Class based components

* Class userClass extends React.Component {

Render(){

Return <div>hi hello</div>

}

}

To add props to class

We use concept of constructor it is similar to passing parameters to class and access through constructor In js.

**Super()** is used to inherit the property or variables of parent class to child and with that the props is added

 Call the parent class’s constructor (super(...))

 Access methods or properties of the parent class

* Create instance of class means?
  + When loading class component to webpage it is called instance of class
  + And when we create instance of class we can use constructor to get props and state also

   super(props);

useState in class component

  constructor(props) {

    super(props);

    // create state

    this.state = {

      count: 0,

      countTwo: 1,

    };

  }

**State is big object variable and holds all the state variable**

**How to update state In class**

Never update state variables directly like below

this.state.count + 1

how to update?

React gives access to method through which we can pass the old state and update it

  onClick={() =>

            // never update the state directly

            // this.state.count + 1

            this.setState({

              count: this.state.count + 1,

            })

          }

If there are 5 class variables and we pass only one state variable it will only update the passed variables not all the classes

How the component is mounted in ui when the component is instantiated in class?

1.when code see class based component it instantiate the component then it calls constructor and after that render is called

There is another method provided by react called **componentDidMount** which will be called **after** **render method is called and component is rendered to page**

When there is child component present In parent how it works without constructor and render is

Parent constructor

Parent render

Child constructor

Child render

**What happen when there is componentDidmount in parent and child**

Parent constructor

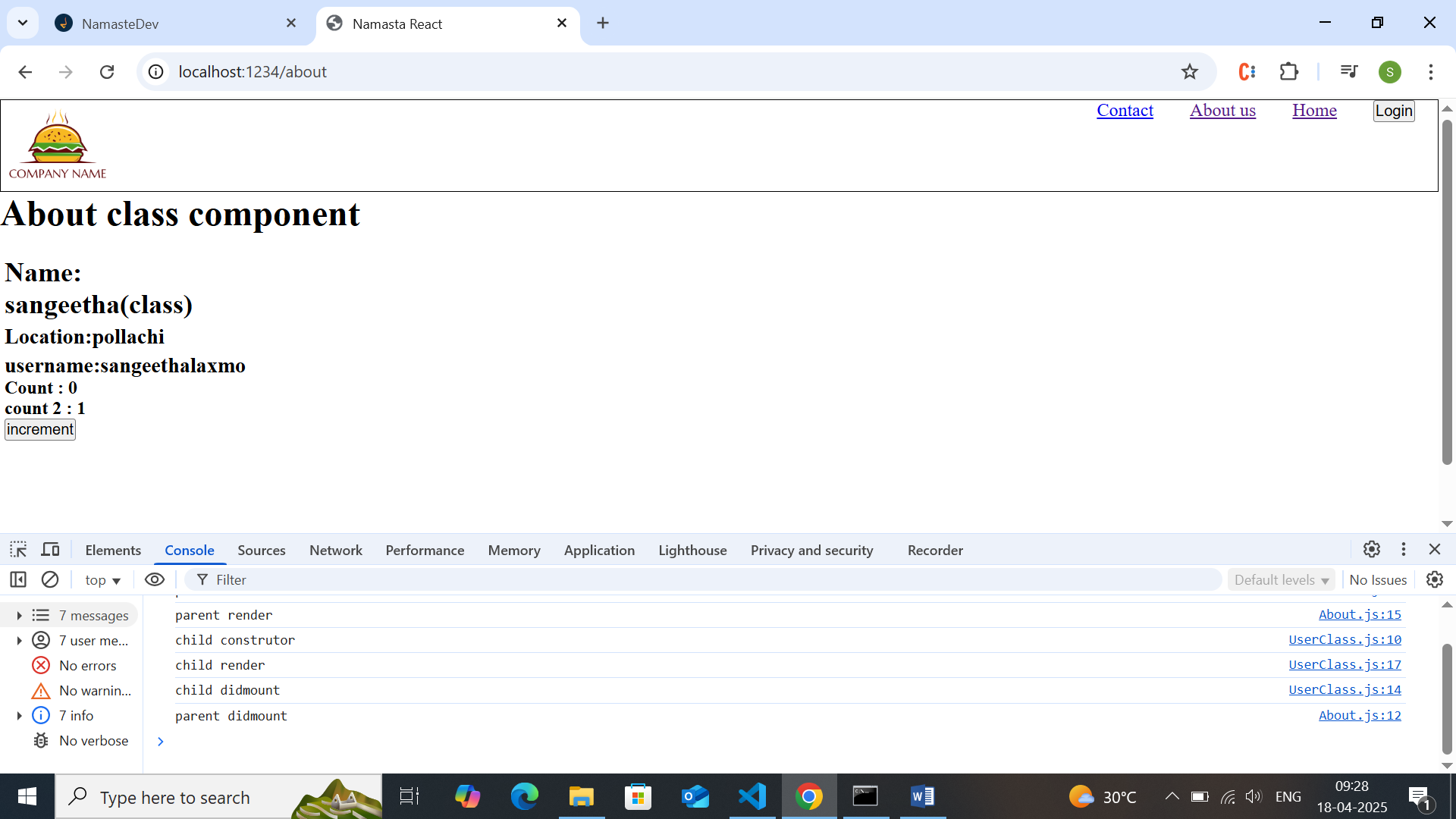
Parent render – if it see the child class and parent is not completed its render it but it starts the child process

Child constructor

Child render

Children componentDidMount -once child completed render the parent componentDidmount is called

parent componentDidmount



What is use of componentdidmount?

**1.There are some actions that we do after component is rendered like useEffect in functional component we use componentDidMount in class based component**

2.we first render the component and then make api call so we don’t want to wait for the api call to return the data – neeed to render component as soon as possible and then call api

**3.similer to functional based component we have useEffect we use componentDidMount**

**What happen if there are 2 children inside parent class component?**

Parent constructor

Parent render

Sees first child and first child constructor

1st Child render

**1st component did mount**

2nd constructor

2nd child render

**2nd component did mount**

Parent component did mount

Above order is wrong why?

Correct order

Parent constructor

Parent render

Sees first child and first child constructor

1st Child render

2nd constructor

2nd child render

<DOM updated in single batch>

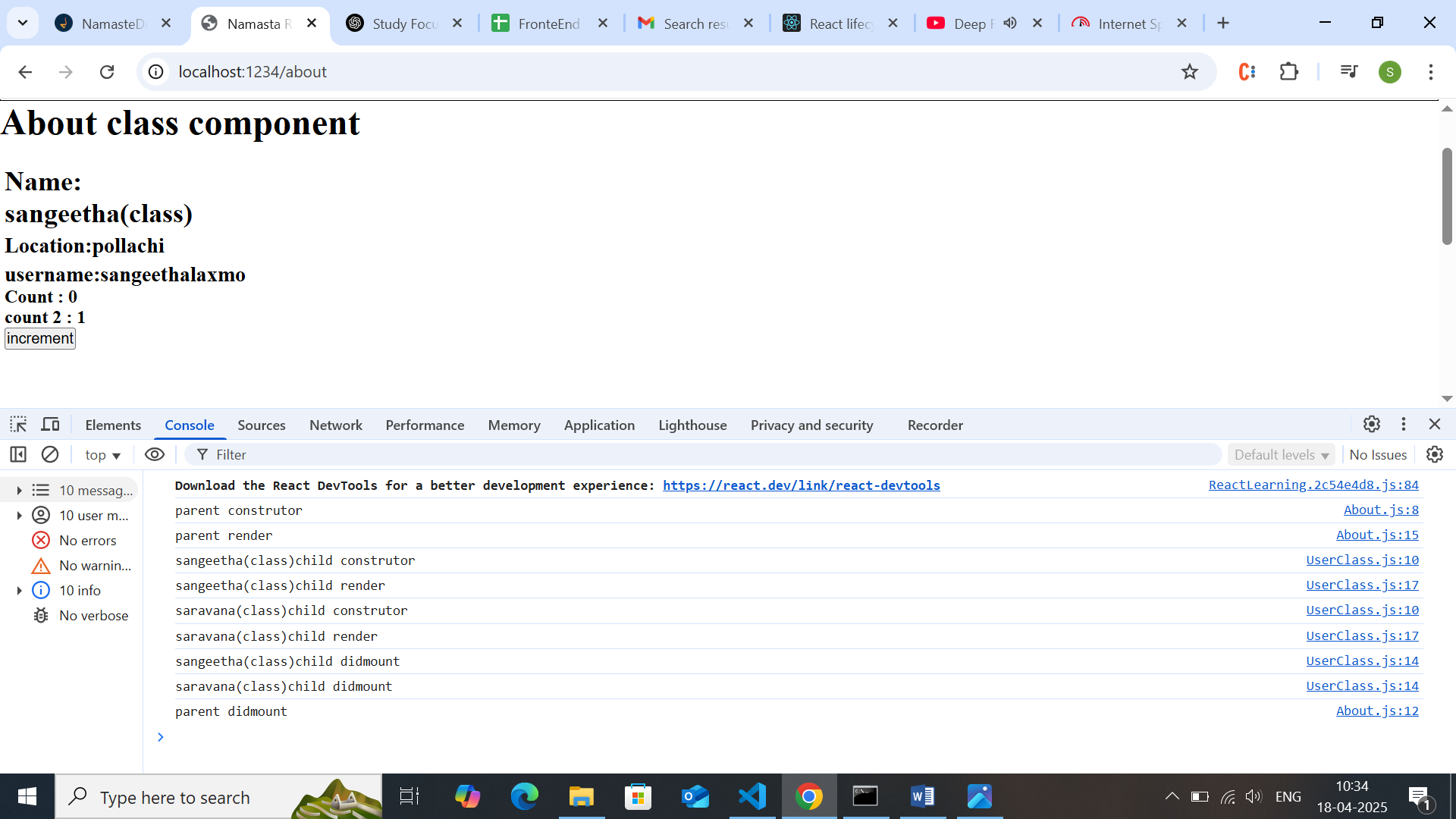
Note: till above they are render phase and diffing algorithm is performed after DOM is updated did mount is called

**1st component did mount**

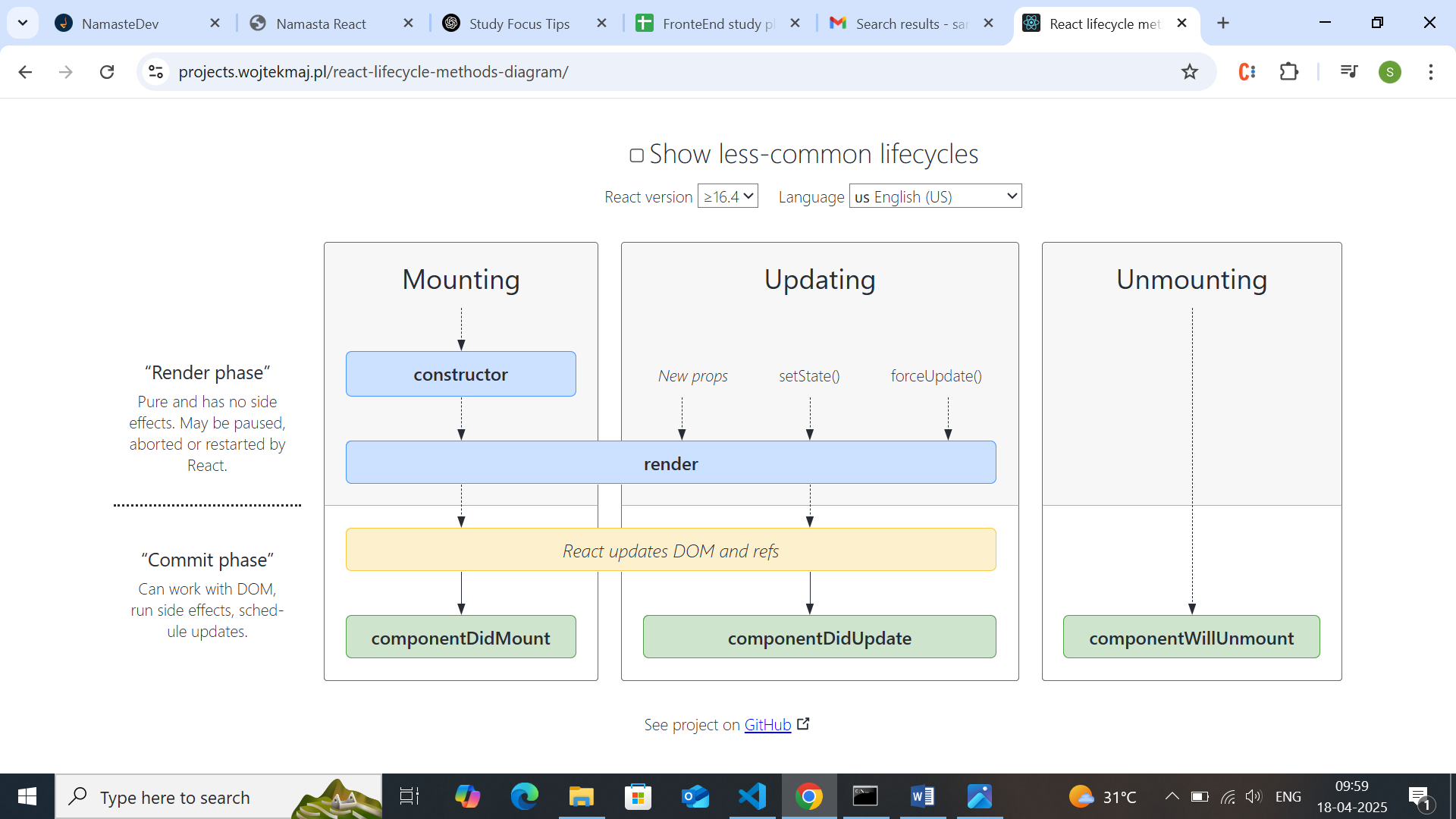
**2nd component did mount**

Parent component did mount

**Why did mount is called at last?**



There is react life cycle method in react



There are 2 phases in react on how component is mounted

* Render phase
* Commit phase

When component is mounted first render phase

**Render phase 🡪** constructor Is called then render method 🡪 in render phase the diffing algorithm is done and finds the different🡪render phase occurs with virtual DOM and react find difference b/t virtual DOM through diffing and gets all the difference in DOM

**Commit phase ->** react update DOM and refs , componentDidMount is called 🡪 in commit phase is very expensive and slow it is place where the DOM gets updated based on diffing algorithm and re-render the component which updates only the changed part of DOM

**So react uses concept of batching to batch all the render phase together and commit phase together this is reason why componentDidMount is first and second child is batched together and called after render is completed**

**Why react is doing batching? -> for better performance**

1.once the commit phase start the react will start to update DOM which is expensive and slow so this is reason why react try batches the render phase as separate and child phase sepratly and do componentDidMount after render and commit is completed it calls componentDidMount

It will optimize the react app in performance

**How to make api call with componentDidMount?**

1.we can make the componentDidMount method as async and use await to fetch the data inside it and set data

The component mount is called after the component is mounted for first time and calls api

 \* component life cycle

 \* ----Mounting cycle

 \* Constructor (dummy data)

 \* render(dummy data)

 \*    Render html with dummy data

 \* Component Did mount     --> called only once after first time component is rendered -->similar to useEffect with [] dependcy array

 \*    <Api clls)

 \*    <this.setState-->updates state>

 \* ------Update cycle

 \* render  (api data)

 \* update html with api data

 \* componentDidUpdate is called and then ->this is called when ever there is change in state variable and components re-render again -->similar to useEffect without dependency array or useEffect with particular dependency

 \* componentDidunmount --> once we move out from one component to another -->event listerner removing in component,data clean to prevennt memory leaks -->similar to clean up function in useEffect🡪return from useEffect

**ComponentDidUpdate:**

This is called when there is update in any state variables and react need to re-intiate the render and commit process again to update the DOM

When we need to do any action after any state is changed we write those logics here

**ComponentWillunMount**

This is called when user leave current component and moves away from it

What we do during componentdidunMount?

1.it is called when we leave the component

We need to clear lot of those before we leave a component.. what are the things we need to clean up?

Eg: if we have setintervel in one component and it will set even if we move away from component

Why?

In single page application we just change the components from one to another but the page remains without refresh so when we setintervel in once component it attaches to browser without removing it and continue to be in browser which is huge memory lose and performance lose

This is the reason why we need to componentunmount .. where we will clear the interval

**In class ‘this’ can be used throughout the class component**

We cannot use async as callback with useEffect why?

useEffect(async () => {

const data = await fetchData(); // ❌ This gives a warning!

}, []);

🔴 This is not allowed because:

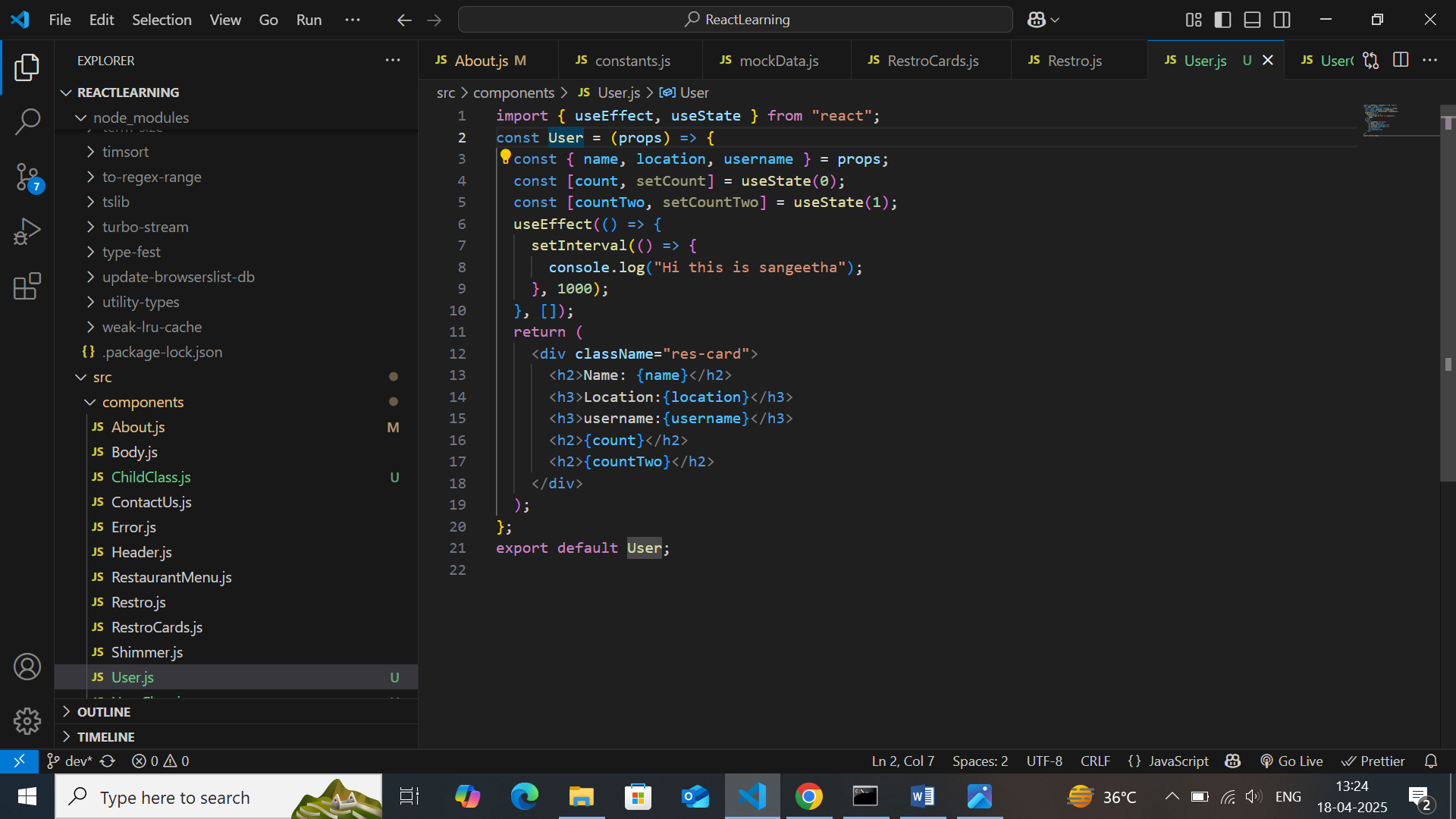
**useEffect expects a function that either:**

* Returns nothing, or
* Returns a cleanup function

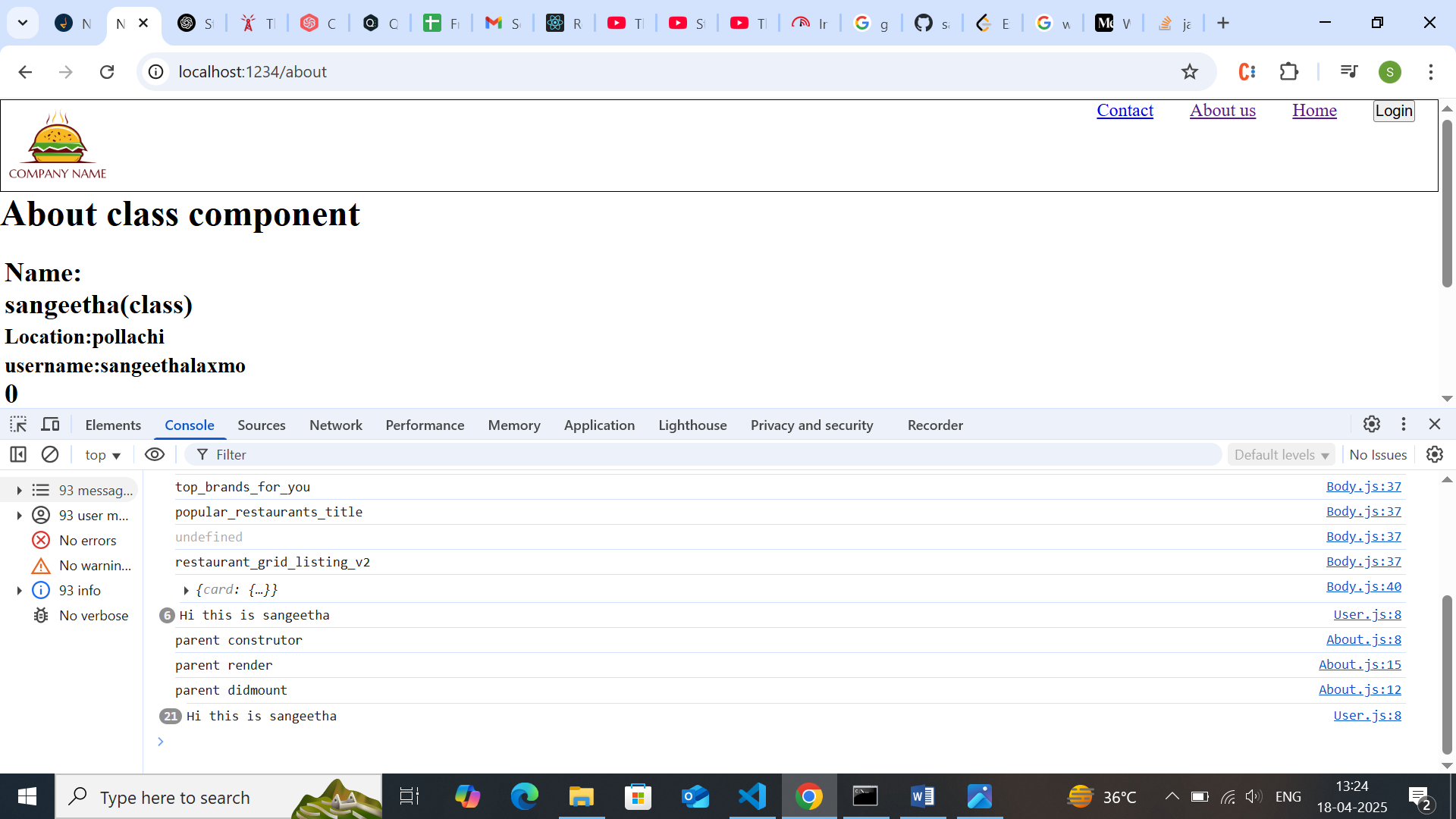
**But an async function always returns a Promise, which breaks the expected pattern.**

React doesn't know what to do with the Promise returned from an async function.

**Similar to componentwillunmount we use return inside useEffect() to call once user move from one component to another component**



**above** time interval will not be stoped even if we leave the component



  useEffect(() => {

    const timer = setInterval(() => {

      console.log("Hi this is sangeetha");

    }, 1000);

    return () => {

      clearInterval(timer);

    };

  }, []);

The return function will be called after user leave the component and clean the interval

**super(props) in Constructor**

1. **You're extending a parent class (React.Component)**
2. **You need to call the parent's constructor**
3. **You want to access this.props inside your constructor**

In JavaScript, when using **class extends**, if you create a constructor, you **must** call super() **before** using this.

If you don’t, you'll get an error like:

ReferenceError: Must call super constructor in derived class before accessing 'this'

**What does super(props) do?**

* Calls the **constructor of the parent class** (React.Component)
* Passes props up to React.Component, so it can initialize this.props

This makes this.props available inside the constructor and lifecycle methods.

### ❗️What if you just write super() without props?

constructor(props) {

super(); // ❌ `this.props` will be undefined in the constructor

}

You won’t get a crash, but this.props won’t be set properly inside the constructor.