v-01 Introduction to REACT JS

Install node and yarn

Download node from site and next next

npm install --global yarn

v-02 (createAPP)

npx create-react-app first-app00

cd first-app00

npm start

v-03 (Files overview)

it will run the server then just go inside of the folder and open cmd:  
code .

if we remove node\_modules folder we can install it again by just typing : npm install and everything again installed automatically.

npm update will install the latest version

package.json have the dependencies and run command things.

Package-lock.json if we give another one drive to our repo then it needs to install all the dependencies to the specific order of the previous version itself.

Then = > we have gitignore which ignore unnecessary files like node module which we can create by writing npm install.

Then we have 2 more folder:

Src: is the main project folder

And Public: is the traditional html website.

Manifest included on the html file and it’s the progressive web app or PWA.

So first we go to the public indext.html there we noticed that it linked to the src > index.js and it also linked to the src > app.js

To get start the server : npm start

v-04 (Components)

we can declare our own component :  
import './App.css';

*// to create a different component*

function OurText(){

  return <p>This is our text</p>

}

function App() {

  return (

    <div className="App">

      <header className="App-header">

        <p>

          Hello from HEAVEN

        </p>

        {*/\* lets use our component here \*/*}

        <OurText />

      </header>

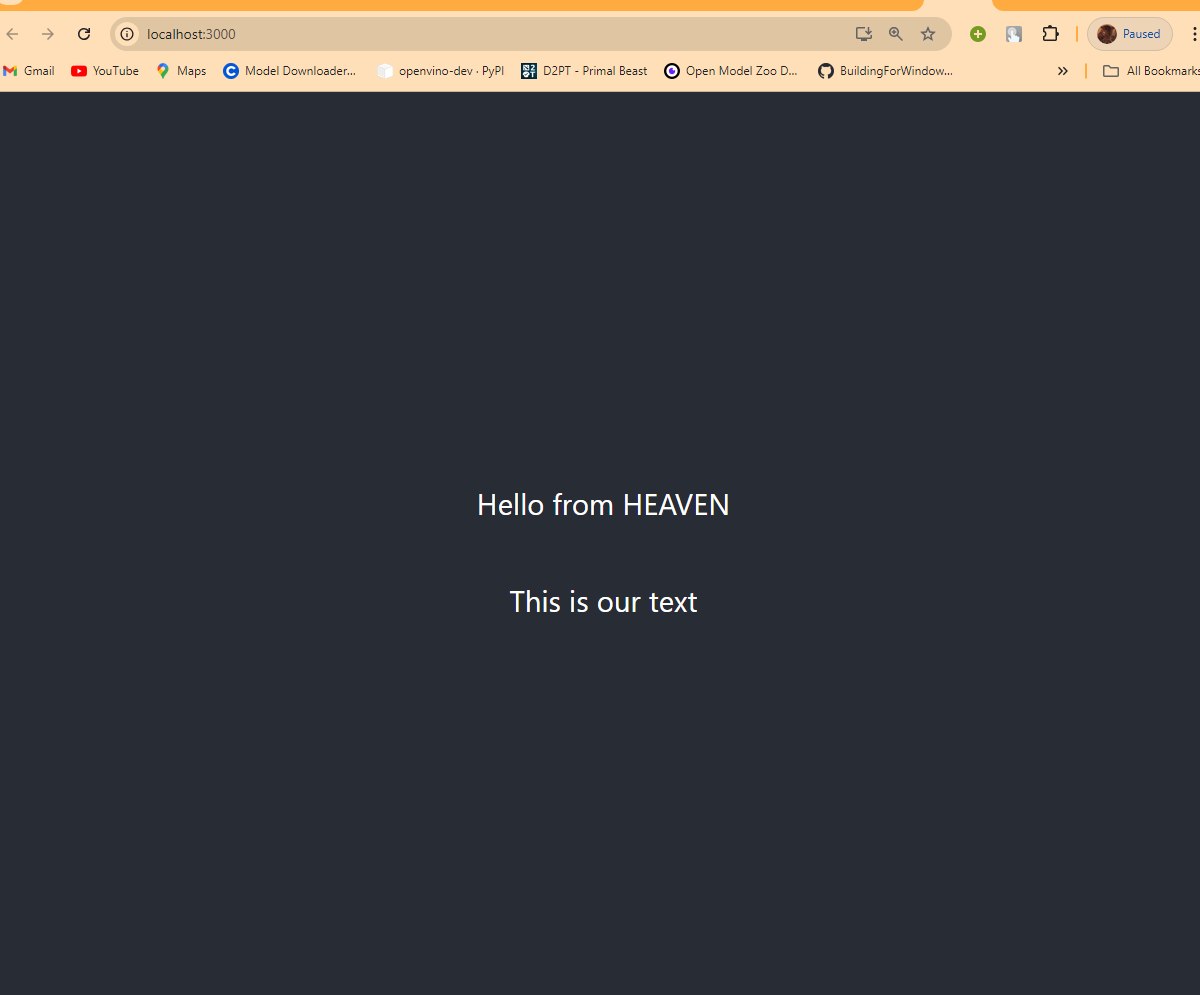
    </div>

  );

}

export default App;

it will show the below text:



Any function inside of our react is COMPONENT.

So we have two different ways of create a component. First just create a new folder called components inside of src folder.

Then open the header.js

Alright so lets write a function inside of   
header.js

import React from 'react';

function Header(){

    return <h1>This is HEADER section</h1>

}

export { Header };

Now edit the app.js file header to initialize :

import React from 'react';

import './App.css';

import { Header } from './components/header'

Now use the header component

import React from 'react';

import './App.css';

import { Header } from './components/header'

*// to create a different component*

*// function OurText(){*

*//   return <p>This is our text</p>*

*// }*

function App() {

  return (

    <div className="App">

      <Header />

    </div>

  );

}

export default App;

2nd way :

We can also export :

export default Header;

this way is header.js file and then initialize it on the App.js

import Header from './components/header';

this way.

if we have the default and 1 component no need to pass through curly braces.

This is the 2 way of import and export components.

Next way is to create a component is classbased components:

Class based component:

footer.js

import React, { Component } from 'react';

class Footer extends Component{

    render() {

        return <h2>This is our footer</h2>

    }

}

export default Footer;

Update the App.js

import React from 'react';

import './App.css';

*// import { Header } from './components/header'*

import Header from './components/header';

import Footer from './components/footer';

*// to create a different component*

*// function OurText(){*

*//   return <p>This is our text</p>*

*// }*

function App() {

  return (

    <div className="App">

      <Header />

      <p>Main Content</p>

      <Footer />

    </div>

  );

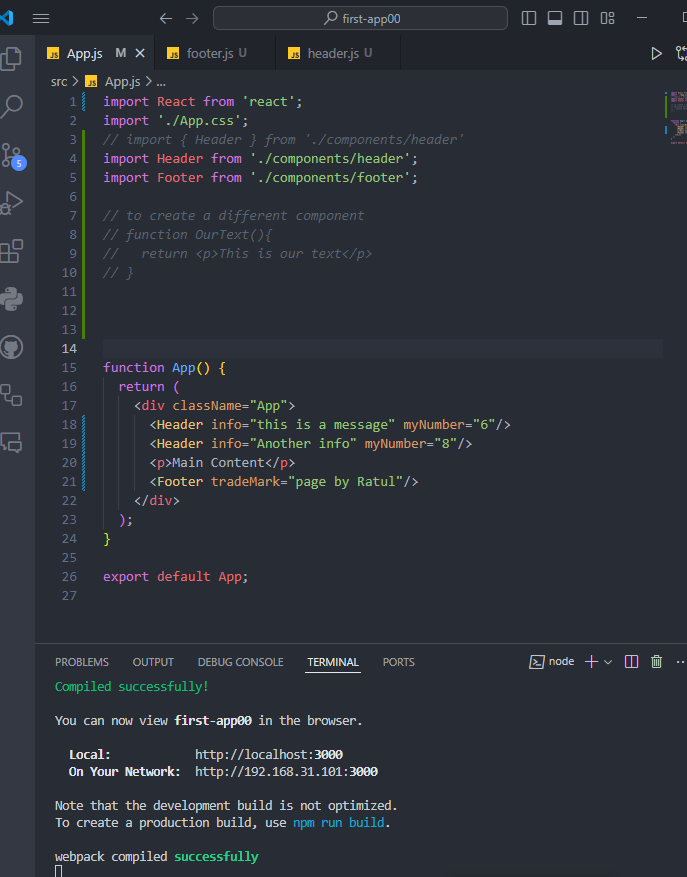
}

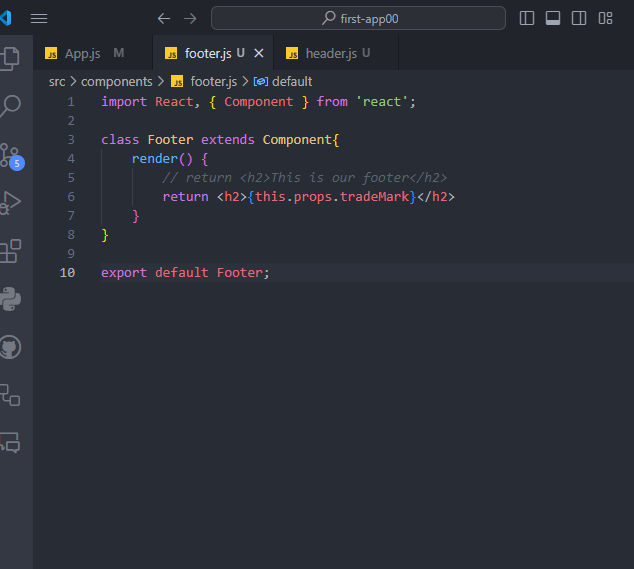
export default App;

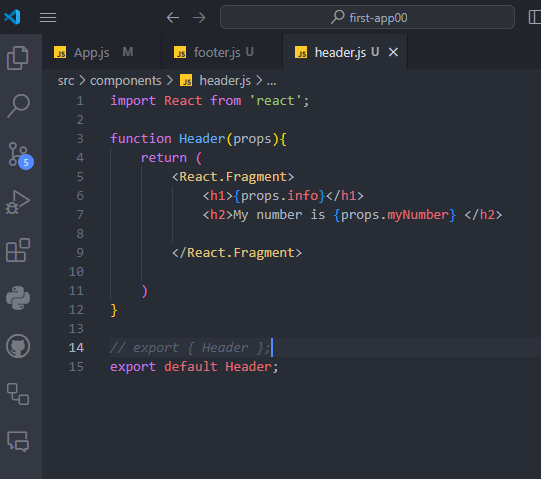
Component use Functional if no complex things want to add. If want then go for classbased because of more complex things related to state.

v-05 (Props)

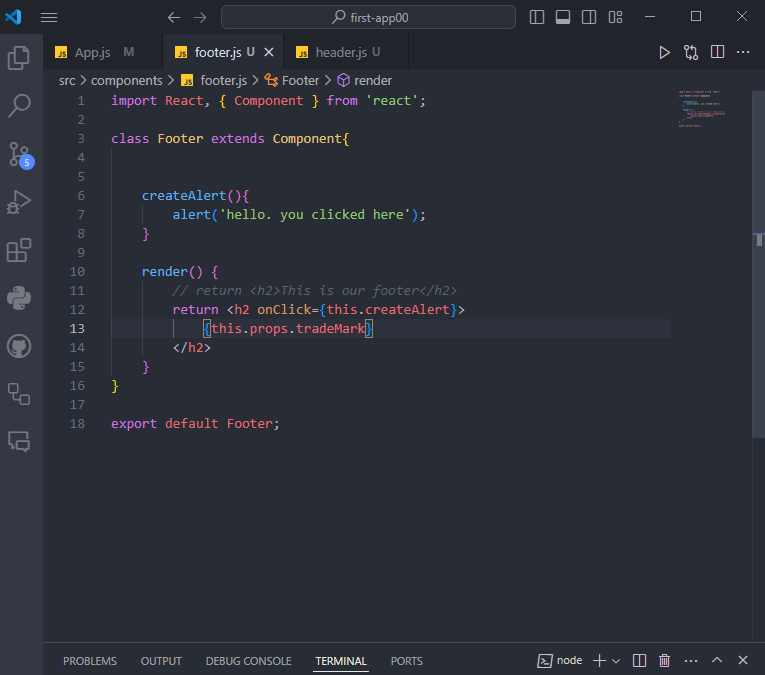
props pass the information to child .

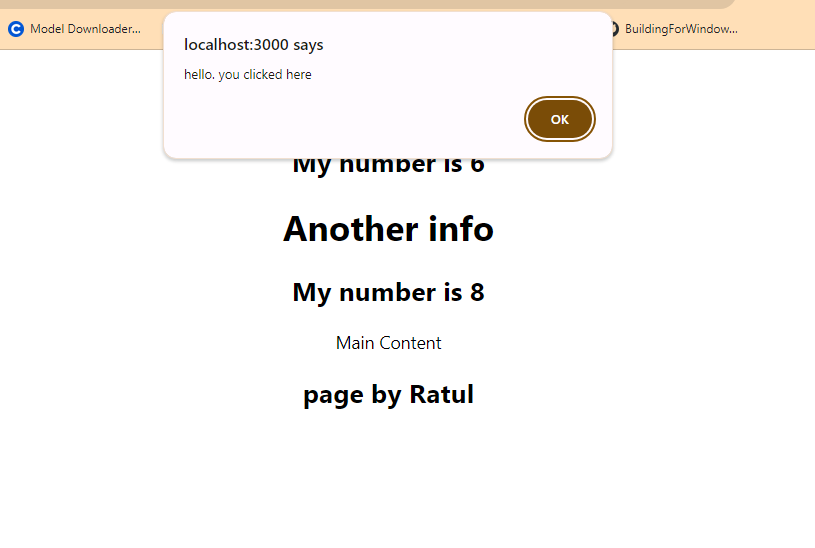




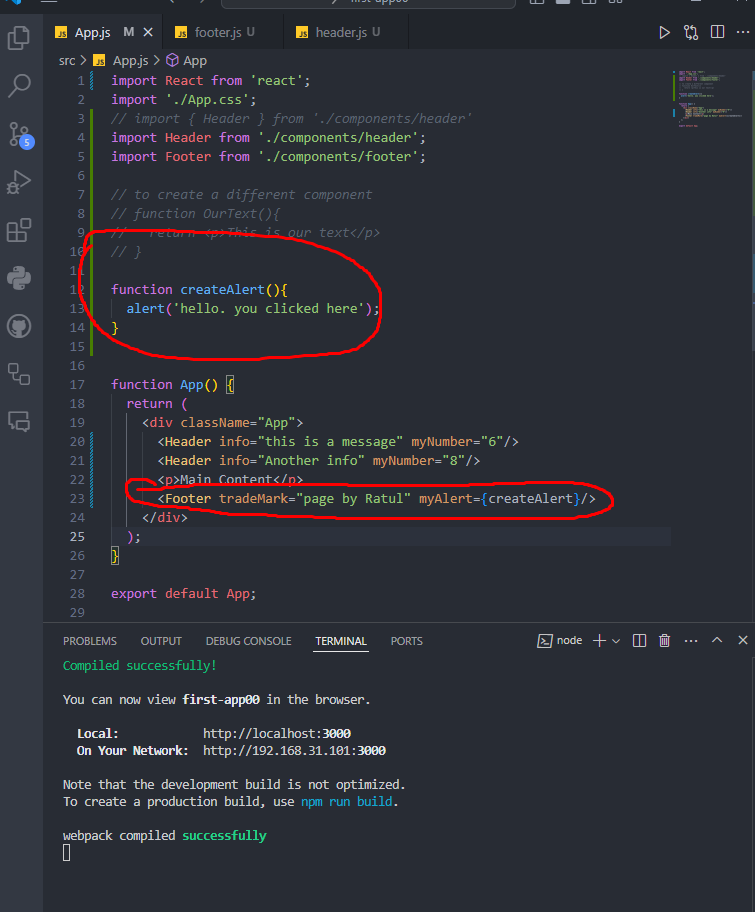


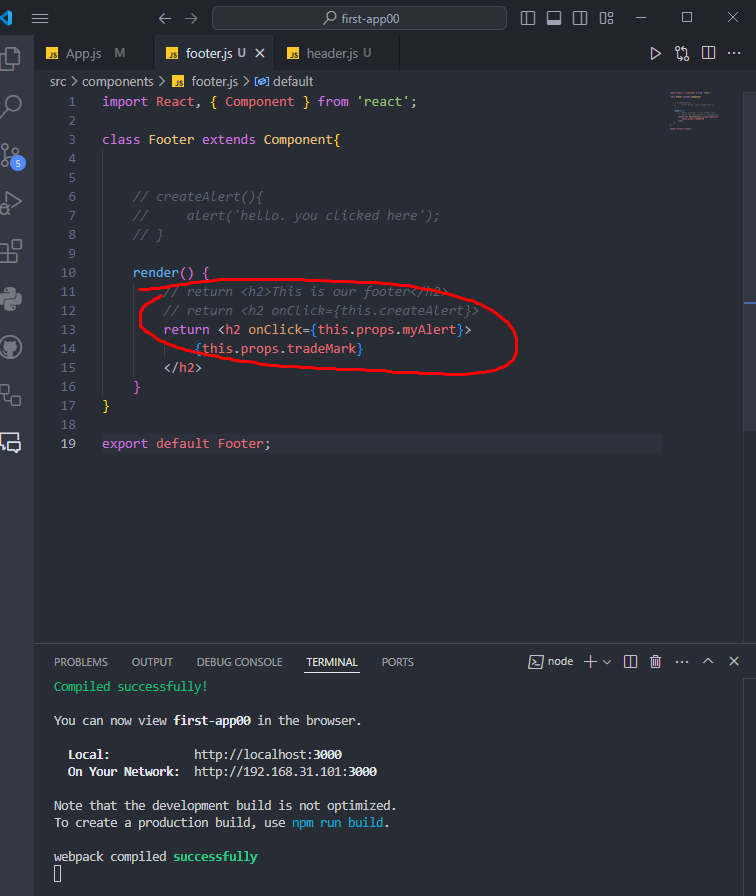
v-06 (Events)



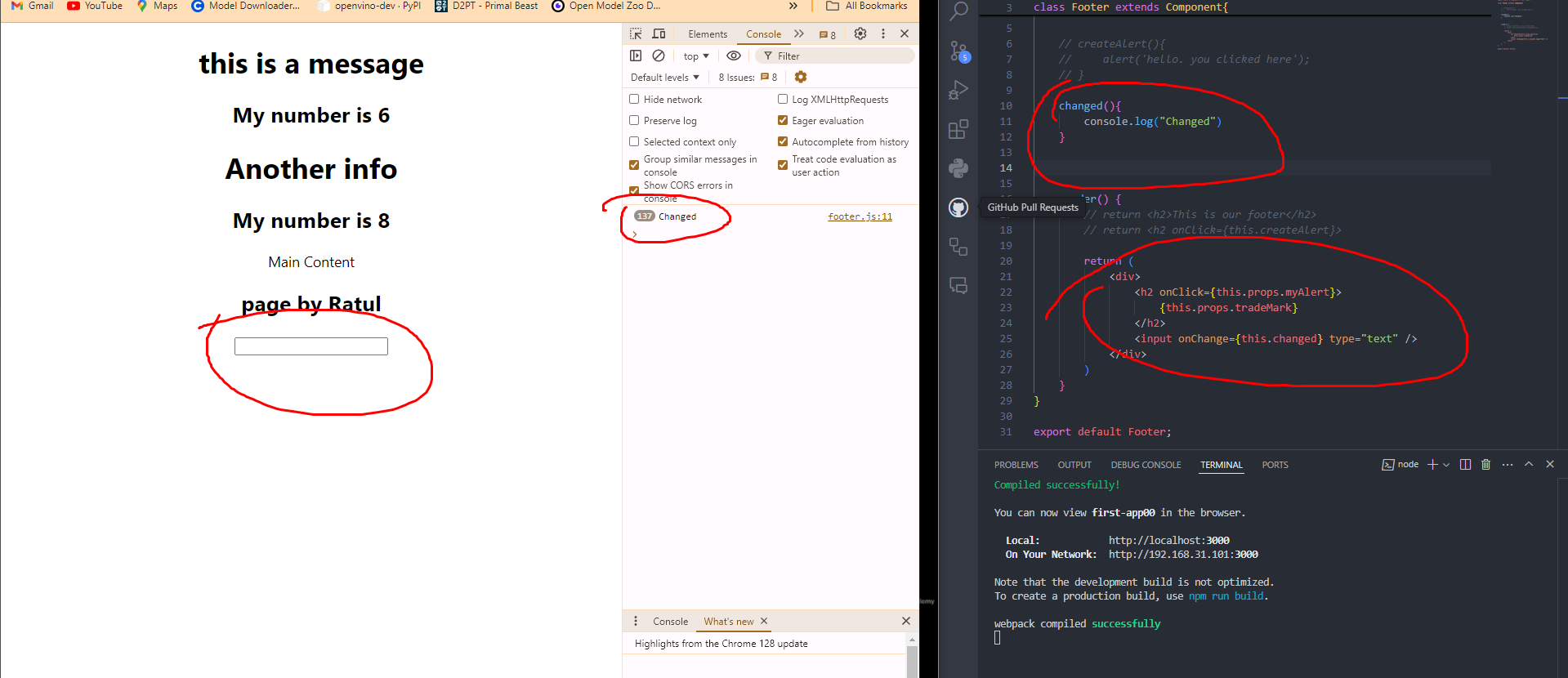


Another way:





Another example of events:



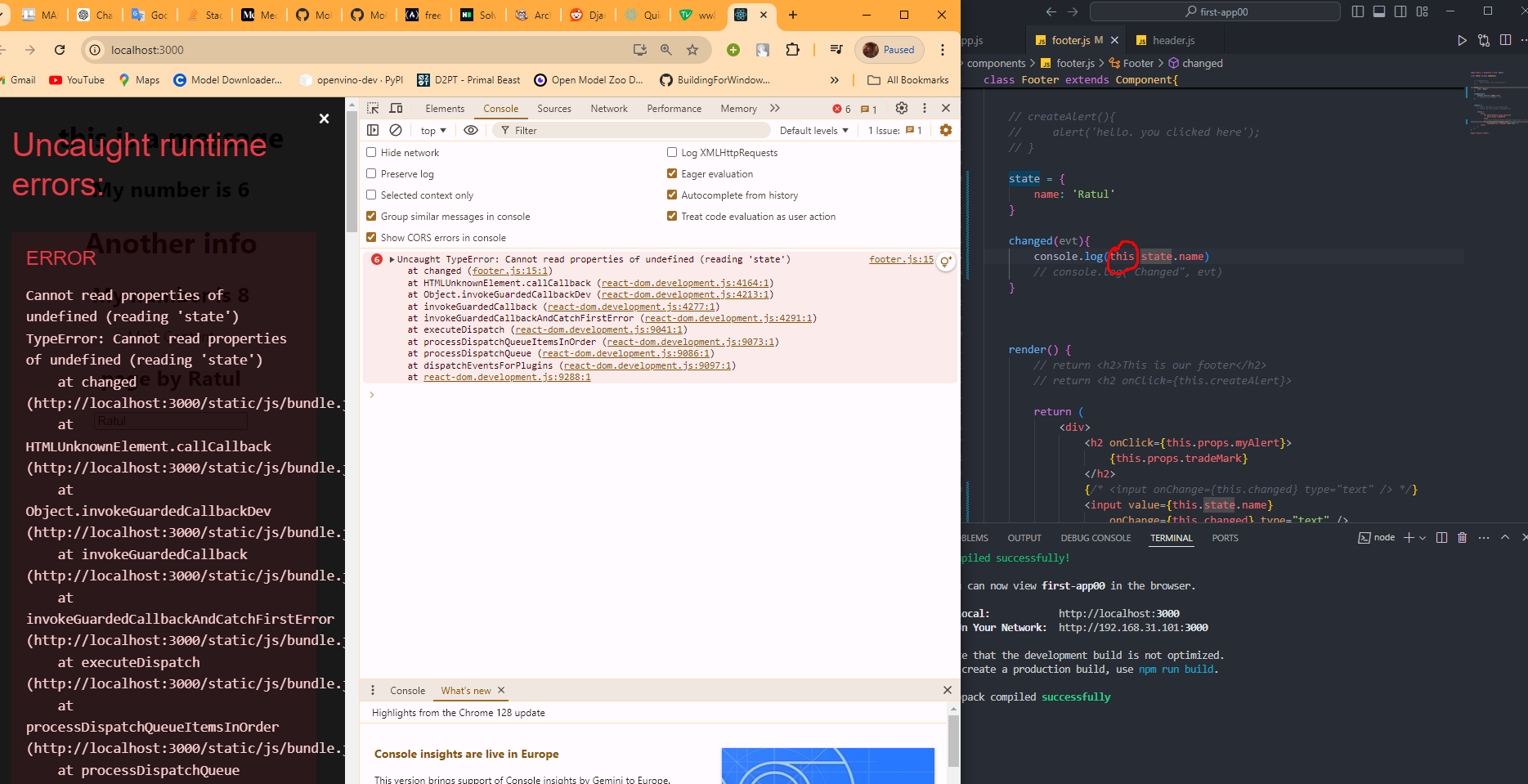
There is a lot of events such as onClick , onChange and hover the mouse after write on……

So 2 way showed classbased and functional event.

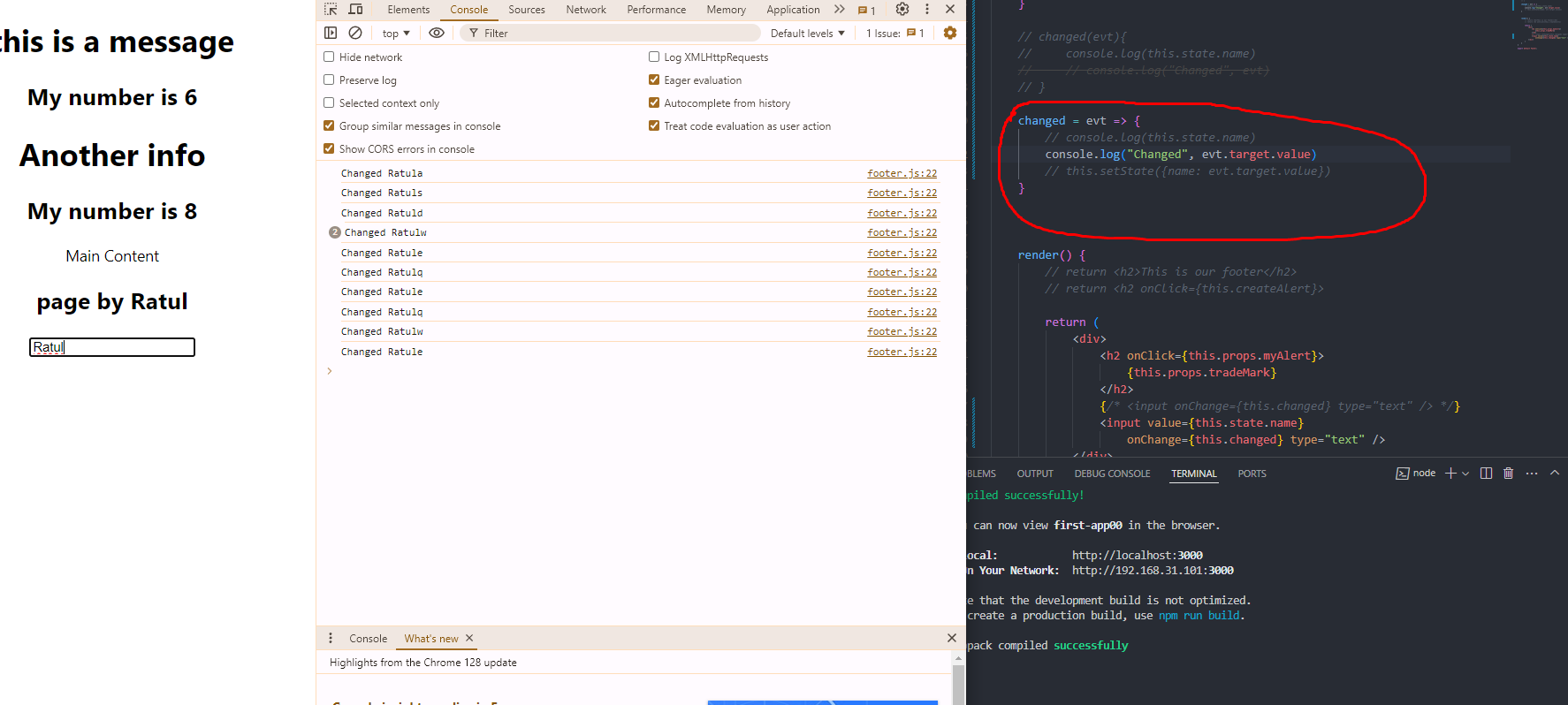
v-07 (State)

if you would like to use state then you need to go for class based.

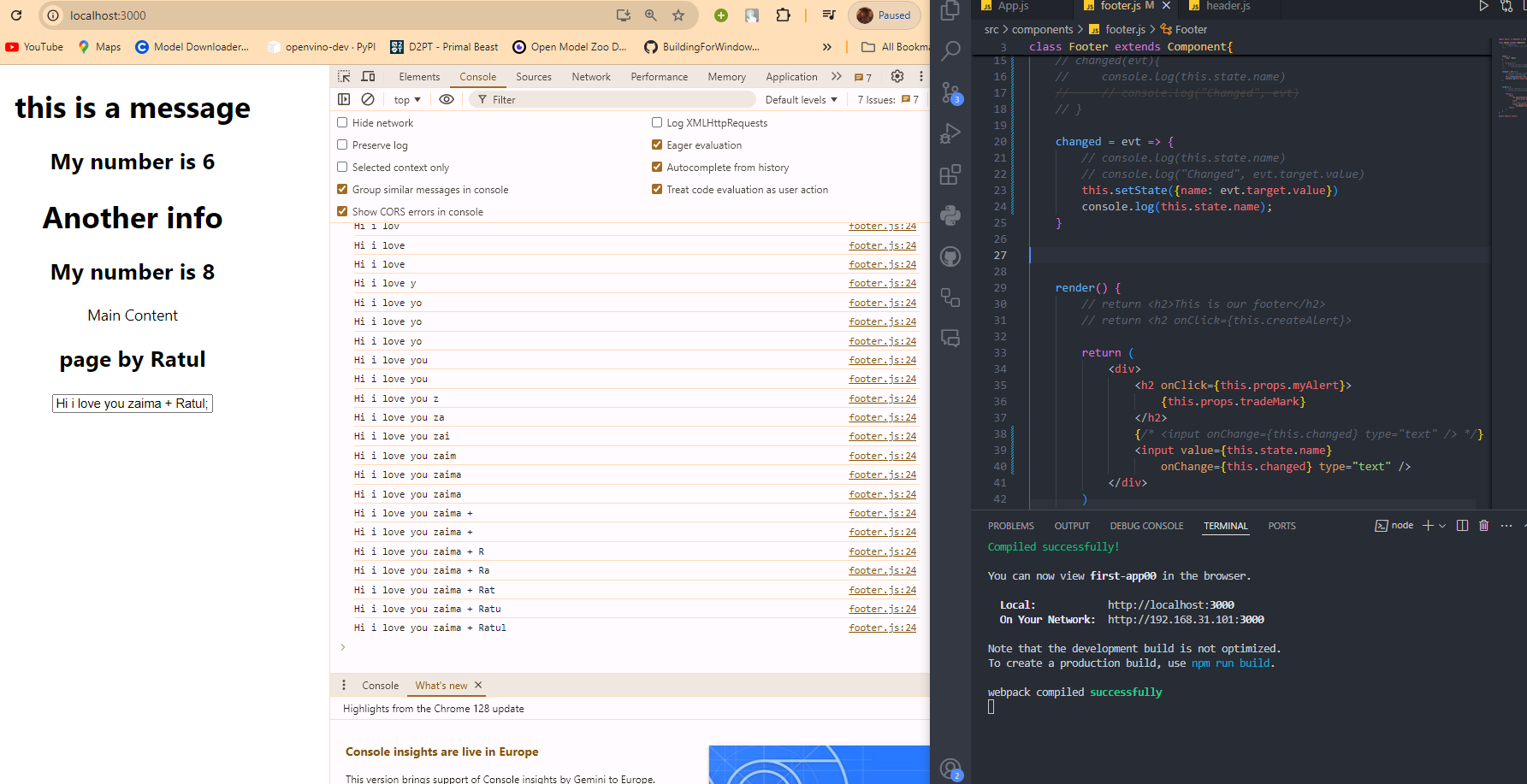
State is used for using like props. But props are immutable. If we want to change the state of those then we should use STATE.

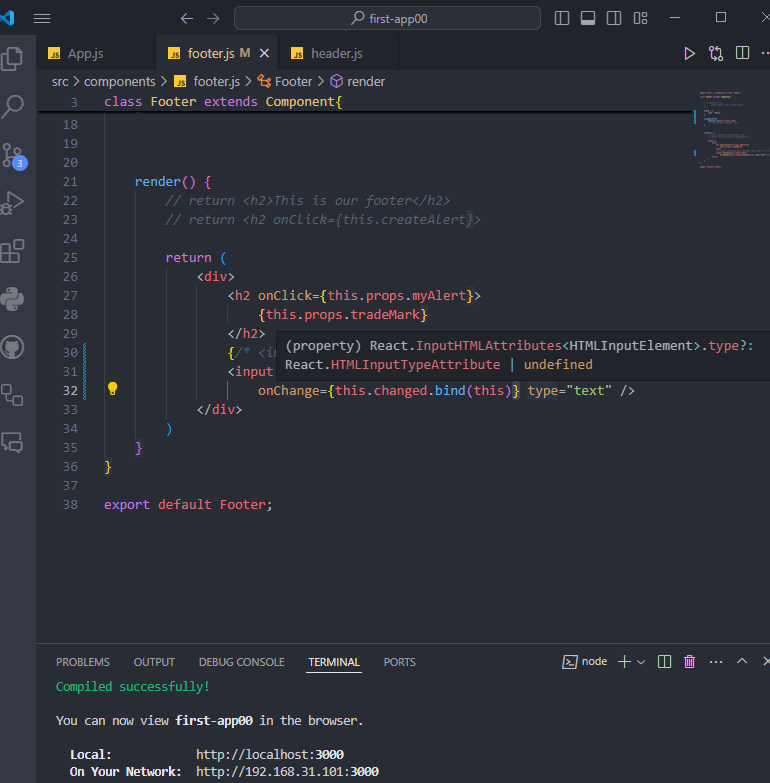


Because (this) is not bound with the class.

When im just using the evt.target.value its not updating the actual value:  


So for updating :

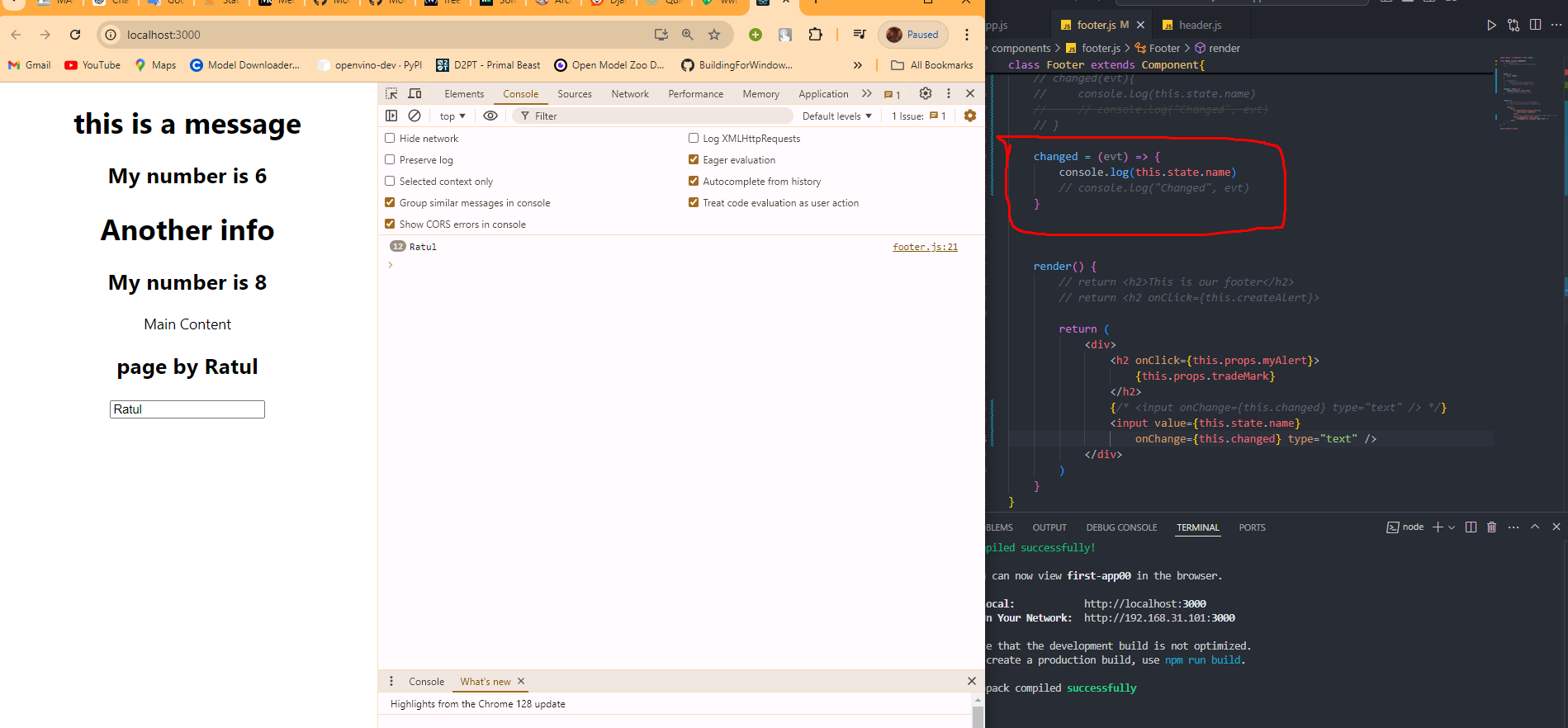


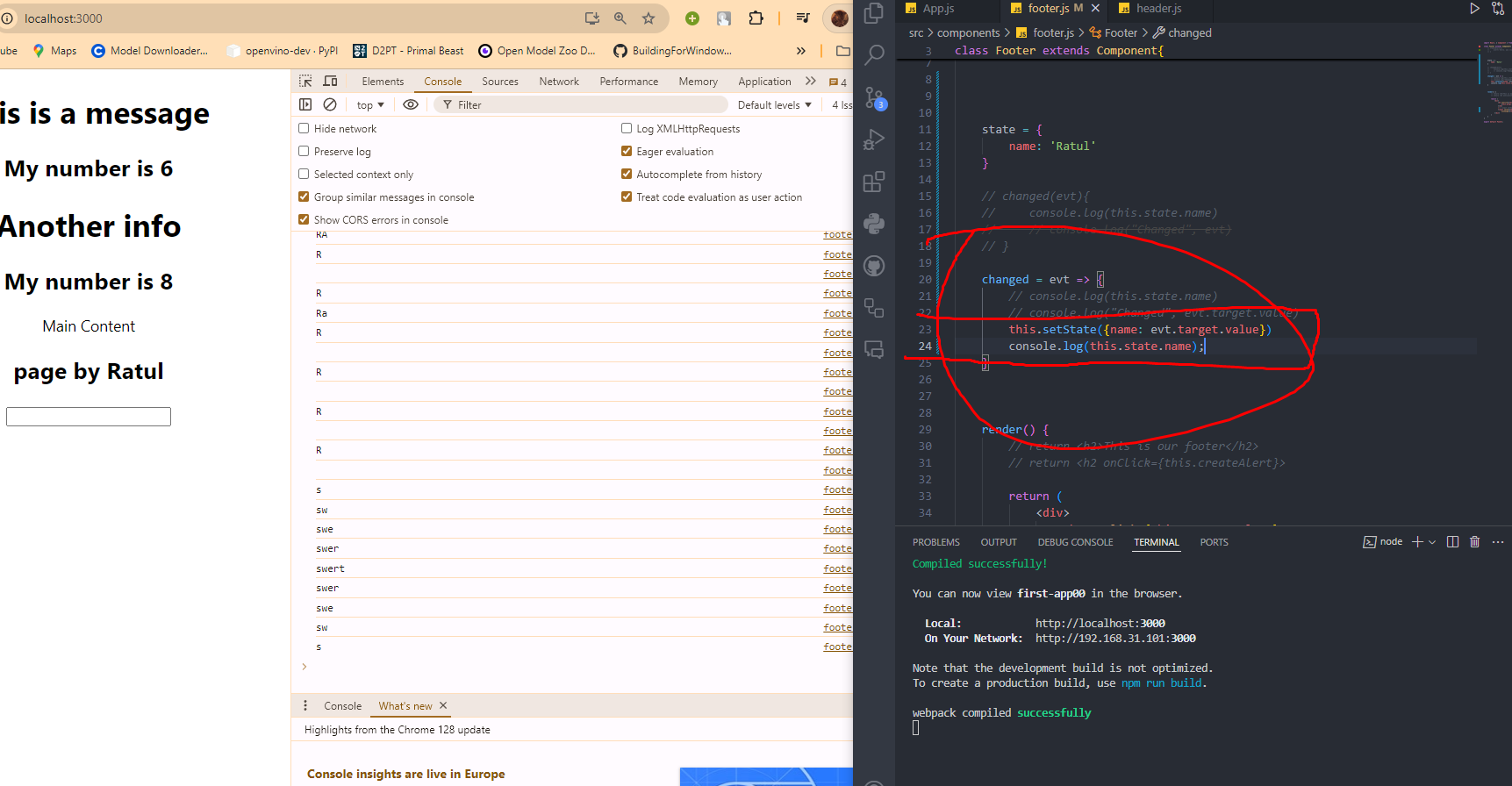


Now It will work but syntax kinda hard.

Contructor is another terrible way to define this.

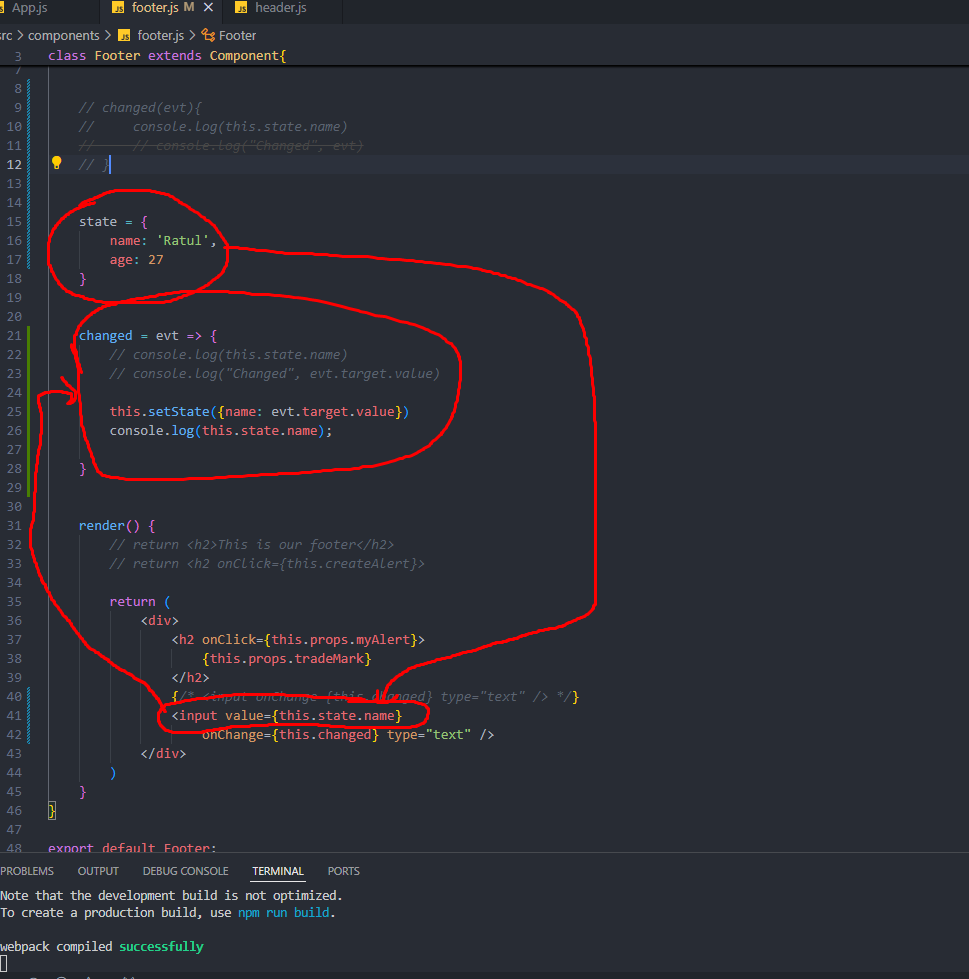
Another function is using arrow function.

Arrow function no need to declare this.bind() 



SO, The state is to hold our data into component and do something about it.

So when we want to change the state of any name like variable:

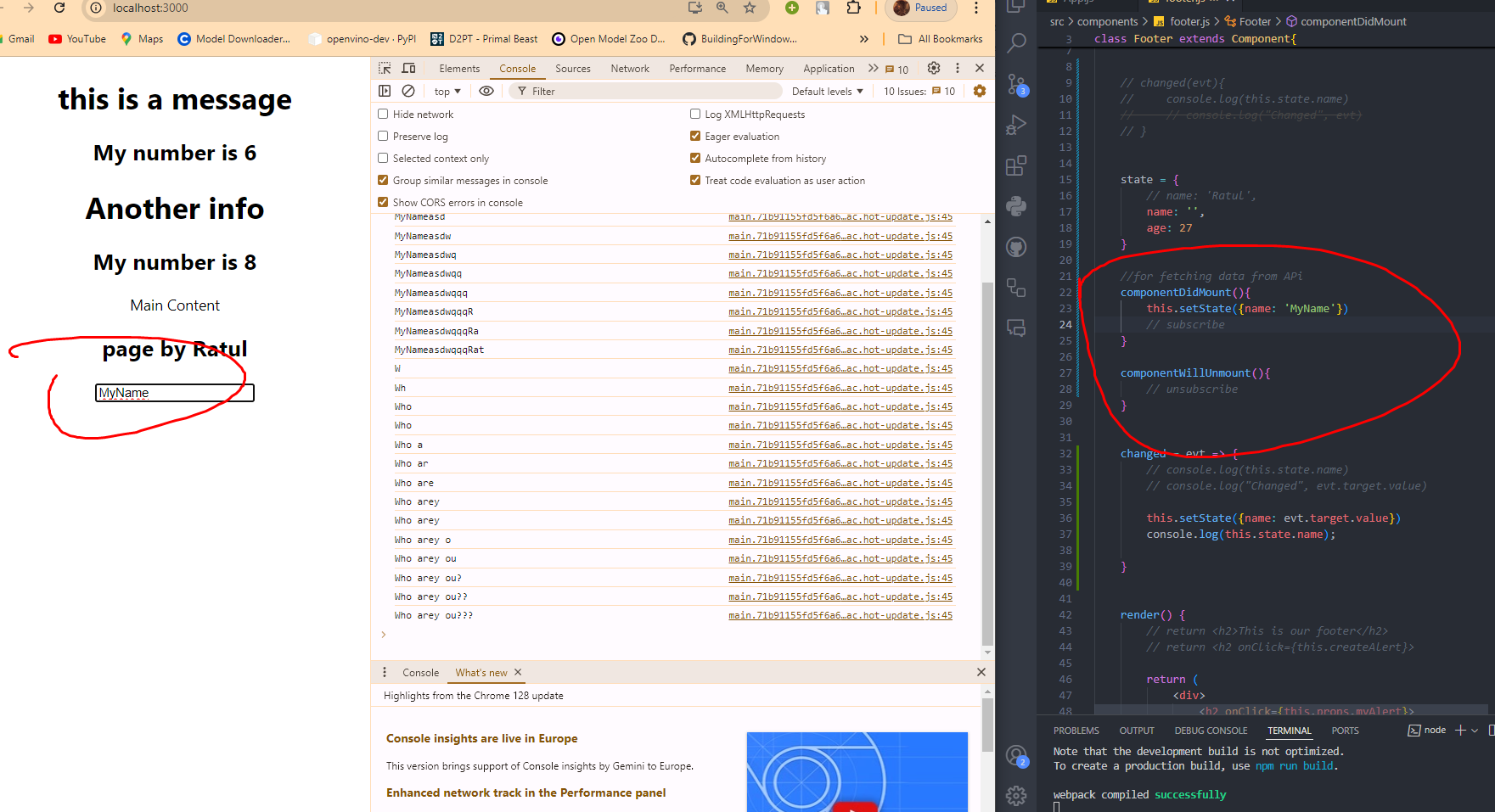


v-08 (Life cycle methods)

Again functional components have no way to use LCM.

When we use something more complex then we have to use classbased one.

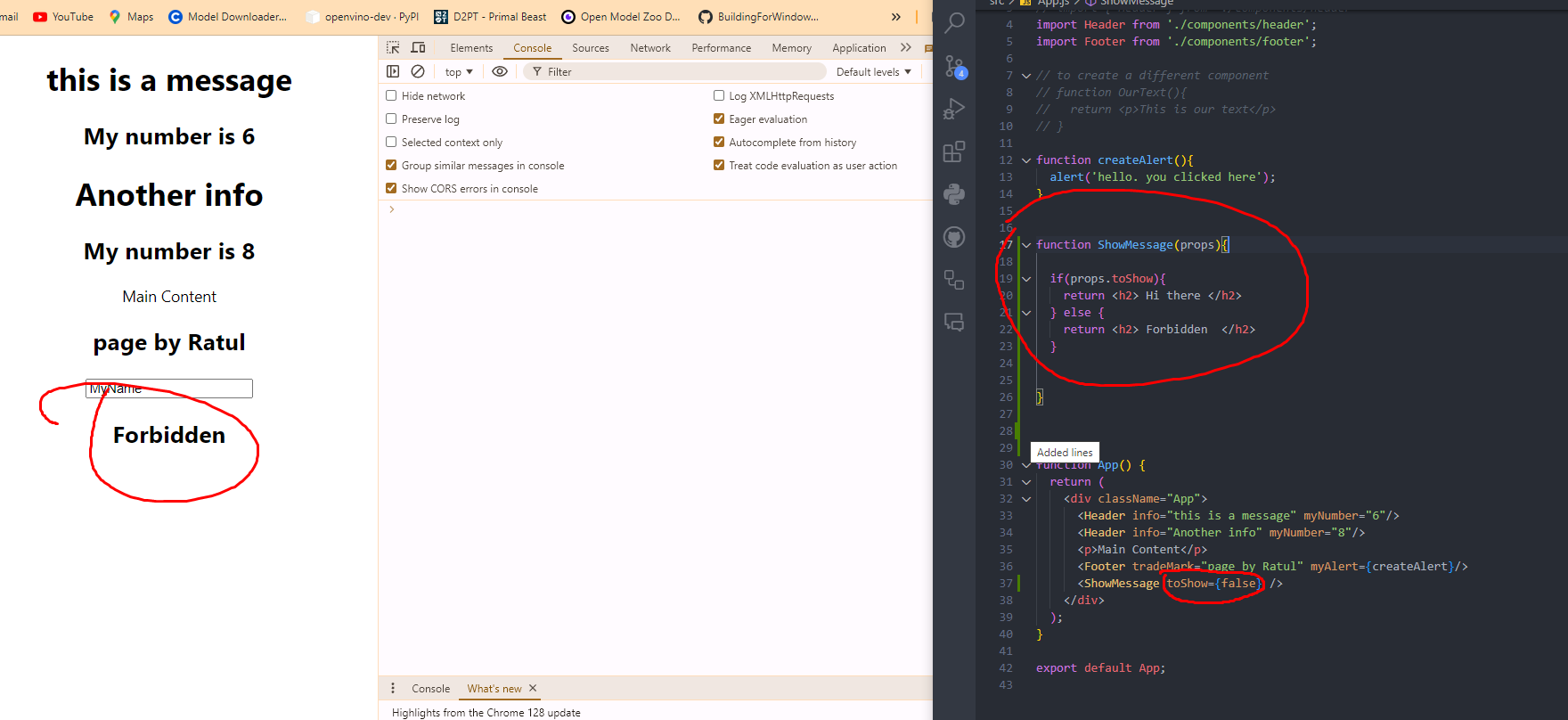
So lets do it in classbased:



Use it whenever its useful.

v-09 (Conditional Display)

we can toggle our components using condition typical if else condition also.



It will work on anything in our page. So suppose we have our App function:

