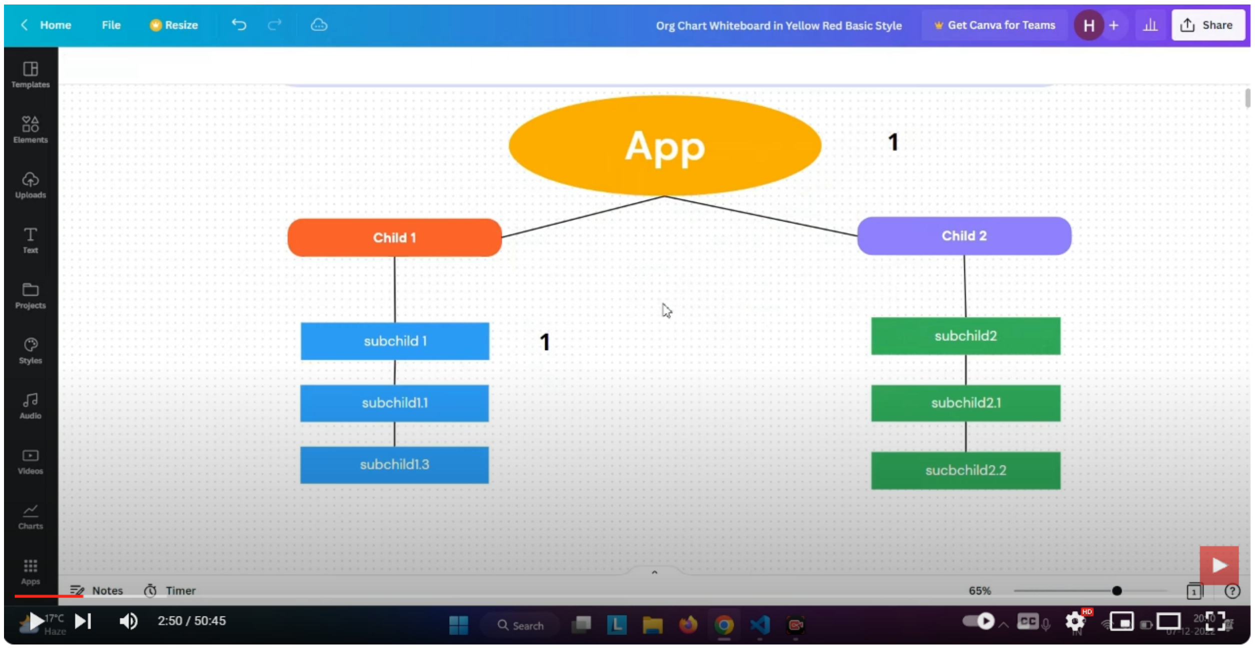
Redux Toolkit Understanding

First we are going to learn that why we are using the Redux

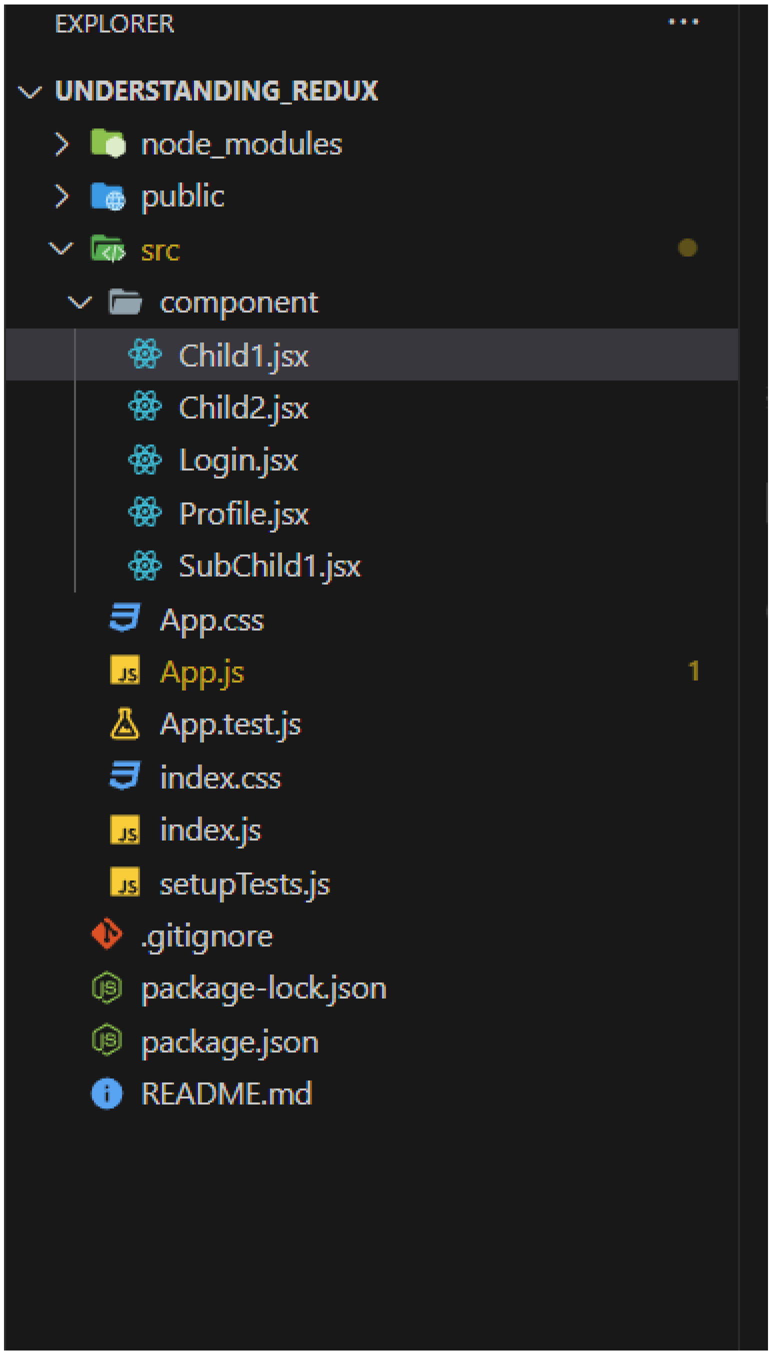


If we have a value 01 in App component and we have to pass it to my subchild 1 from Child 1

App—→child 1——> subchild 1 { we have to pass 01 value than we have to follow this structure }.

Let’s jump to code to understand better

our folder structure


EXPLORER
UNDERSTANDING_REDUX
- node_modules
- public
- src
  - component
    - Child1.jsx
    - Child2.jsx
    - Login.jsx
    - Profile.jsx
    - SubChild1.jsx
  - App.css
  - App.js 1
  - App.test.js
  - index.css
  - index.js
  - setupTests.js
- .gitignore
- package-lock.json
- package.json
- README.md

Code 01 —>App.js

```
import { useState } from 'react';
import './App.css';
import Child1 from './component/Child1'
function App() {

  const [val,setValue]=useState(1)

  return (
    <div className="App">
      <h1>App-{val}</h1>
      <Child1 value={val}/>
    </div>
  );
}

export default App;
```

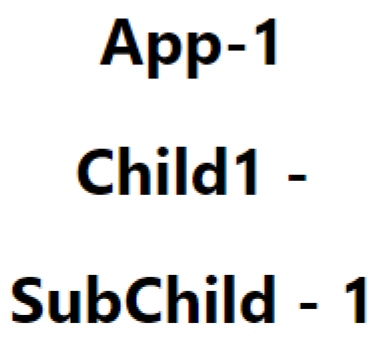
code 02→> child1

```
import React from 'react'
import SubChild1 from './SubChild1'

const Child1 = (props) => {
  return (
    <div>
    <h1>Child1 -</h1>
      <SubChild1 subval={props.value}/>
    </div>
  )
}

export default Child1
```

code 03→subchild1

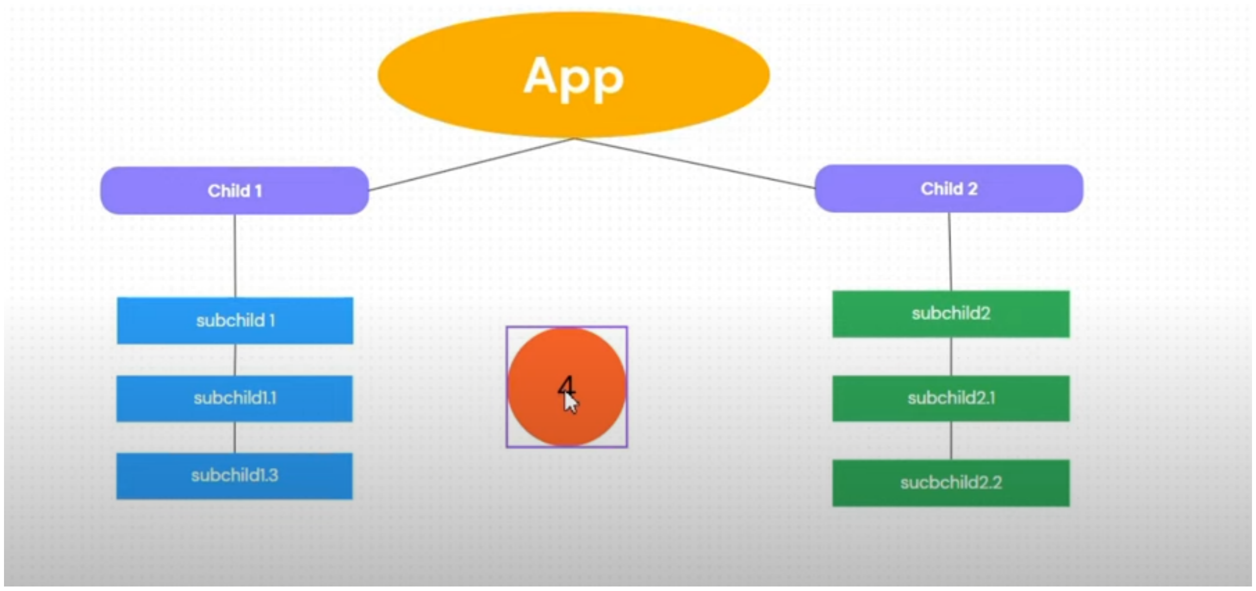
```
import React from 'react'

const SubChild1 = (props) => {
  return (
    <div>
      <h1>SubChils - {props.subval}</h1>
    </div>
  )
}

export default SubChild1
```

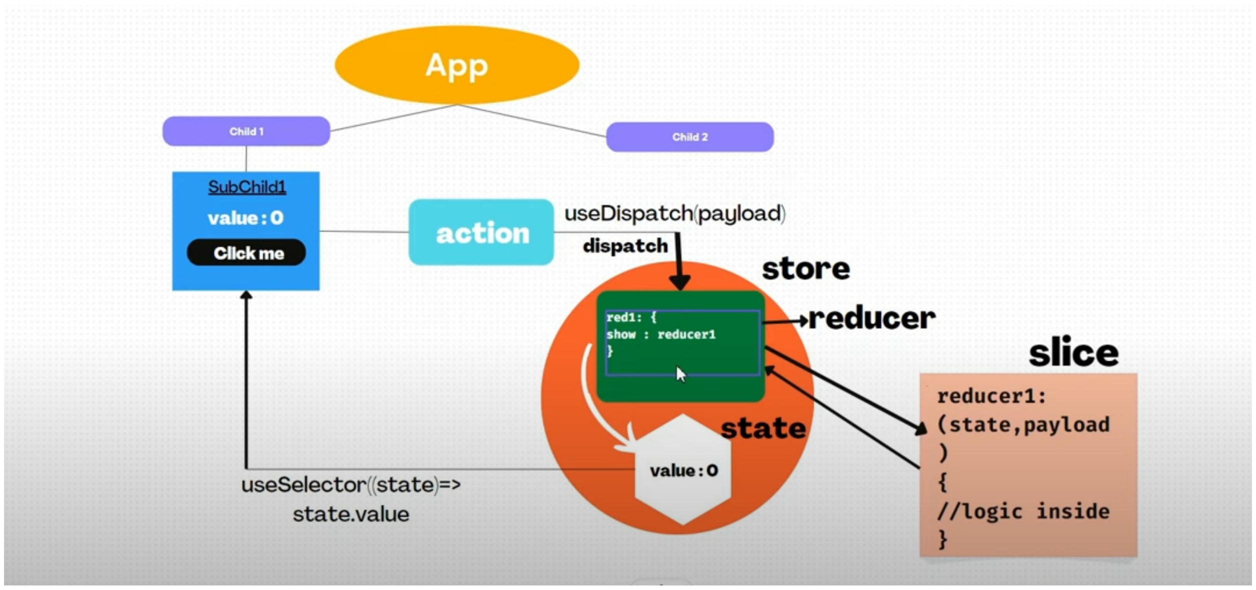
output



```
App-1
Child1 -
SubChild - 1
```

so as we saw if we have to pass our any value from App component to its subChild1 than we have to traverse from staring to end. Suppose if I have subchild99 than we have to traverse it to to 99 times means we have to pass the props from 1 to 99 which is very head-ache work to simply this process we have introduce the **REDUX**





Here we are going to use Redux Toolkit

Why do we need redux ?

Redux allows you to manage your app’s state in a single place and keep changes in your app more predictable and traceable.

Let’s Build an Redux App

Well to make interesting , we will learn redux toolkit along with making a simple counter application.

Let’s start.

Step 1 – Install Redux and Redux Toolkit package in an react app

Thankgod we only need two packages now , so go ahead and install these two.

```
npm install --save react-redux @reduxjs/toolkit
```

Step 2 – Create a global store

Create **src/app/store.js** –

```
import { configureStore } from "@reduxjs/toolkit";

export const store = configureStore({
  reducer: {},
});
```

`configureStore` accepts a single object rather that multiple function arguments. It’s because under the hood, the store has been configured to allow using the Redux DevTools Extension and has had some Redux middleware included by default.

Step 3 – Providing store to complete react app

This will provide store globally.

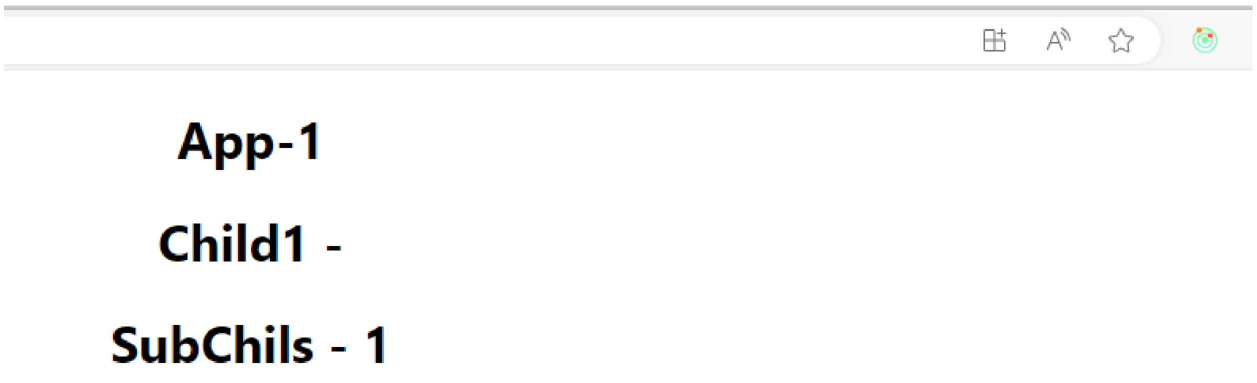
Go to **src/index.js** :

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import { store } from "./app/store";
import { Provider } from "react-redux";

const root = ReactDOM.createRoot(document.getElementById("roo
t"));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

`Provider` wraps the `App` and the whole application has access to redux store.

Now check your *redux dev tool*

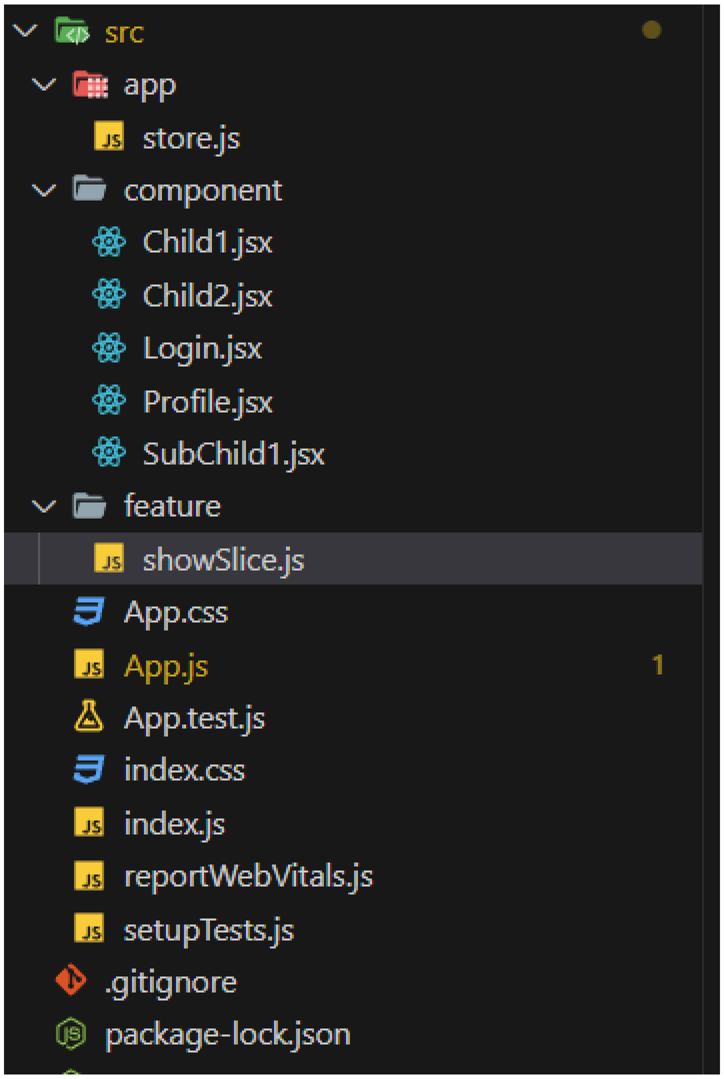
After that you can check that our react page have redux


App-1

Child1 -

SubChils - 1

Create a file ShowSlice.js inside the feature folder slice is the place where logic is implemented. slice is connected to the reducer.



```
src
  app
    store.js
  component
    Child1.jsx
    Child2.jsx
    Login.jsx
    Profile.jsx
    SubChild1.jsx
  feature
    showSlice.js
  App.css
  App.js                1
  App.test.js
  index.css
  index.js
  reportWebVitals.js
  setupTests.js
.gitignore
package-lock.json
```

Example to create the Slice

Here value is a state.

```
import { createSlice } from "@reduxjs/toolkit";
const initialState={value:0}
export const showSlice= createSlice({
    name:"showData",
    initialState,
    reducers:{
        addData:()=>{
        },
        showData:()=>{

        }
    },
})

export const {addData,showData} = showSlice.actions;
export default showSlice.reducer;
```

```
import { createSlice } from "@reduxjs/toolkit";
const initialState={value:0}
export const showSlice= createSlice({
    name:"showData",
    initialState,
    reducers:{
        showData:(state,action)=>{
            state.value=state.value;
        }
    },
})

export const {showData} = showSlice.actions;
export default showSlice.reducer;
//
```

state is like value.

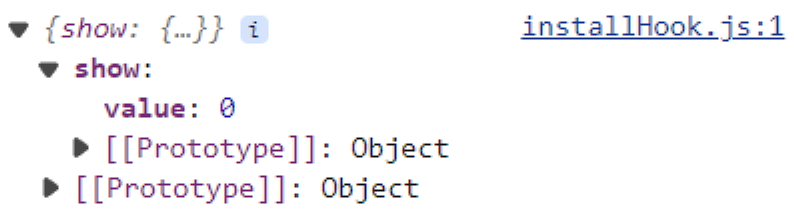
Step 4: - Add this slice to store

```
import { configureStore } from "@reduxjs/toolkit";
import  showData  from "../feature/showSlice";

export const store = configureStore({
  reducer: {
    show:showData
  },
});
```

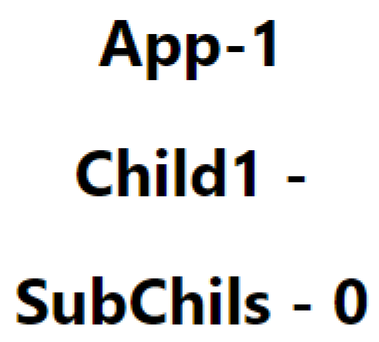
Step 5

use of useSelector hook to access the value

```
import React from 'react'
import { useSelector } from 'react-redux'

const SubChild1 = () => {
    const data=useSelector((c)=>{
        console.log(c)
        return c.show.value;
    })
  return (
    <div>
      <h1>SubChils - {data}</h1>
    </div>
  )
}

export default SubChild1
```



```
▼ {show: {…}} i                    installHook.js:1
  ▼ show:
      value: 0
    ▶ [[Prototype]]: Object
  ▶ [[Prototype]]: Object
```



App-1

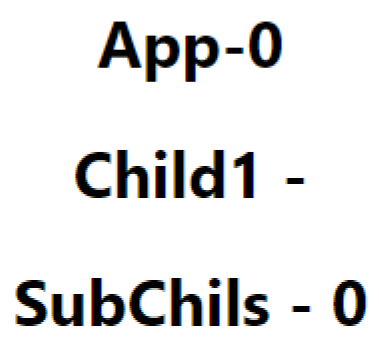
Child1 -

SubChils - 0

File : - App.js

```
import { useState } from 'react';
import './App.css';
import Child1 from './component/Child1'
import { useSelector } from 'react-redux';
function App() {

  const [val,setValue]=useState(1)
  const data=useSelector((c)=>{
    return c.show.value;
  })
  return (
    <div className="App">
      <h1>App-{data}</h1>
      <Child1 value={val}/>
    </div>
  );
}
```

```
export default App;
```



App-0

Child1 -

SubChils - 0

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { increment } from '../feature/showSlice';
```

```
const SubChild1 = () => {
  const dispatch=useDispatch();//send the data to reducer

  const data=useSelector((c)=>{ //fetch the data from reducer
    return c.show.value
  })
  return (
    <div>
      <h2>SubChild1--{data}</h2>
      <button onClick={()=>dispatch(increment())}>click me </bu
    </div>
  )
}

export default SubChild1
```

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { increment,incrementByValue } from '../feature/showSlic

const SubChild1 = () => {

    const dispatch = useDispatch();

    const data=useSelector((c)=>{
        console.log(c)
        return c.show.value;
    })
  return (
    <div>
      <h1>SubChils - {data}</h1>
      <button onClick={()=>dispatch(increment())}>Click me</but
```

```
        <button onClick={()=>dispatch(incrementByValue(10))}>Clic

      </div>
    )
}

export default SubChild1
```

Now i will my final code for all my file

Child 1

```
import React from 'react'
import SubChild1 from './SubChild1'
import { useSelector } from 'react-redux'

const Child1 = () => {

  return (
    <div>
    <h1>Child1 -</h1>
      <SubChild1 subval/>
    </div>
  )
}

export default Child1
```

subchild1.jsx

```
import React from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { increment,incrementByValue } from '../feature/showSlic

const SubChild1 = () => {

    const dispatch = useDispatch();

    const data=useSelector((c)=>{
        console.log(c)
        return c.show.value;
    })
  return (
    <div>
      <h1>SubChils - {data}</h1>
      <button onClick={()=>dispatch(increment())}>Click me</but
      <button onClick={()=>dispatch(incrementByValue(10))}>Clic

    </div>
  )
}

export default SubChild1
```

store.js

```
import { configureStore } from "@reduxjs/toolkit";
import  showData  from "../feature/showSlice";

export const store = configureStore({
  reducer: {
    show:showData
  },
});
```

showslice

```
import { createSlice } from "@reduxjs/toolkit";
const initialState={value:0}
export const showSlice= createSlice({
    name:"showData",
    initialState:,
    reducers:{
        showData:(state,action)=>{
            state.value=state.value;
        },
        increment:(state)=>{
            state.value=state.value+1;
        },
        incrementByValue:(state,action)=>{
            state.value=state.value+action.payload;
        }
    },
})

export const {showData,increment,incrementByValue} = showSlice.a
export default showSlice.reducer;
```

app.js

```
import { useState } from 'react';
import './App.css';
import Child1 from './component/Child1'
import { useSelector } from 'react-redux';
function App() {

  const [val,setValue]=useState(1)
  const data=useSelector((c)=>{
    return c.show.value;
  })
  return (
    <div className="App">
      <h1>App-{data}</h1>
      <Child1 value={val}/>
    </div>
  );
}

export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { Provider } from 'react-redux';
import { store } from './app/store';
const root = ReactDOM.createRoot(document.getElementById('root'
root.render(
  <React.StrictMode>
  <Provider store={store}>
    <App />
  </Provider>
  </React.StrictMode>
);
```