React :: LifeCycle Methods

Components are basically the pieces of functions which can independently called as Tags within a program .

Like in the HTML , we know that simple Tag of

<select id=”name”>

<option>Anuj</option>

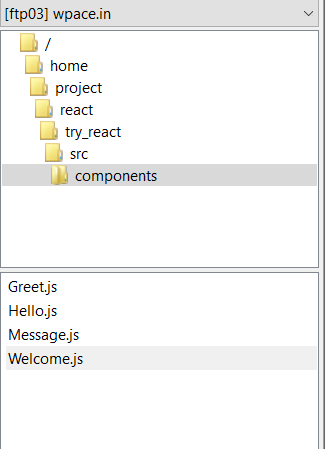
<option>Ayaan</option>

<option>Sonal</option>

</select>

Will generate the combo box on the browser . But I cant make a function where in there is a Select combo box along with the Submit button which automatically submits the combo box .

So in react

1. **Create a Folder components in the SRC folder**
2. 
3. **Now create a file name named *Message.js***

Import React from ‘react’

function Message(){

return <h1> HEllo Anuj</h1>

}

export default Message

1. **Now this File is being called by the *App.js***

Import React from ‘react’

Import Abc from ‘./components/Message’

function App() {

return (

<div className="App">

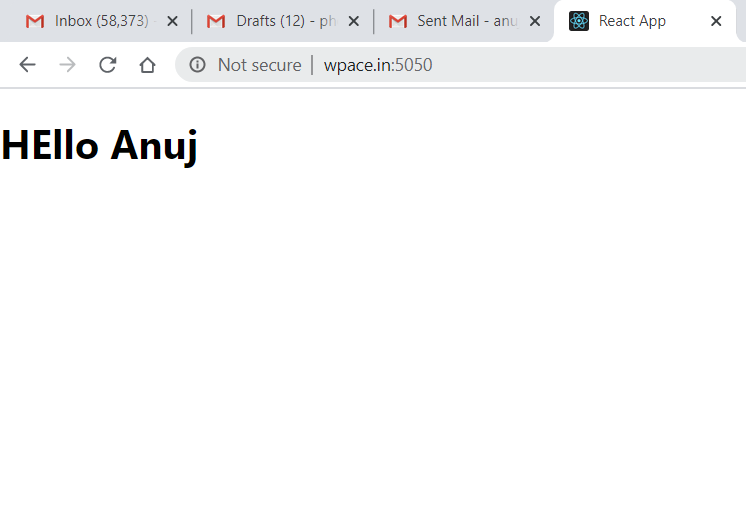
<Abc />

</div>

} ;}

Export default App;

1. The Output on the Browser



React :: LifeCycle Methods

Rest steps remain as such

1. **Now create a file name named *Message.js***

Import React from ‘react’

function Message(props){

return <h1> HEllo ~~Anuj~~ {props.name} your Gender is {props.gender}</h1>

}

export default Message

1. **Now this File is being called by the *App.js***

Import React from ‘react’

Import Abc from ‘./components/Message’

function App() {

return (

<div className="App">

<Abc name="Anuj" gender="male" />

<Abc name="Ayaan" gender="male" />

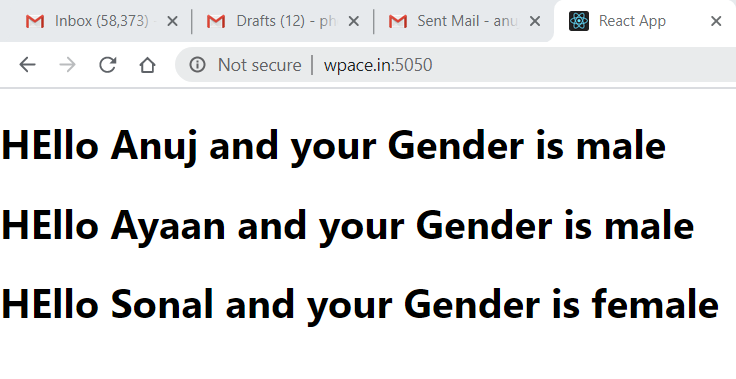
<Abc name="Sonal" gender="female" />

</div>

} ;}

Export default App;

1. The Output is



Point to note that Since we are using the default in export syntax that’s why we can use Abc as name of our component .

Else other option is

Import React from ‘react’

export function Message(props){

return <h1> HEllo {props.name} your Gender is {props.gender}</h1>

}

//export default Message

Here the Error comes

Failed to compile

./src/App.js  
Attempted import error: './components/Greet' does not contain a default export (imported as 'Abc').

This error occurred during the build time and cannot be dismissed.

Because it is NAMED EXPORT since it is exporting only the function with name “Message”

Replace ‘Abc’ with ‘Message’

Import React from ‘react’

Import {Message} from ‘./components/Message’

function App() {

return (

<div className="App">

<Message name="Anuj" gender="male" />

<Message name="Ayaan" gender="male" />

<Message name="Sonal" gender="female" />

</div>

} ;}

Export default App;

Now with JSX Element

return(

React.CreateElement(

“div”,

{className:”App”},

React.CreateElement(

“Message”,

{name:”Anuj”,gender:”male”});

React.CreateElement(

“Message”,

{name:”Anuj”,gender:”male”});

React.CreateElement(

“Message”,

{name:”Anuj”,gender:”male”});

React :: LifeCycle Methods

Like the function they can also accept properties and generate the JSX (HTML)

Class Components

They can also maintain **State locally.**

Generates JSX (HTML)

Props as Input

Lets create a basic component

1. **It should first import the Component class from the ‘react’**

Import React,{Component} from ‘react’

1. **Next it should have class with name of component**

class CMessage extends Component {

}

1. **Then it should have a render function which can return NULL or HTML**

class CMessage extends Component {

render(){

return <h1> Hello Anuj </h1>

}

}

1. **Now create a file name named C*Message.js***

Import React,{Component} from ‘react’

class CMessage extends Component {

render(){

return <h1> Hello Anuj </h1>

}

}

export default Message

1. **Now this File is being called by the *App.js***

Import React from ‘react’

Import CMessage from ‘./components/CMessage’

function App() {

return (

<div className="App">

< CMessage name="Anuj" gender="male" />

< CMessage name="Ayaan" gender="male" />

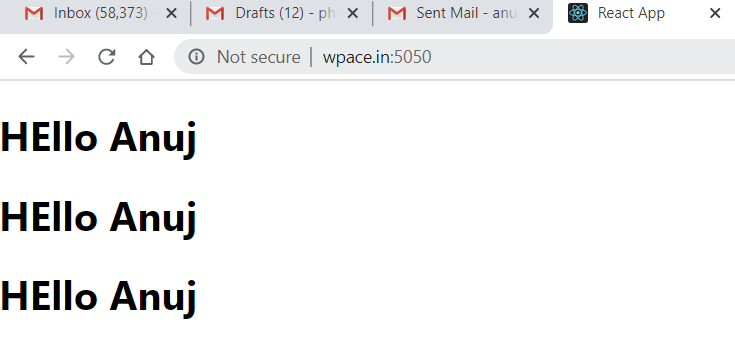
< CMessage name="Sonal" gender="female" />

</div>

} ;}

Export default App;

1. The Output is



1. Passing Variables in the CLASS COMPONENT ***CMessage.js***

Import React,{Component} from ‘react’

class CMessage extends Component {

render(){

const {name,gender} = this.props  
<div>

<h1>HEllo {this.props.name } , your gender is {this.props.gender}</h1>

<h1>Hello {name}, your gender is {gender}</h1>

<p>{props.children}</p>

</div>

}

}

export default Message

Few things to note :-

1. The props property is IMMUTABLE ie its values cant be changed
2. It returns only one HTML element . So they have to be wrapped in single <div> element.

**De Structuring props**

render(){

const {name,gender} = this.props  
<div>

<h1>HEllo {this.props.name } , your gender is {this.props.gender}</h1>

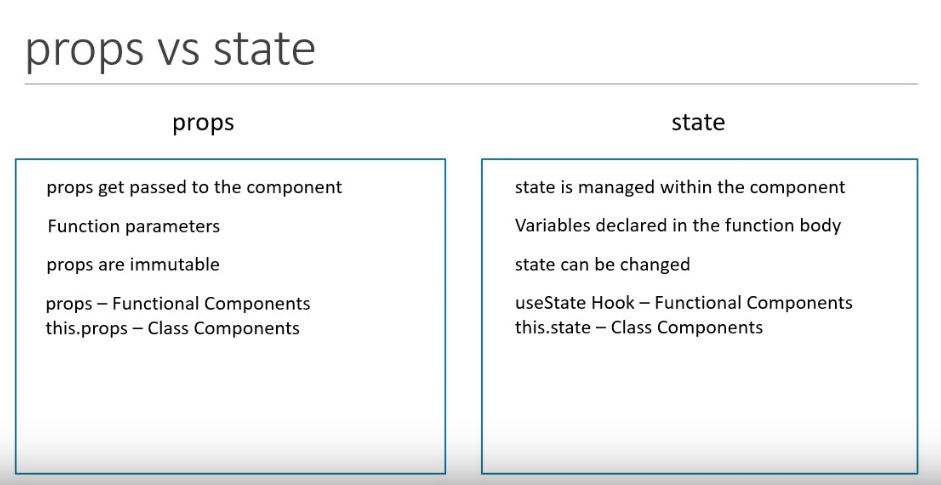
<h1>Hello {name}, your gender is {gender}</h1>

<p>{props.children}</p>

</div>

React :: LifeCycle Methods

Another way of passing the values in the JSX which renders the HTML by way of children component .



For using the STATE , it is like session variables of PHP which stores the values and changes on runtime .

Lets work around this using an example where the div displays the message

“Welcome Visitor”

And once the Subscribe button is clicked it says

“Thank you for subscribing “

1. We need to constructor to define the original value in the STATE

constructor() {

super()

this.state = {

message: "Welcome Visitor"

}

}

1. We need to call the function inside the JSX by calling the EVENT method

<button onClick={() => this.changeMessage()}>Subscribe</button>

1. Lets Define the event function by changing the value of the state in the Event Function

changeMessage(){

this.setState({

message:"Thank you for subscribing"

})

}

1. The complete Program CMessage.js

import React , {Component} from 'react'

class CMessage extends Component

{

constructor(){

super()

this.state ={

message: "Welcome Visitor"

}

}

changeMessage(){

this.setState({

message:"Thank you for subscribing"

})

}

render(){

return (

<div>

<h1>

{this.state.message}

</h1>

<button onClick={() => this.changeMessage()}>Subscribe</button>

</div>

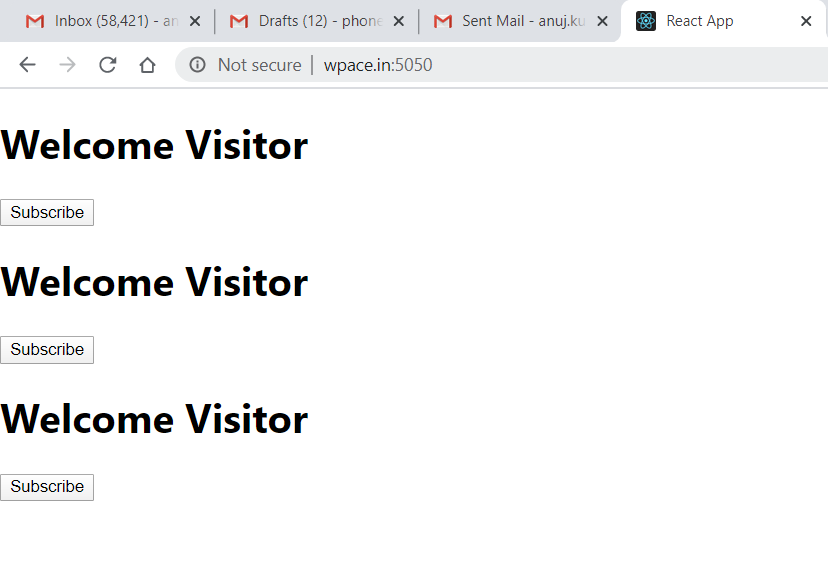
)

}

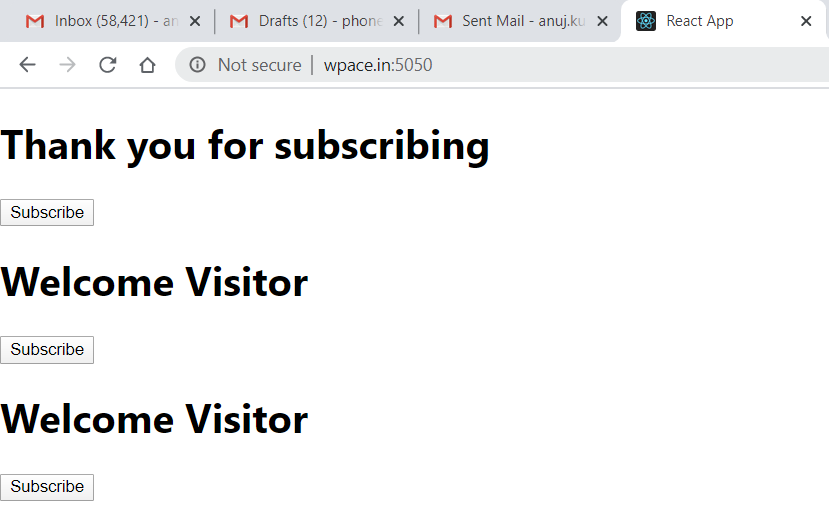
}

export default CMessage

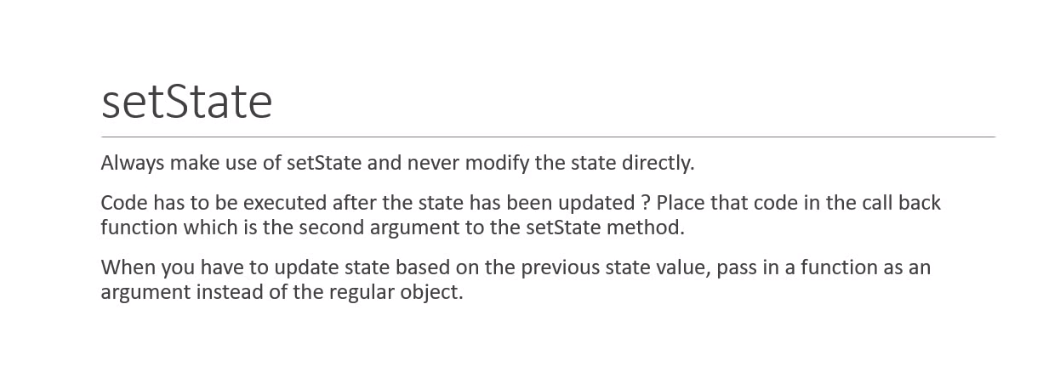
Before :-



After :-



React :: LifeCycle Methods



1. Now we should do it by incrementing the counter on click of the button

***Counter.js***

import React,{Component} from 'react'

class Counter extends Component

{

constructor(){

super()

this.state={

count:0

}

}

increment(){

this.setState({

count:this.state.count + 1

})

console.log(this.state.count)

}

render(){

return(

<div>

<h1>{this.state.count}</h1>

<button onClick={() => this.increment()}>increment</button>

</div>)

}

}

export default Counter

**Synchronous vs Asynchronous CALLS**

Note two sentences

<button onClick={() => this.increment()}>increment</button>

Whenever the Event is clicked , it makes an asychrounous call to the setState function

Whereas the console.log function is called synchronously .

Increment (){

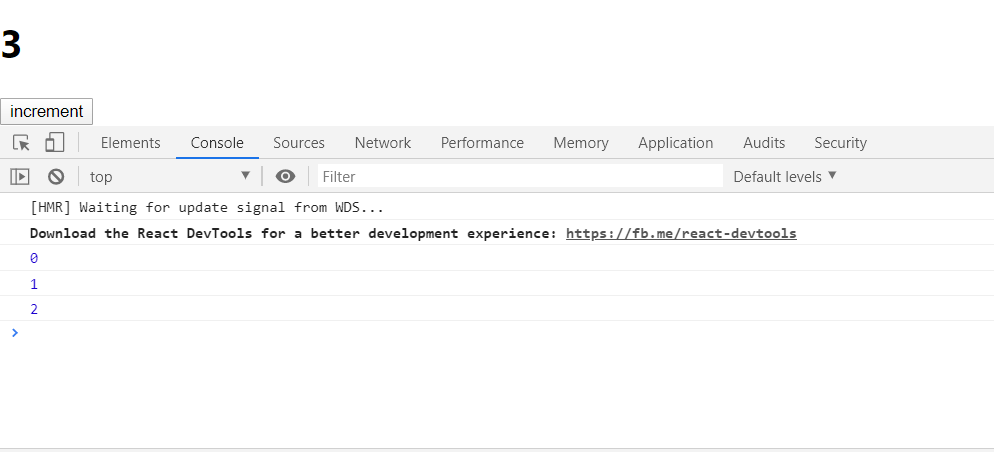
:

:

console.log(this.state.count)

}

In output



To Rectify this we call this as an CALLBACK function to setState Method

increment(){

this.setState({

count:this.state.count + 1

},() => {console.log(this.state.count)})

}

**CALLING PREVSTATE**

To implement this lets do the chaining of the event by calling the increment function 5 times .

import React,{Component} from 'react'

class Counter extends Component

{

constructor(){

super()

this.state={

count:0

}

}

increment(){

this.setState({

count:this.state.count + 1

},() => { console.log(this.state.count)

})

}

incrementFive(){

this.increment()

this.increment()

this.increment()

this.increment()

this.increment()

}

render(){

return(

<div>

<h1>{this.state.count}</h1>

<button onClick={() => this.incrementFive()}>increment</button>

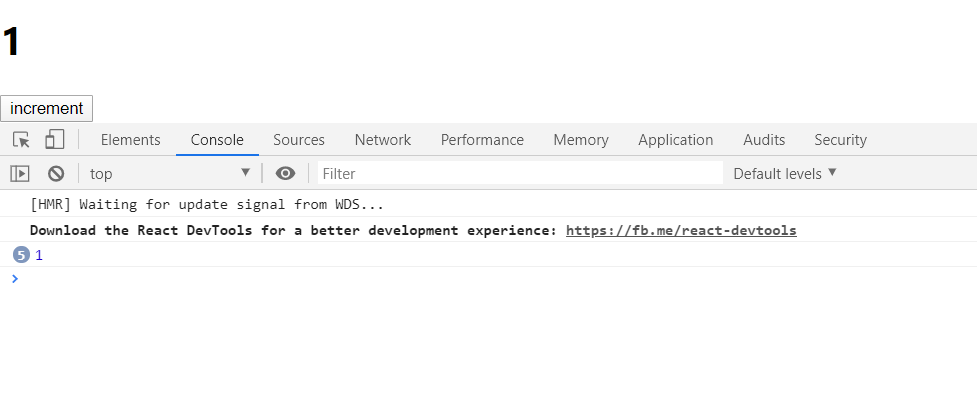
</div>)

}

}

export default Counter

Output



This means that on clicking only 1 time , count is increment by 1 as for all the count is grouped into 1 only.

Thereby we need to CHAIN it with previous state .

*this.setState({*

*count:this.state.count + 1*

*},() => {console.log(this.state.count)})*

Is changed to

increment(){

this.setState(prevState => ({

count:prevState.count + 1

}),() => {console.log(this.state.count)})

}

React :: LifeCycle Methods

Why We need to Bind ?

In Event , ***this*** is a Javascript keyword and thereby it remains undefined

Lets try this with an example

**EventBind.js**

import React,{Component} from 'react'

class EventBind extends Component

{

constructor(){

super()

this.state={

message:"Hello"

}

this.clickHandler = this.clickHandler.bind(this)

}

clickHandler(){

this.setState=({

message:"GoodBye"

})

console.log(this)

}

render(){

return(

<div>

<h1>{this.state.message}</h1>

<button onClick={this.clickHandler}>Click Me</button>

{/\*Option 1\*/}

<button onClick={this.clickHandler.bind(this)}>Click Me</button>

{/\*Option 2\*/}

<button onClick={() => this.clickHandler ()}>Click Me</button>

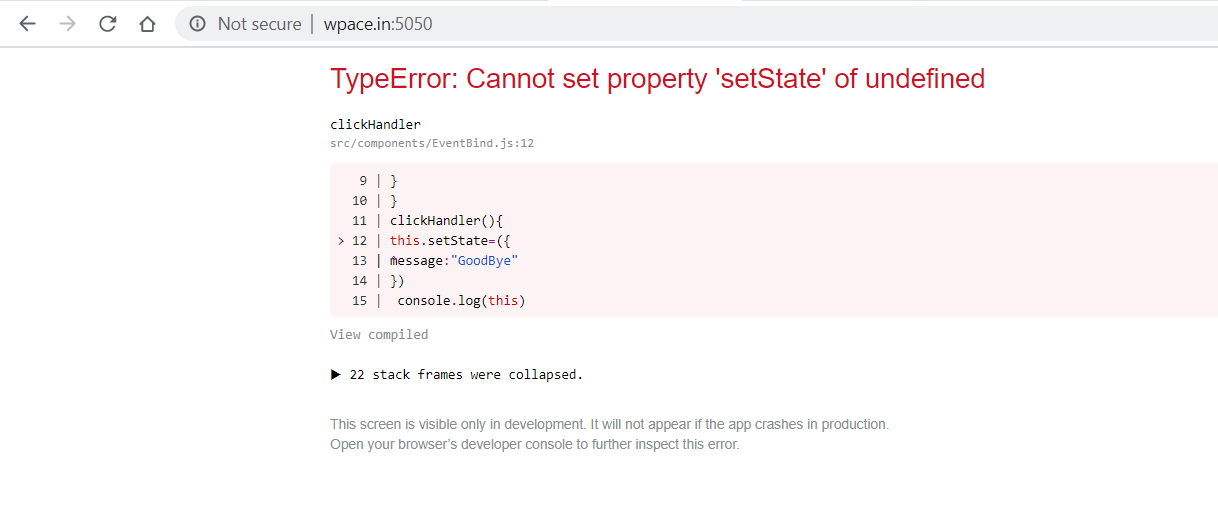
</div>)

}

}

export default EventBind

Output



It has two Options

1. Inside the render method with
   1. **<button onClick={this.clickHandler.bind(this)}>Click Me</button>**
2. Inside the render Method
   1. **<button onClick={() => this.clickHandler ()}>Click Me</button>**

Both of the Above methods are in render function and have issues with the performance of the application

**So the best option has to be Bind Inside the Constructor**

Putting the below in the constructor

this.clickHandler = this.clickHandler.bind(this)

Render method will have simply

<button onClick={this.clickHandler}>Click Me</button>

Final the Most efficient way is to change the ***clickHandler method to the Arrow Function***

import React,{Component} from 'react'

class EventBind extends Component

{

constructor(){

super()

this.state={

message:"Hello"

}

// this.clickHandler = this.clickHandler.bind(this)

}

//clickHandler(){

//this.setState({

// message:"GoodBye"

// })

// console.log(this)

// }

clickHandler = () =>{

this.setState({

message:"GoodBye"

})

console.log(this)

}

render(){

return(

<div>

<h1>{this.state.message}</h1>

<button onClick={this.clickHandler}>Click Me</button>

{/\*<button onClick={this.clickHandler.bind(this)}>Click Me</button> \*/}

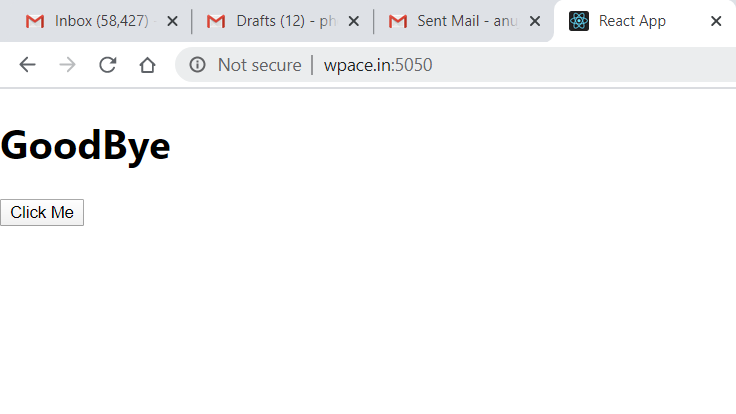
{/\*<button onClick={() => this.clickHandler()}>Click Me</button> \*/}

</div>)

}

}

export default EventBind



React :: LifeCycle Methods

Here the List is pretty simple by using the Array.map method

Lets do this by building a new functional component

**NameList.js**

import React from 'react'

import NameItem from './NameItem'

function NameList(){

const nameList = [{'Name':'Anuj','Age':'43','Gender':'Male'},{'Name':'Sonal','Age':'41','Gender':'Female'},{'Name':'Ayaan','Age':'11','Gender':'Male'}];

//const nameMap=nameList.map(name => <h1>My name is {name.Name} and my age is {name.Age}</h1>)

const nameMap=nameList.map(name => <NameItem item={name} />)

return(

<div>

{nameMap}

</div>

)

}

export default NameList

Now we have made the NameItem -> which describes the layout of each Item.

**NameItem.js**

import React from 'react'

function NameItem(props)

{

return(

<div>

Hi, your name is <h1>{props.item.Name}</h1>

Age :{props.item.Age}

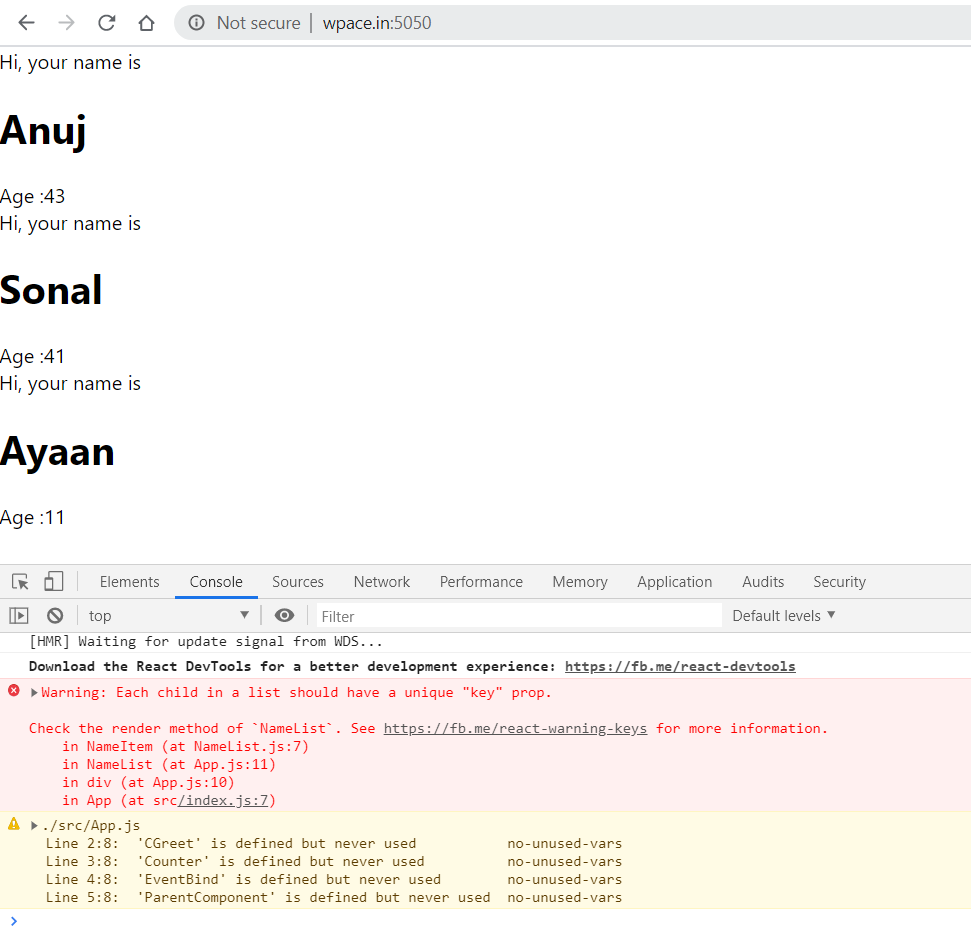
</div>

)

}

export default NameItem

Output :



But in the End there is an error which shows the Key property is not unique

So for this add ‘id’ to an array and change the method to following :-

import React from 'react'

import NameItem from './NameItem'

function NameList(){

const nameList = [{'id':'1','Name':'Anuj','Age':'43','Gender':'Male'},{'id':'2','Name':'Sonal','Age':'41','Gender':'Female'},{'id':'3','Name':'Ayaan','Age':'11','Gender':'Male'}];

//const nameMap=nameList.map(name => <h1>My name is {name.Name} and my age is {name.Age}</h1>)

const nameMap=nameList.map(name => <NameItem key={name.id} item={name} />)

return(

<div>

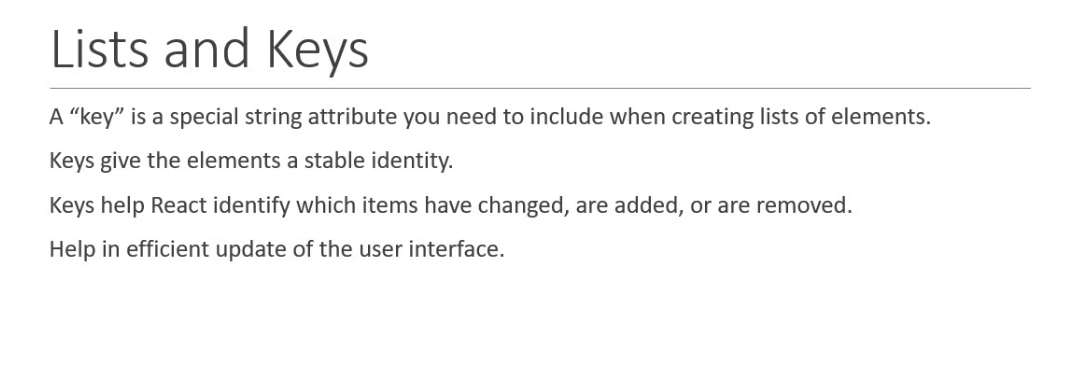
{nameMap}

</div>

)

}

Export default NameList



React :: LifeCycle Methods

Example of using the Form handling methods .

import React,{Component} from 'react'

class Form extends Component

{

constructor(){

super();

this.state={

username:'',

topic:'',

comments:''

}

}

handleUsernameChange = event => {

this.setState({

username:event.target.value

})

}

handleCommentsChange = event => {

this.setState({

comments:event.target.value

})

}

handleTopicChange = event => {

this.setState({

topic:event.target.value

})

}

handleSubmitEvent = event =>{

alert(`${this.state.username} ${this.state.comments} ${this.state.topic}`)

event.preventDefault()

}

render(){

return(

<form onSubmit={this.handleSubmitEvent}>

<div>Username

<input type="text" value={this.state.username} onChange={this.handleUsernameChange} />

</div>

<div>Comments

<textarea value={this.state.comments} onChange={this.handleCommentsChange} />

</div>

<div> Select the Topic

<select value={this.state.topic} onChange={this.handleTopicChange} >

<option value="react">React</option>

<option value="angular">Angular</option>

<option value="vue">Vue</option>

</select>

</div>

<div>

<input type="submit" value="Submit" />

</div>

</form>

)

}

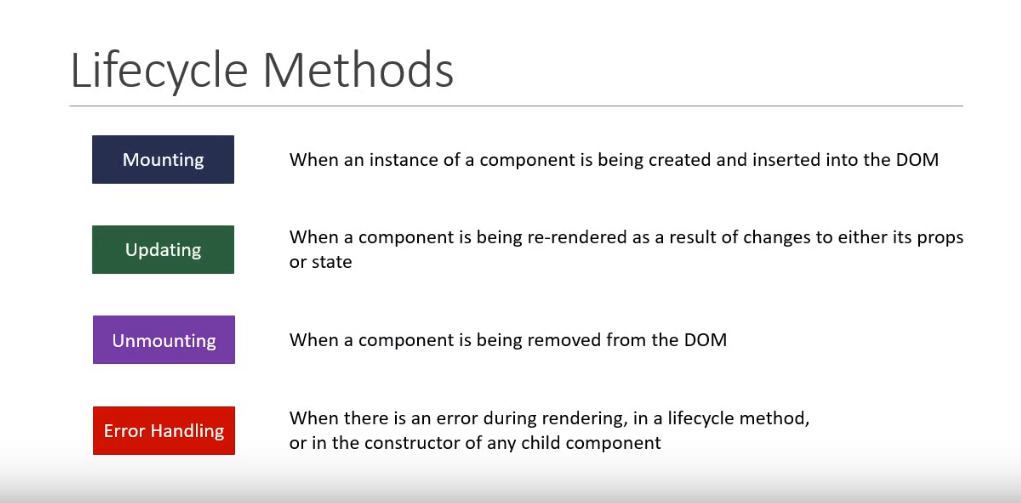
}

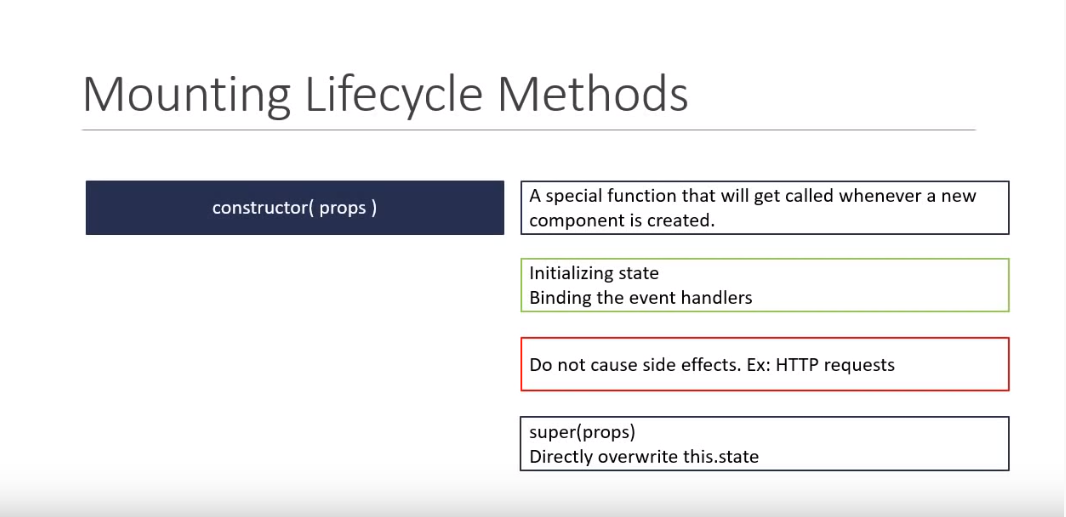
export default Form

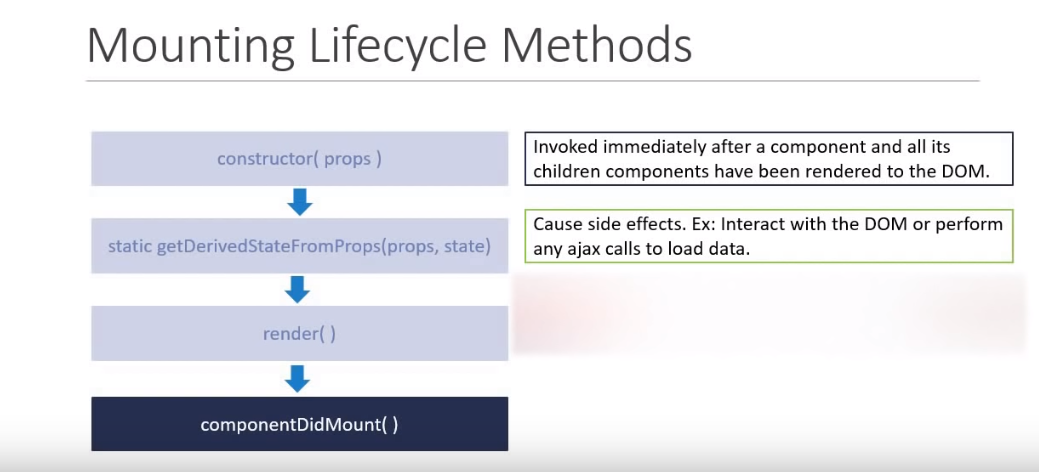
React :: LifeCycle Methods

Lifecycle method are the important part of the react which changes with the component being insetrted into the DOM(ie Mounting) Having the change of Data or the state via the State change methods and Unmounting means removal of the component from the DOM .

Then there is an error handling for it to Error Handling method.







Lets look at the code for different Methods

import React,{Component} from 'react'

class LifeCycleA extends Component

{

constructor(props){

super(props)

this.state={

name:'Anuj'

}

console.log("LifeCycleA Constructor called")

}

static getDerivedStateFromProps(props,state)

{

console.log("LifeCycleA getDerivedStateFromProps called")

return null

}

componentDidMount(){

console.log("LifeCycleA component Did Mount called")

}

render(){

console.log("LifeCycleA render called")

return(

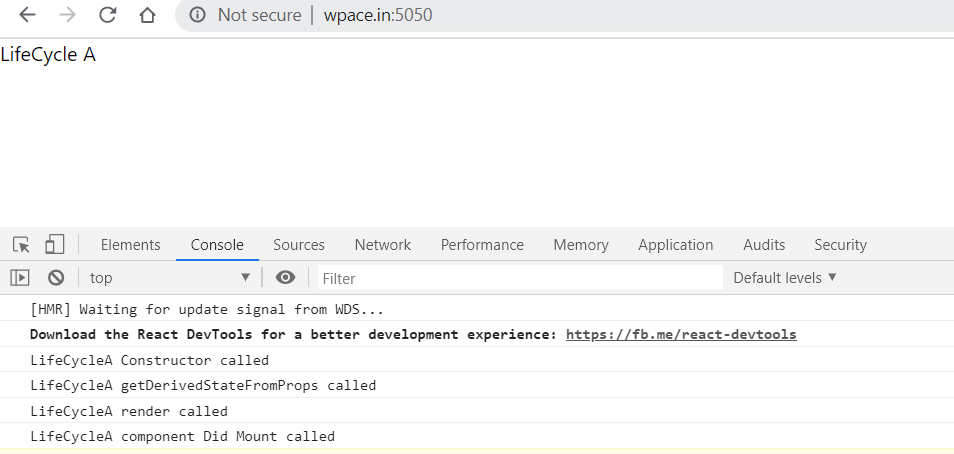
<div>LifeCycle A</div>

)

}

}

export default LifeCycleA



Now When we include another child component by duplicating the same file and including it in the LifeCycleA file

Render method is changed to

render(){

console.log("LifeCycleA render called")

return(

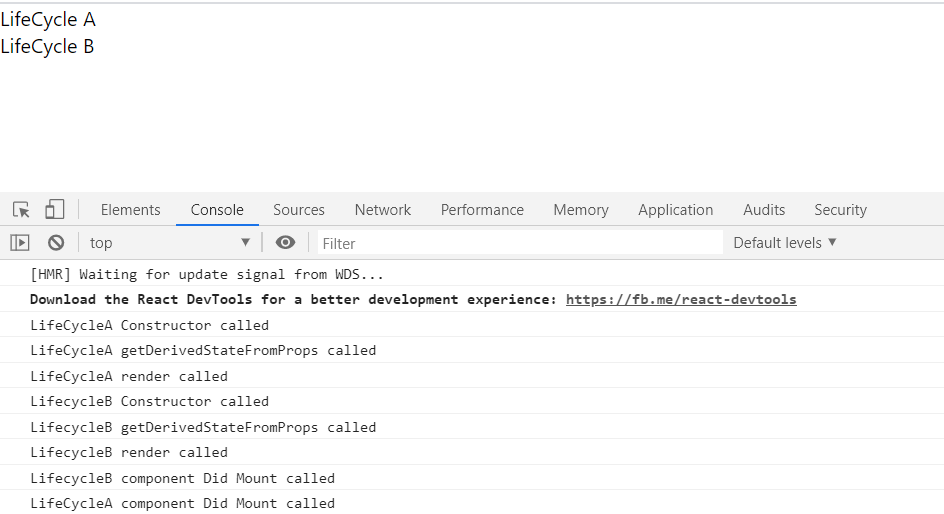
<div>LifeCycle A

<LifeCycleB />

</div>

)

}



Now Try Updating the Update Method

import React,{Component} from 'react'

import LifeCycleB from './LifeCycleB'

class LifeCycleA extends Component

{

constructor(props){

super(props)

this.state={

name:'Anuj'

}

console.log("LifeCycleA Constructor called")

}

static getDerivedStateFromProps(props,state)

{

console.log("LifeCycleA getDerivedStateFromProps called")

return null

}

componentDidMount(){

console.log("LifeCycleA component Did Mount called")

}

shouldComponentUpdate()

{

console.log("LifeCycleA Should component Update called")

return true

}

getSnapshotBeforeUpdate()

{

console.log("LifeCycleA get Snapshot Before Update called")

return null

}

componentDidUpdate(){

console.log("LifeCycleA component Did Update called")

return true

}

changeState = () => {

this.setState({

name:'CodeVolution'

})

}

render(){

console.log("LifeCycleA render called")

return(

<div>LifeCycle A

<LifeCycleB />

<button onClick={this.changeState}>Change State</button>

</div>

)

}

}

export default LifeCycleA

