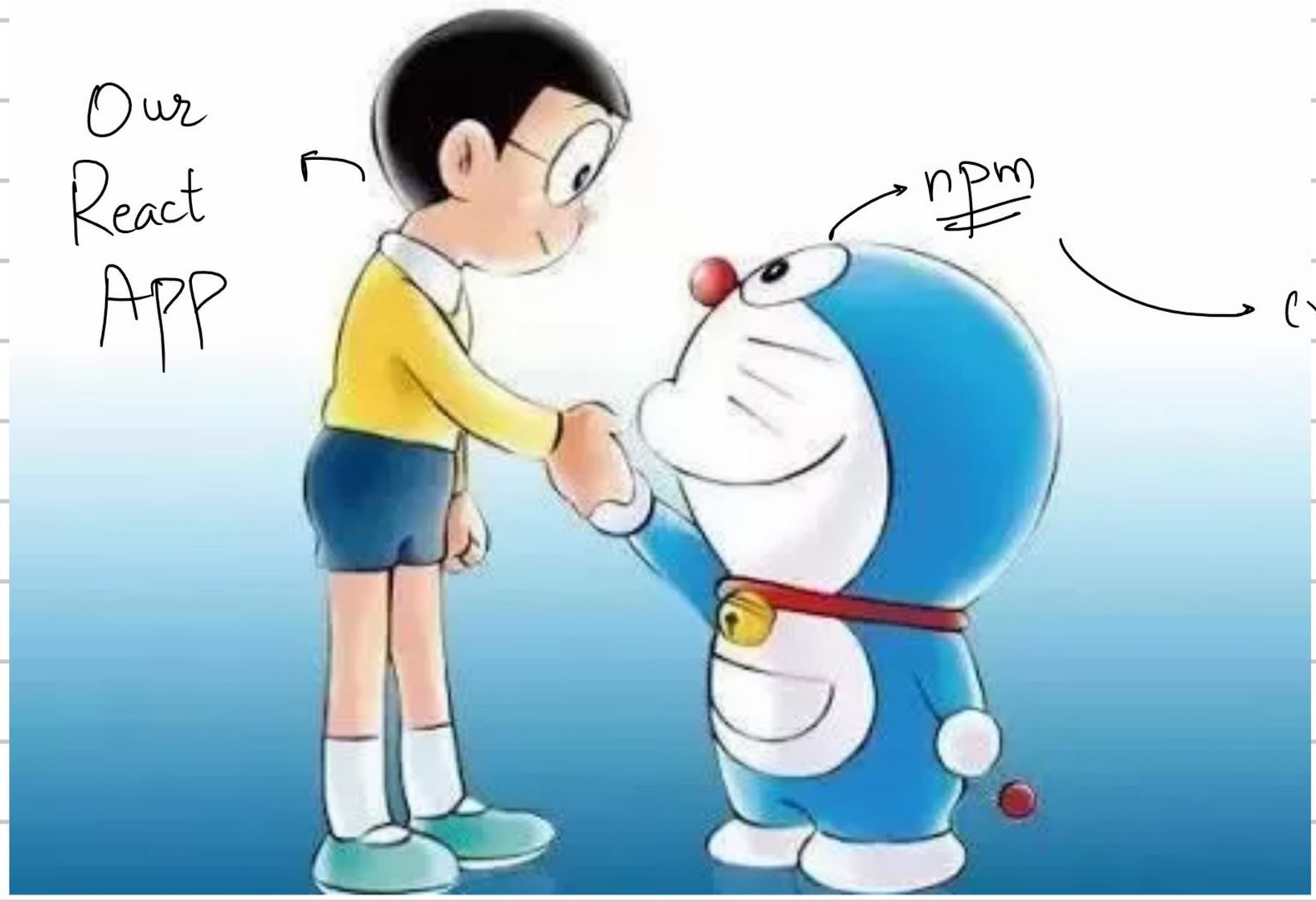
```
import React, {useState, Component} from 'react';
```

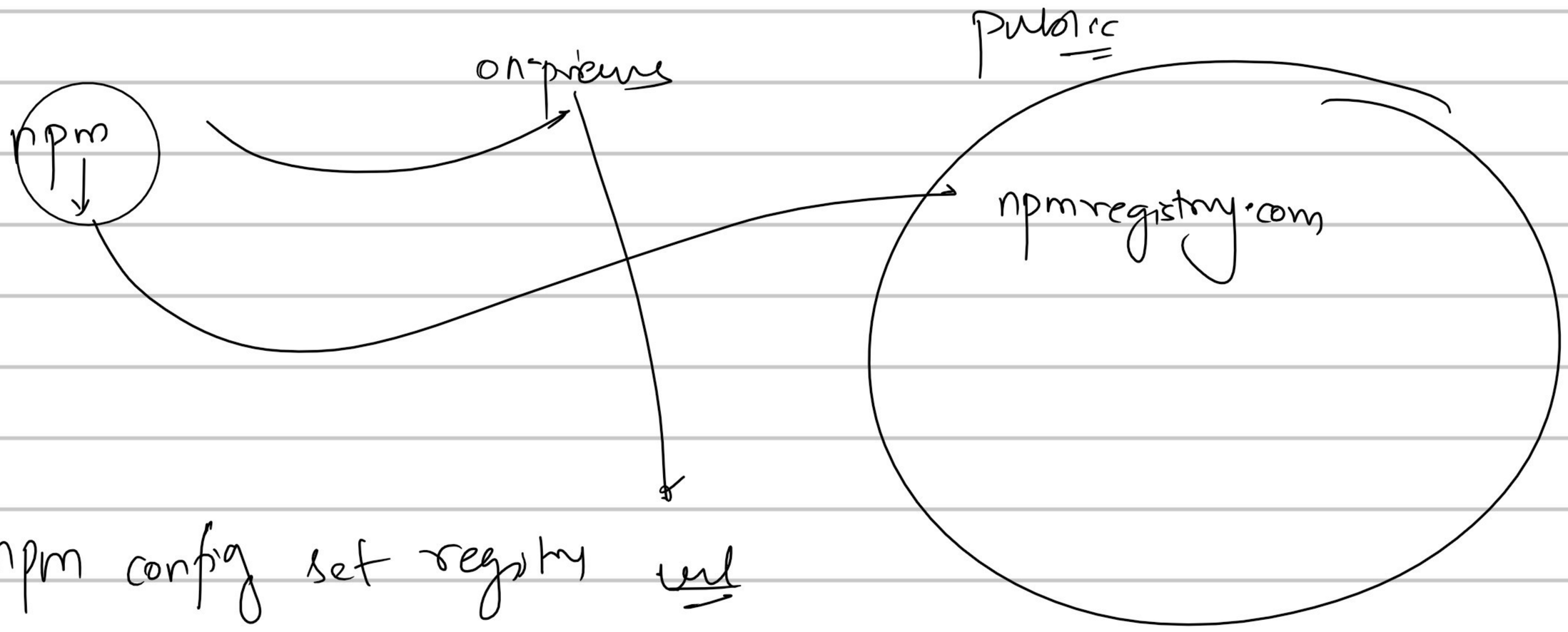
Named Export → import {—, —} from '—'

Default Export → import — from '—'

Setting Up React Environment

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





npm config set registry url

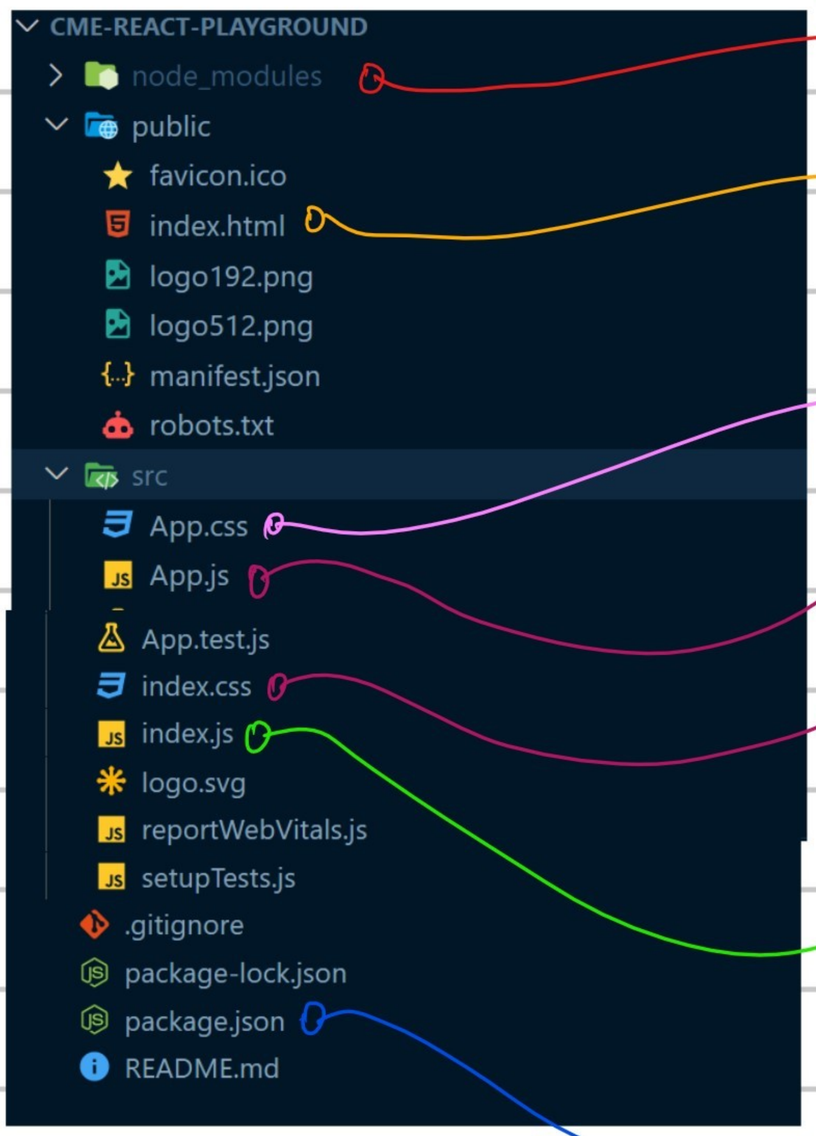
To create a React Application

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

To start the app

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

Structure of React App



contains 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 Componet

Configuration files

Flow

index.html | index.js M | index.js M | App.js | App.js

public > index.html > ...

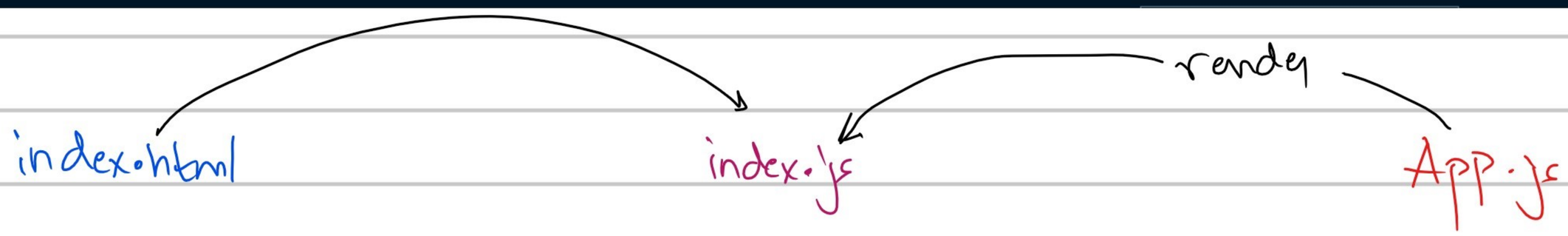
```
    Learn how to configure c
  -->
  <title>React App</title>
</head>
<body>
  <noscript>You need to enab
  <div id="root"></div>
  <!--
    This HTML file is a temp
    If you open it directly

    You can add webfonts, me
    The build step will plac

    To begin the development
    To create a production b
  -->
</body>
</html>
```

src > index.js > root

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVit

const root = ReactDOM.
createRoot(document.getElementById('root'))
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performan
// to log results (for example: reportWebVi
// or send to an analytics endpoint. Learn
reportWebVitals();
```

src > App.js > ...

```
import logo from '
import './App.css'

function App() {
  return (
    <div className
      <header clas
        <img src={
        <p>
          Edit <co
        </p>
        <a
          classNam
          href="ht
          target="
          rel="noo
        >
          Learn Re
        </a>
      </header>
```



React Components

- Stateful
  - Class Component
  - it has a state object
- Stateless
  - Functional Component
  - by default it doesn't have state
  - ↓
  - can have state using hooks

React uses JSX for implementation

Syntax extension to JavaScript

JSX → JavaScript XML

Browser → JS

```
const element = <h1> Hello CME </h1>
```

const element → JS

<h1> Hello CME </h1> → HTML

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

React → Declarative

```
<button onClick={displayName} />
```

JS → Imperative

Hide the complexity

```
displayName(){
}
```

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

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

Basic React Component

```
class Twcet 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 component are time-consuming op

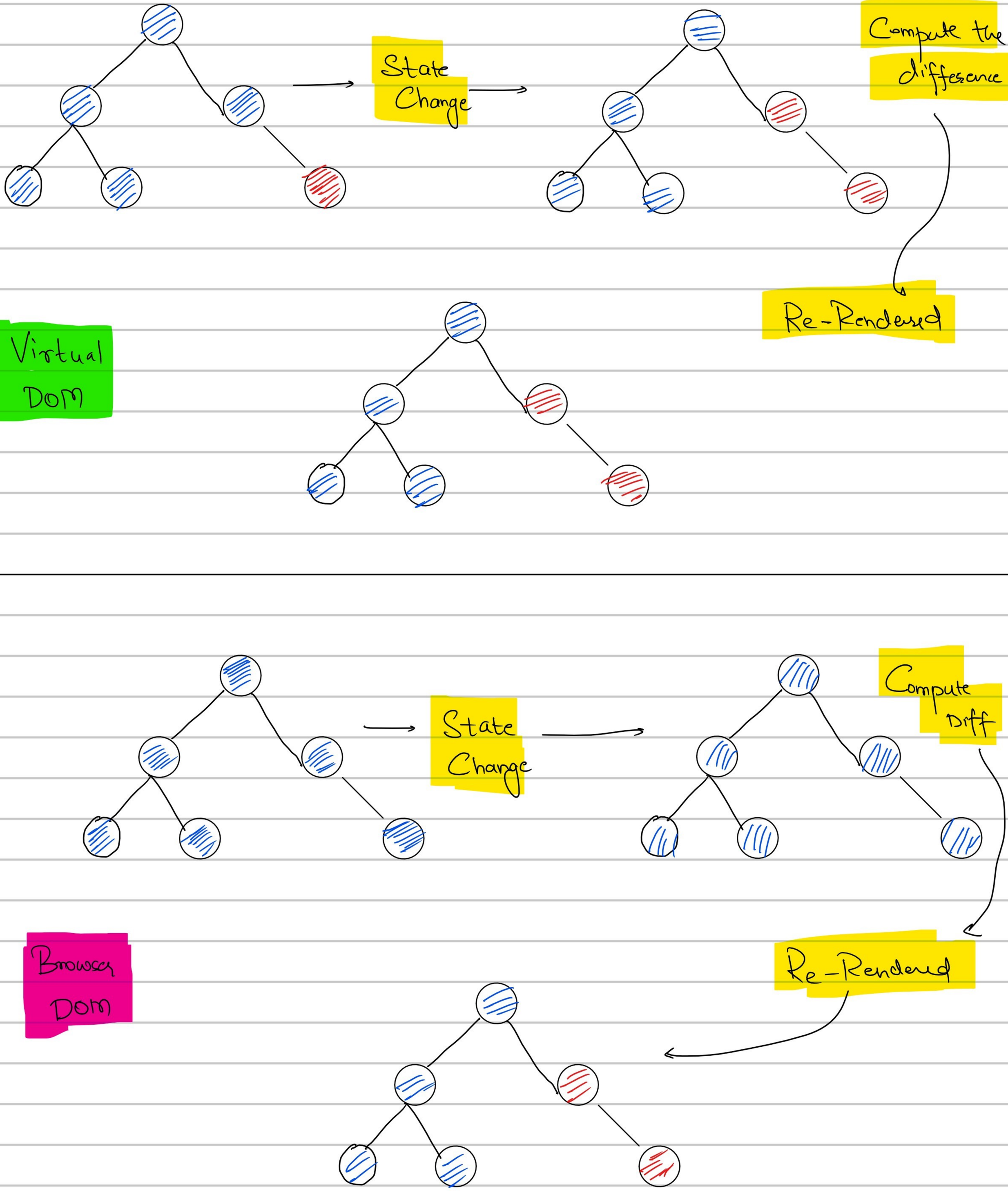
Virtual DOM

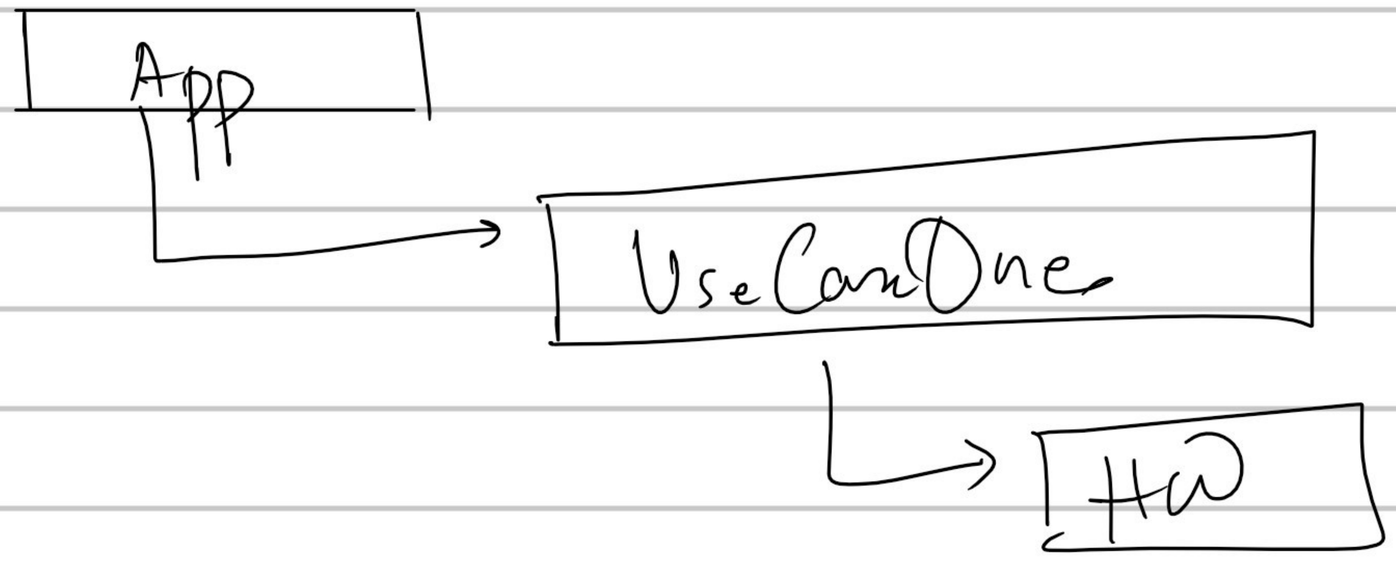
* When new elements are added to the UI, a virtual DOM which is represented as a tree is created.
* Each element is a node in the tree.

If state of this elements changes, new virtual DOM tree is created.

The tree is compared with previous virtual DOM tree

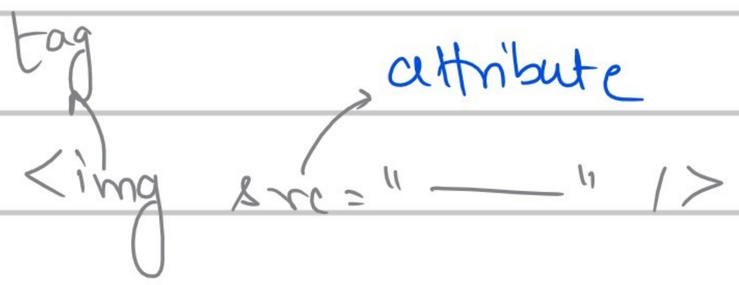
→ Diffed / Diffing


State Change
Compute the difference
Re-Rendered
Virtual Dom
State Change
Compute Diff
Re-Rendered
Browser DOM



* Props *

→ Normal HTML tag have attribute



- React Components have props

```
<NameComponent name="Zaintab" />
```

prop → equivalent to attribute

there can be any number of props &
they can be of any name

```
<NameComponent
   name="Z" />
```

props are captured

① Class Components → available in an object called as props

② Function Component → available as an argument

* State *

State is a plain JS object used by React to represent the information about a components current-situation.

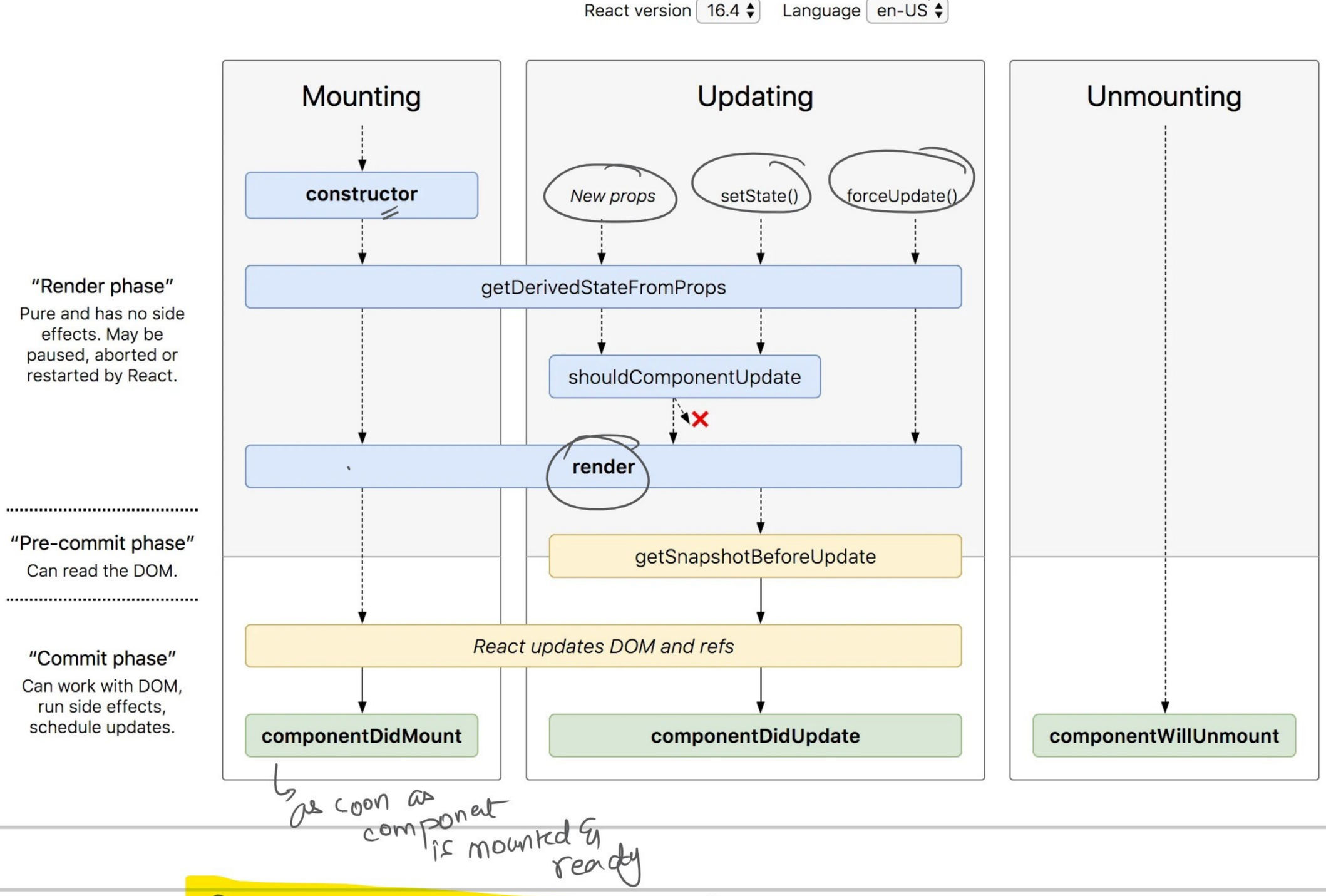
```
class SomeComponent extends &Component {
    state={

    }
    randy(){

    }
}
```

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



as soon as component is mounted & ready

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

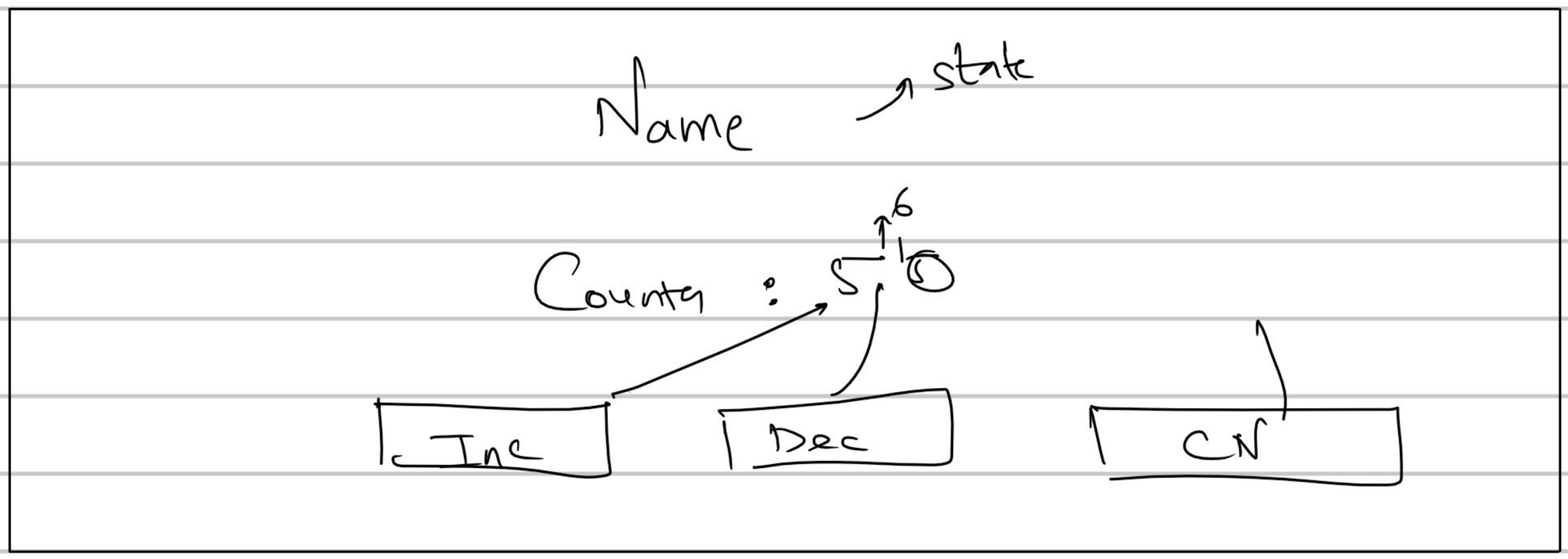
In previous React versions, if a component needed state, it has to use a class component

useState → hook

```
import {useState} from 'react'

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

- variable → state
- setVariable → function to update the state
- useState → hook for creating state in function component
- defaultValue → default value for state variable

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

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

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

const [addres, setAddres] = useState({county:'India'});
```

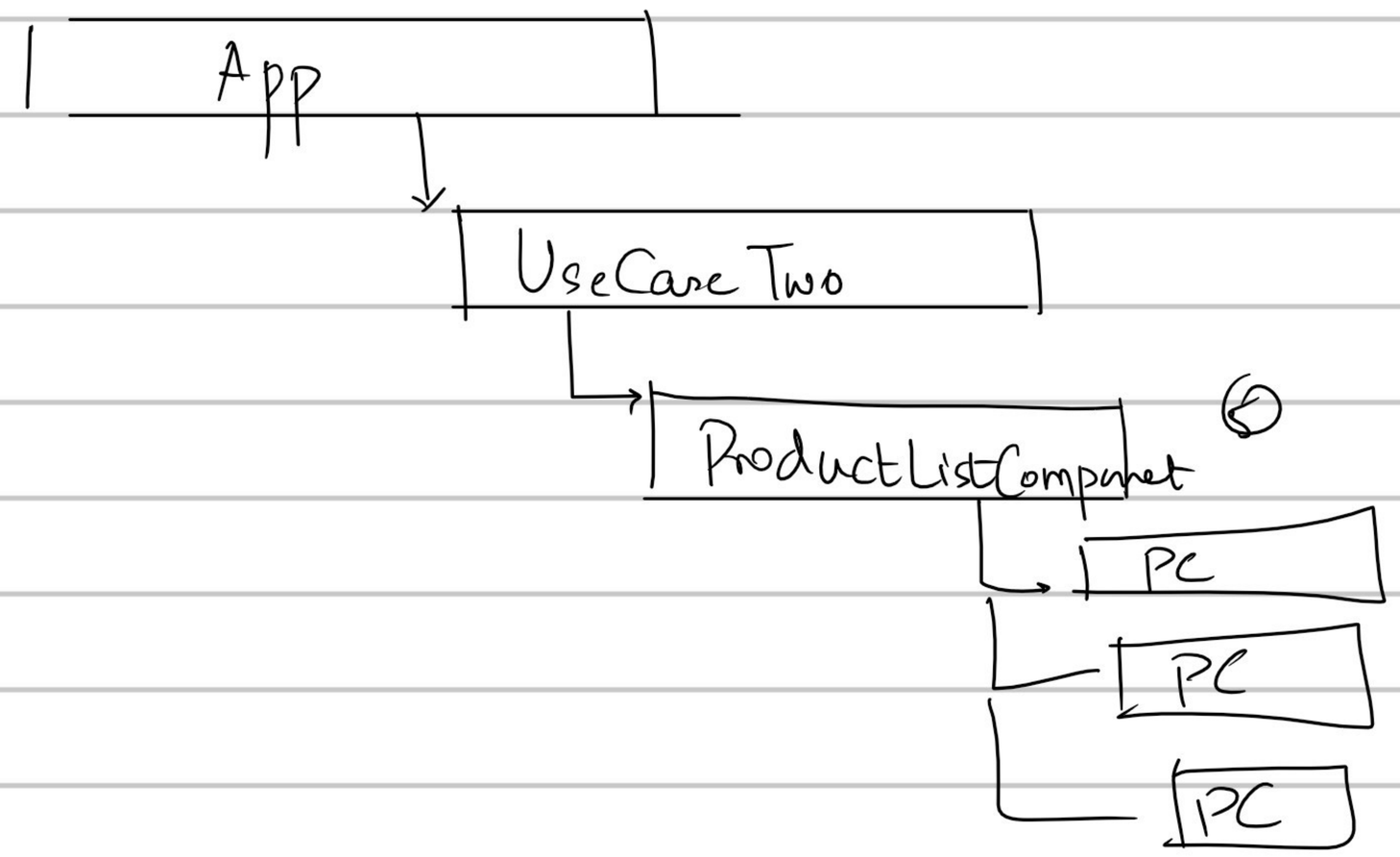


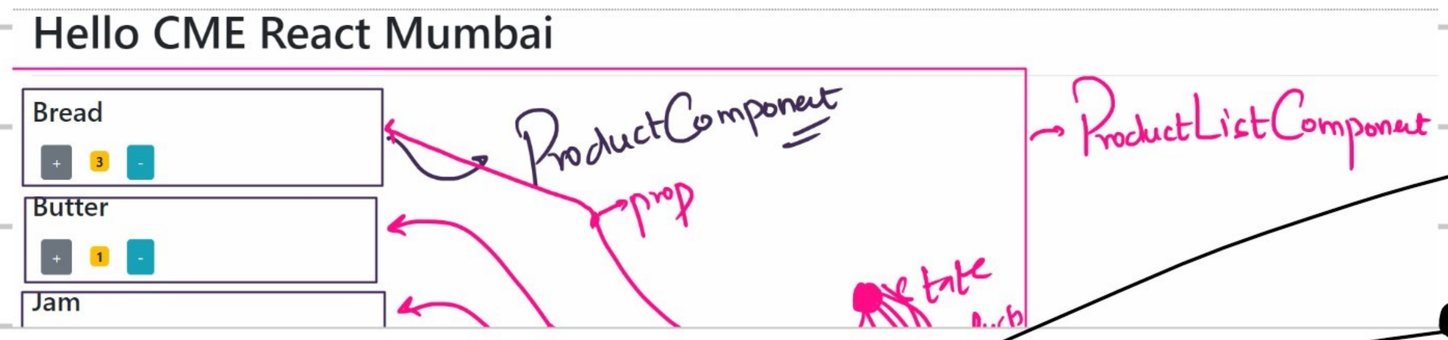
Whenever project need some dependency

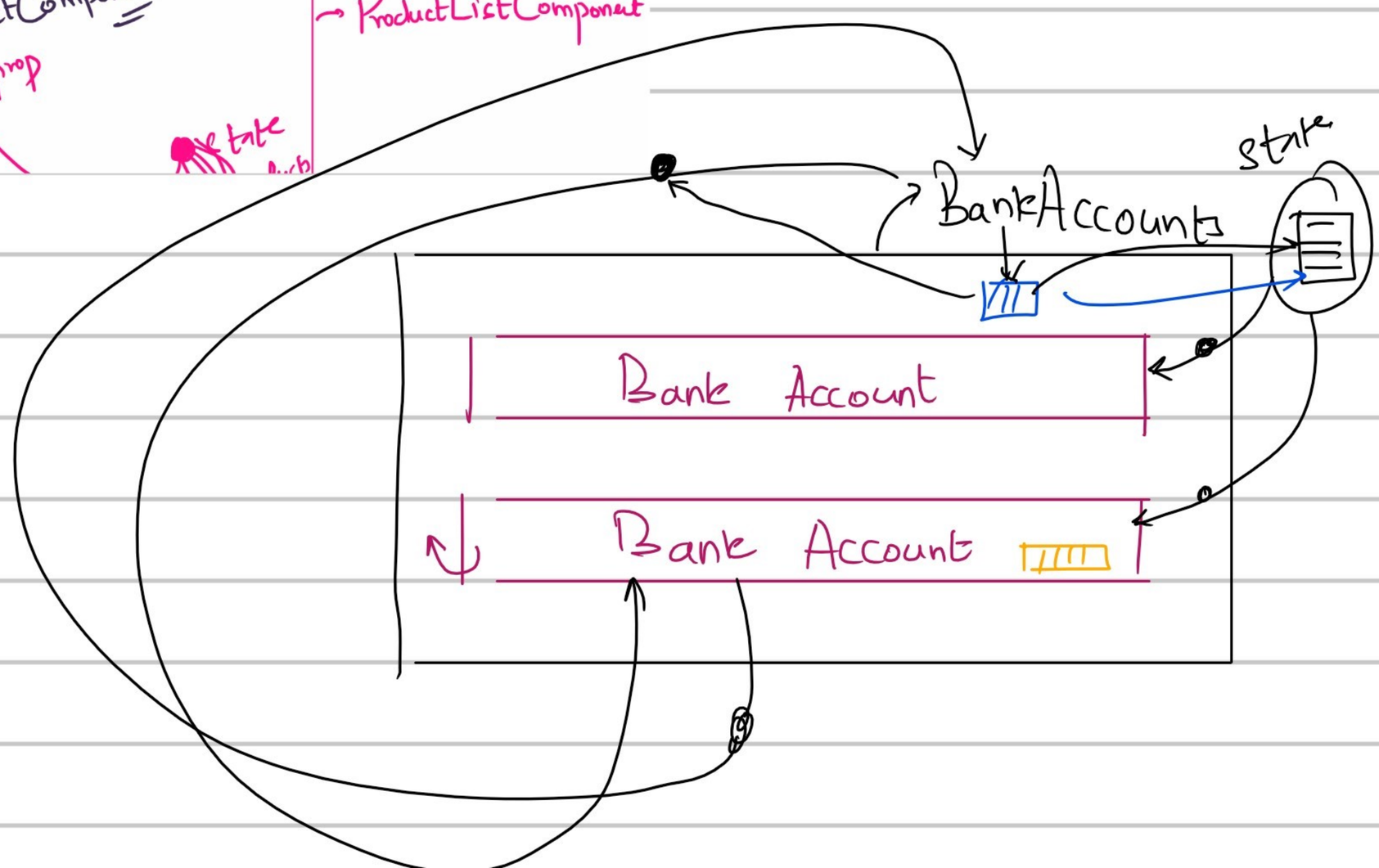
Go inside the project folder

```
npm i dependency@version
```

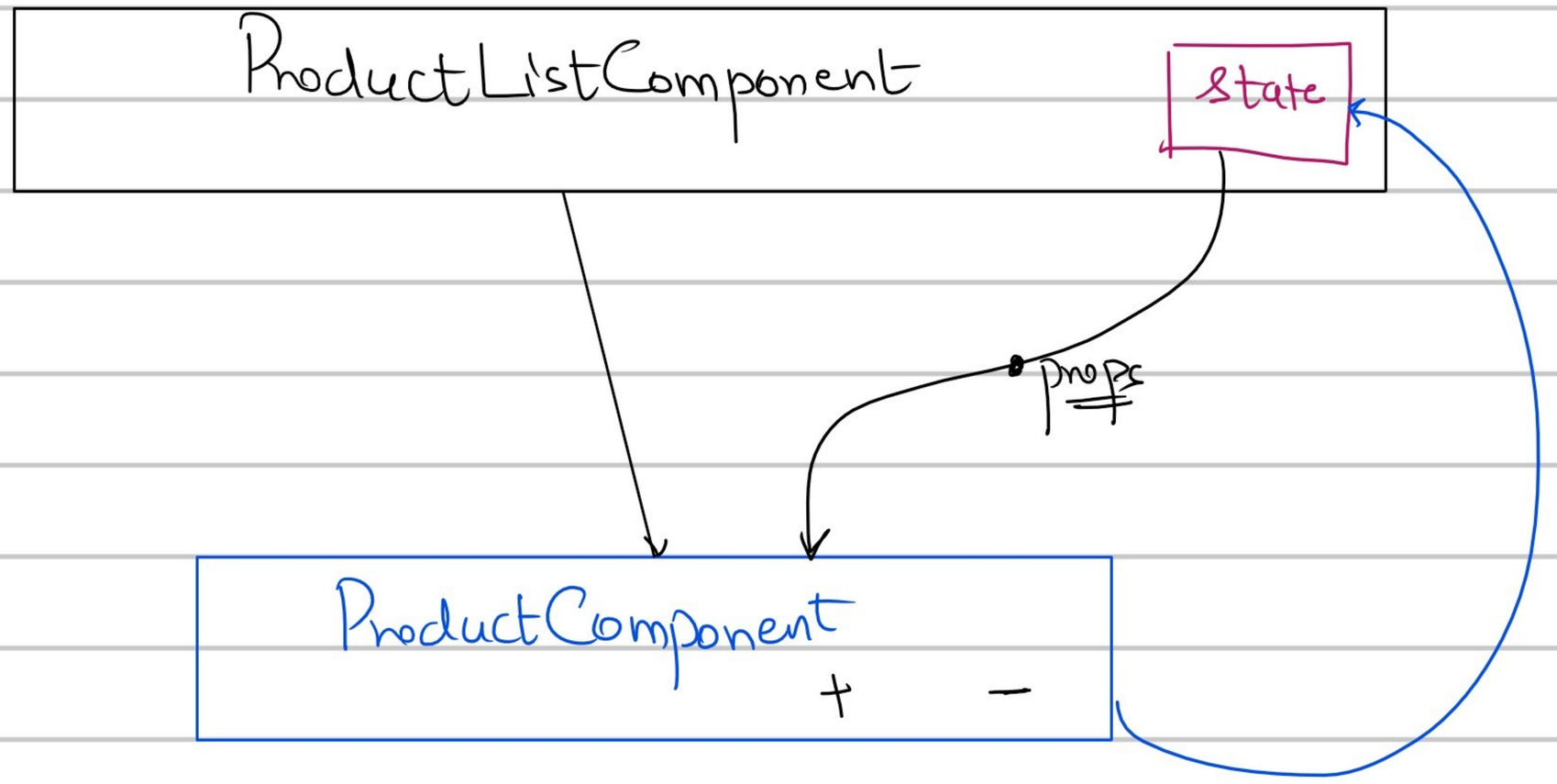
```
npm i bootstrap@4.1.1
```



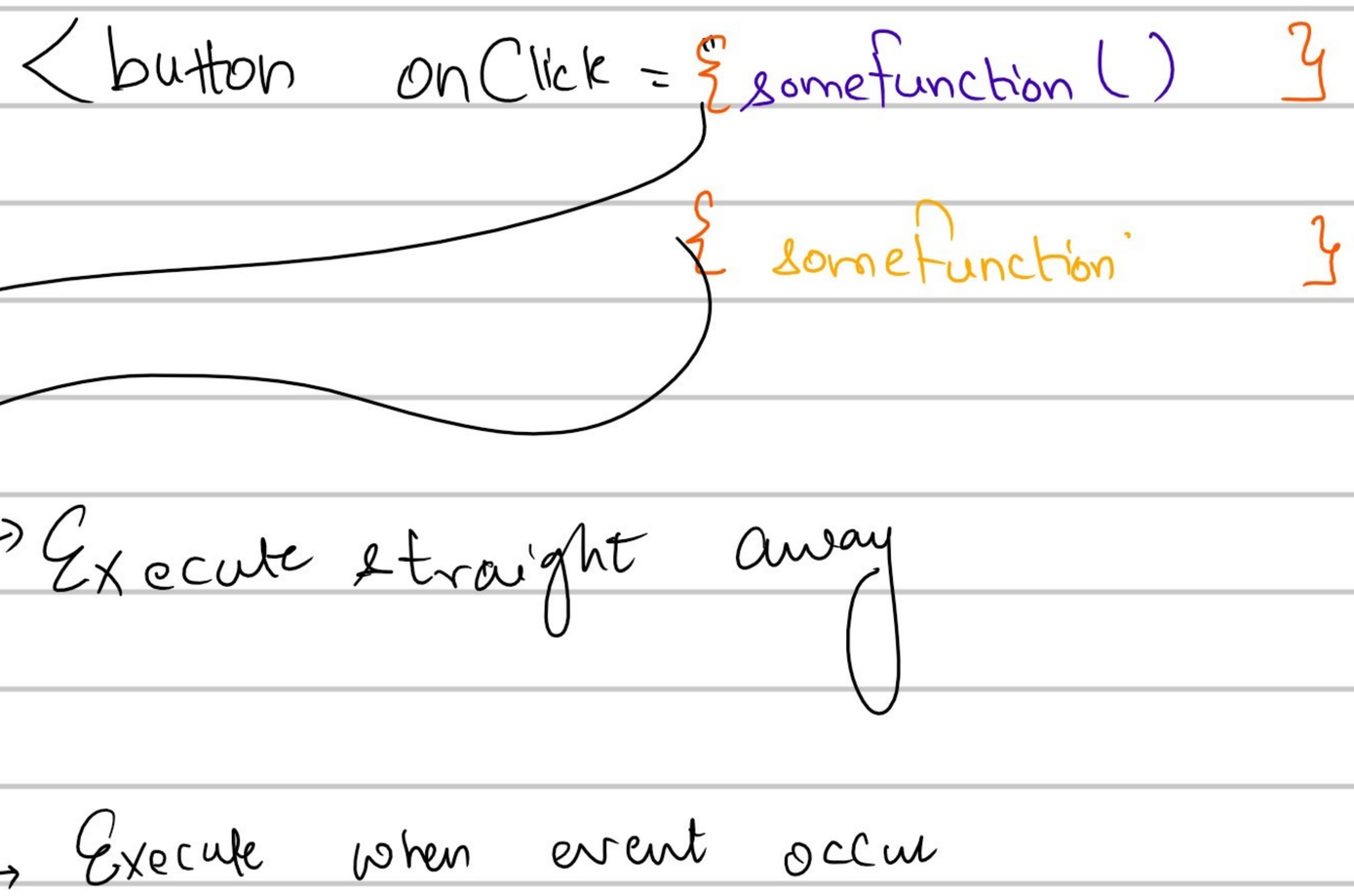


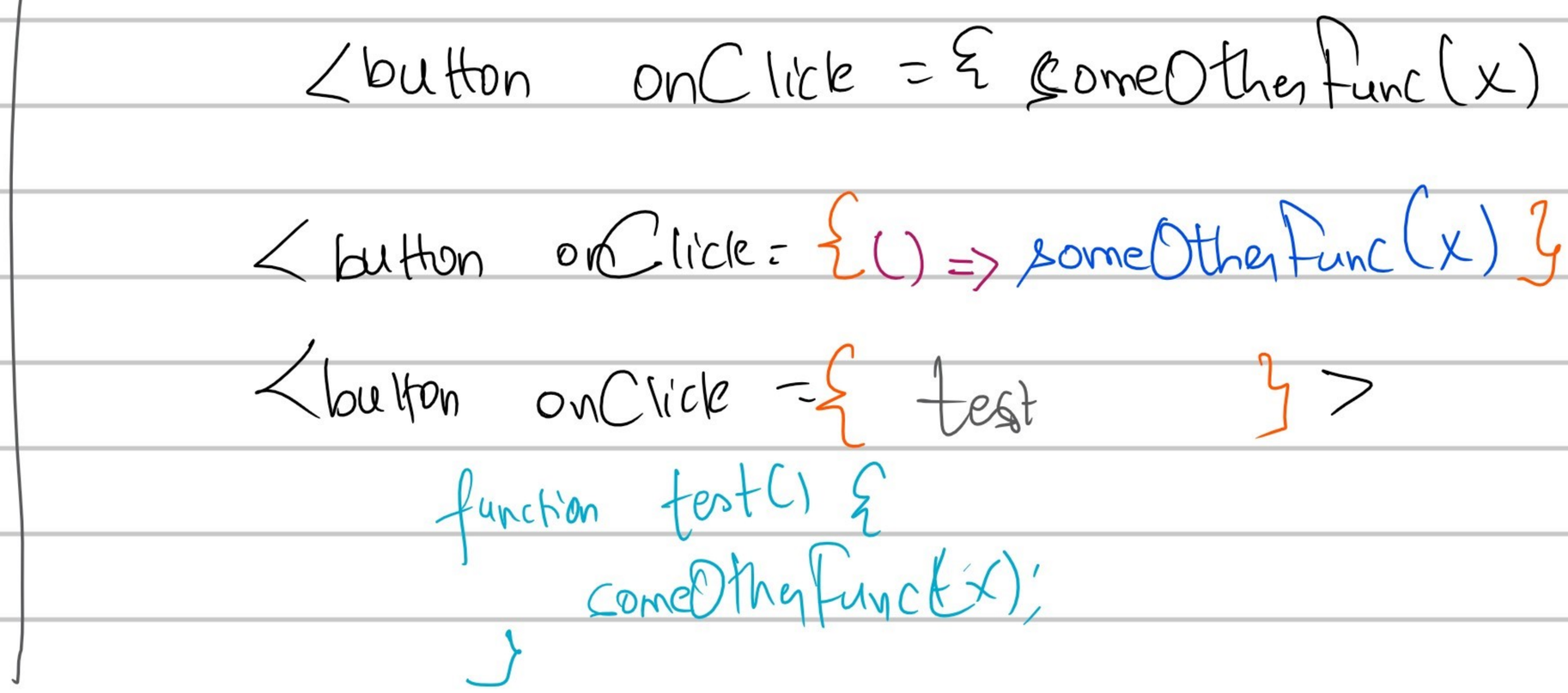


State should be modified by the component to which it belongs!!



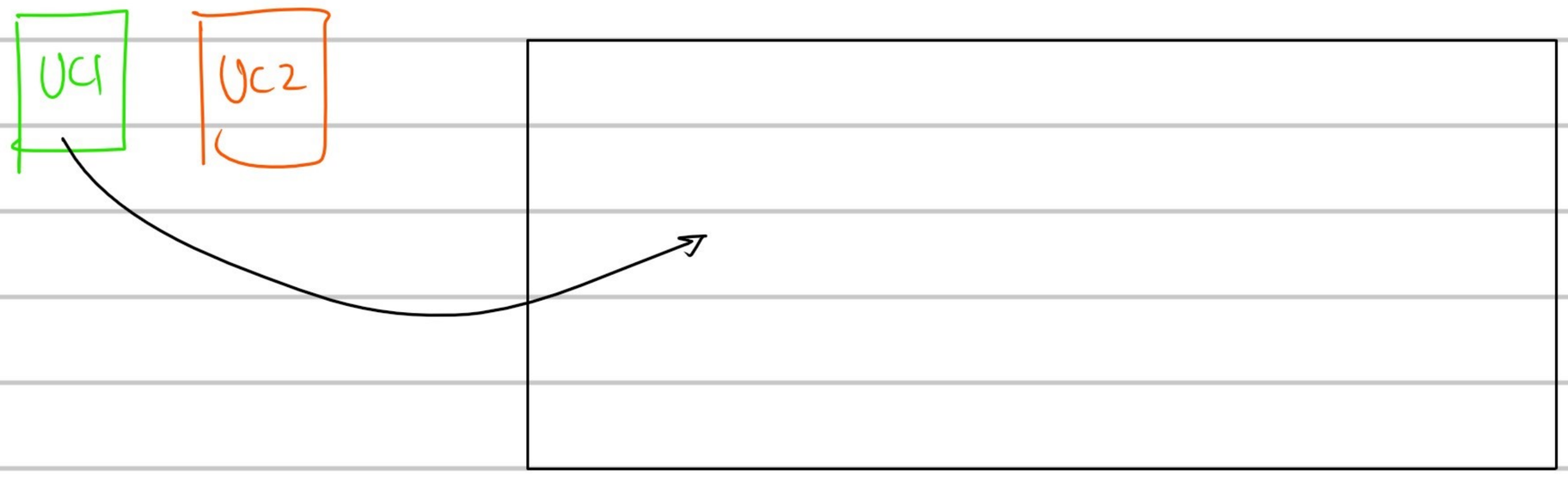
* PLC is responsible for updating the state
* But the action to update the state will come from PC
* PLC will be required to pass a reference of a function to PC via props; which PC can invoke when it requires to make the changes


<button onClick={someFunction()}
{someFunction}
() → Execute straight away
→ Execute when event occur
clicked


<button onClick={someOtherFunc(x)} />
<button onClicle={() => someOtherFunc(x)}
<button onClick={test} />
function test() {
someOtherFunc(x);
}

React Routing

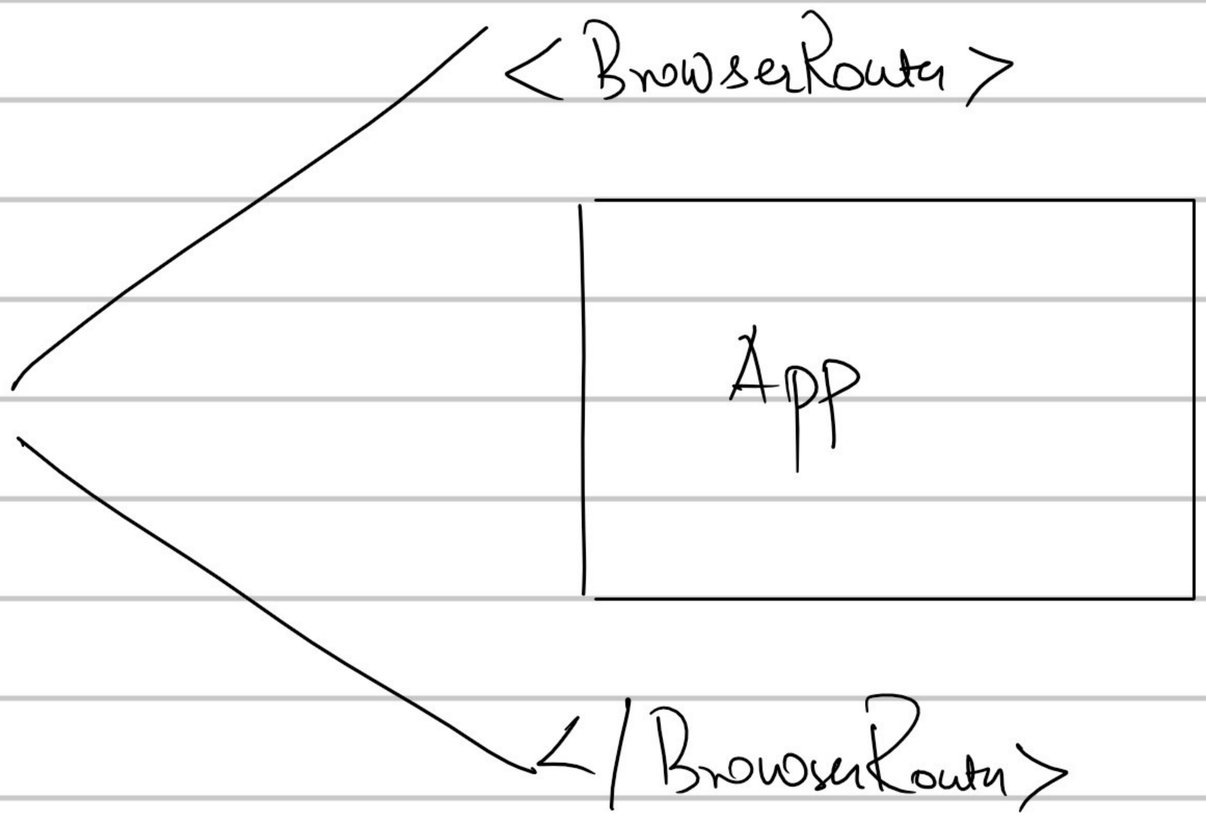
React App → SPA → Single Page Application.




```
npm i react-router-dom
```

* BrowserRouter *

hold all the browser history



* Route *

<Route> → Component of react-router-dom

→ renders the component when URL matches

* Link *

→ declarative, accessible navigation around project.

```
<a href="/products"> Products </a>
```

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

* Route Parameters *

When a component is enclosed `<Route>`

→ navigate

→ my component's current location

→ route parameters

push → user can go backward & forward

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

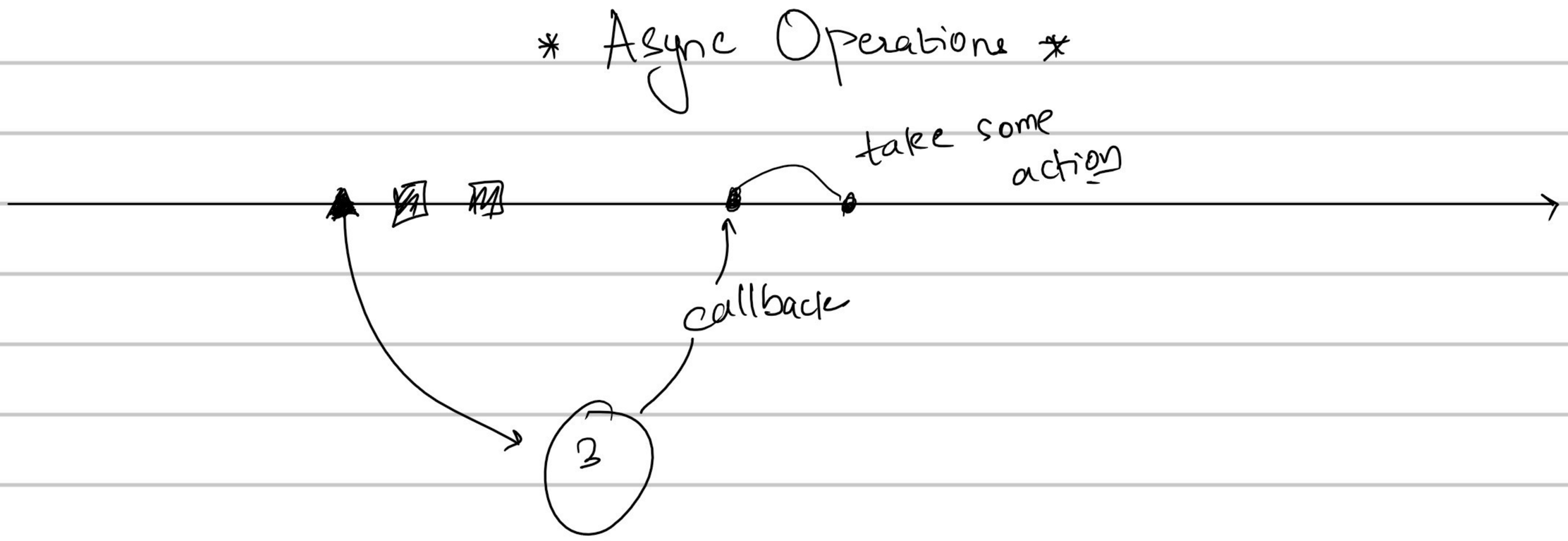
useEffect Hook

↳ lets you perform side effects in functional component

Class	Function
componentDidMount()	
componentDidUpdate()	useEffect(function)
componentWillUnmount()	

useEffect (function, dependency)

* Async Operations *



Promise

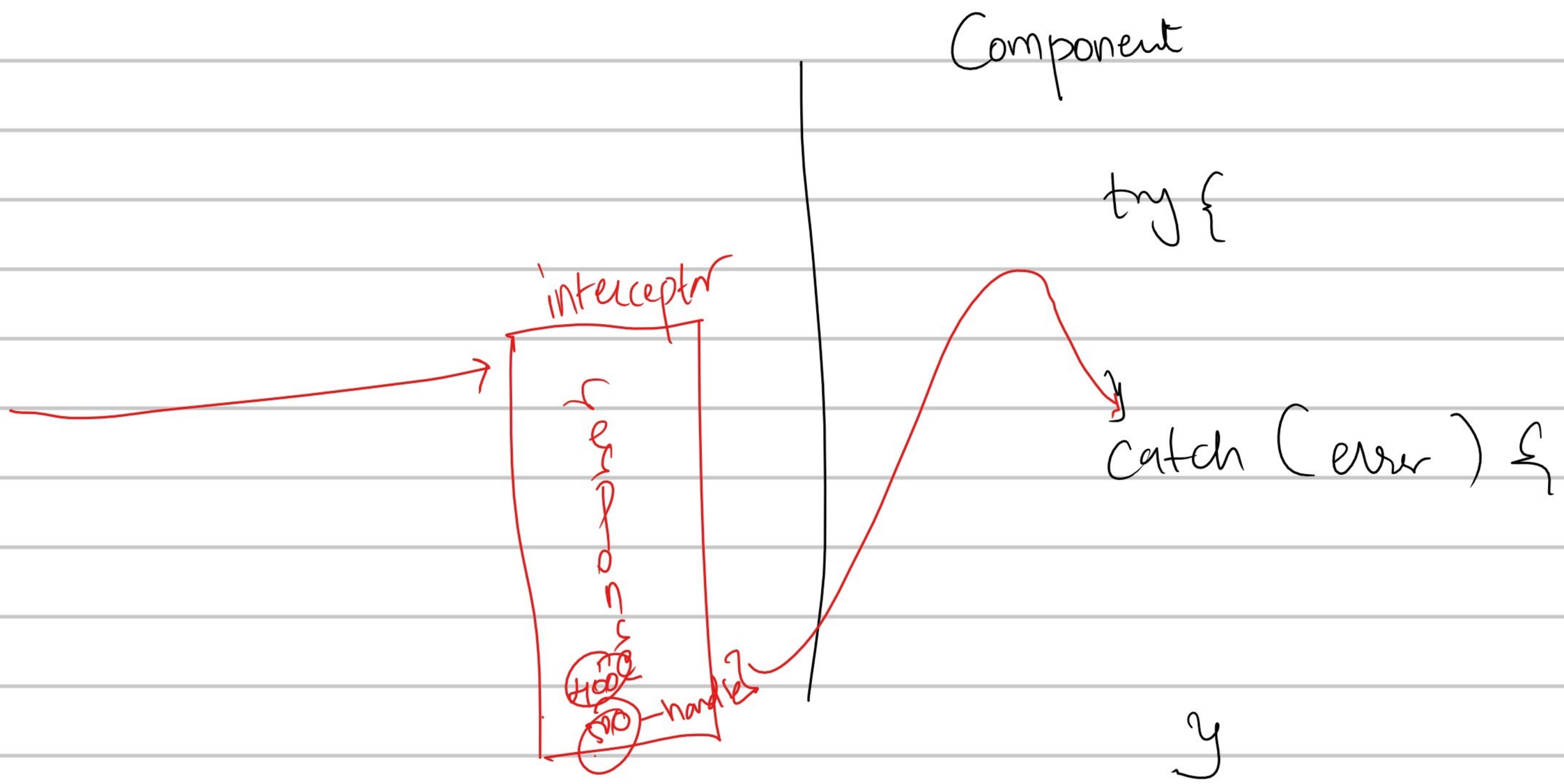
↳ holds the eventual result of an asynchronous operation

HTTP API

Fetch API

JQuery AJAX

Axios




```
Result from Joi:                                    LoginForm.jsx:17
  {error: ValidationError: child "username" fails becau
▼se ["username" is not allowed to be empty]. child "pa
  s…, value: {…}, then: ƒ, catch: ƒ} i
  ▶ catch: ƒ _catch(reject)
  ▼ error: ValidationError: child "username" fails becau
    ▶ annotate: ƒ (stripColorCodes)
    ▼ details: Array(2)
      ▼ 0:
        ▶ context: {value: '', invalids: Array(1), key:
          message: "\"username\" is not allowed to be em
        ▼ path: Array(1)
            0: "username"
            length: 1
          ▶ [[Prototype]]: Array(0)
          type: "any.empty"
        ▶ [[Prototype]]: Object
      ▶ 1: {message: '"password" is not allowed to be em
        length: 2
      ▶ [[Prototype]]: Array(0)
      isJoi: true
      name: "ValidationError"
    ▶ _object: {username: '', password: ''}
      message: "child \"username\" fails because [\"user
      stack: "ValidationError: child \"username\" fails
    ▶ [[Prototype]]: Object
  ▶ then: ƒ then(resolve, reject)
  ▶ value: {username: '', password: ''}
  ▶ [[Prototype]]: Object
```

schema = {

username = Joi.string()

Pass

}