# شروع و نصب ها و معرفی

## پیش نیاز نصب

برای شروع باید فایل های زیر را نصب کنی

Node -v

Npm -v

**How To Install React on Windows**

1. Step 1: Install Node. js and npm.
2. Step 2: Install Create React App.
3. Step 3: Create a New React Project.
4. Step 4: Go To the Project Directory and Start the Development Server.

.

C:\Users>d:

D:\>cd Prpjects

D:\Prpjects>cd React

D:\Prpjects\React>cd ReactTorturial

D:\Prpjects\React\ReactTorturial>npx create-react-app myreact-app

…

Npx start

…

Cd .. یک فولدر به عقب بر میگرده

نکته اسامی باید تمام حروف کوچک باشد

..................

## روش نصب و استفاده از bootstrap

Npm Install –save bootstrap

.

import React, {Component } from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

## روش کپی گرفتن از آبجکت

var dataPass = { ...duty }

.

# نکته ها

## برای اینکه نمایش یک فانکشن جدا کنم و ساده ترش کنم

برای اینکه نمایش یک فانکشن جدا کنم و ساده ترش کنم میتوانیم بصورت زیر عمل کنیم

import { toast } from "react-toastify";

export const successMessage = message => {

    toast.success(message, {

        position: "top-right",

        closeOnClick: true

    });

};

export const errorMessage = message => {

    toast.error(message, {

        position: "top-right",

        closeOnClick: true

    });

};

...

و به صورت زیر استفاده میکنیم

import { successMessage, errorMessage } from "./../../utils/message";

....

 if (validator.current.allValid()) {

                const { status, data } = await loginUser(user);

                if (status === 200) {

                    successMessage("ورود موفقیت آمیز بود.");

                    localStorage.setItem("token", data.token);

                    dispatch(addUser(decodeToken(data.token).payload.user));

                    history.replace("/");

                    resetStates();

                }

            } else {

                validator.current.showMessages();

                forceUpdate(1);

            }

        } catch (ex) {

            console.log(ex);

            errorMessage("مشکلی پیش آمده.");

        }

.................

# Java Script

## Get element by name , class , type

گرفتن المنت از طریق نام و یا گرفتن تمام المنت های از نوع input

 <input type='checkbox'

 onChange={handleChangeData}

 value={item.id}

 name='chkDutyNotExecutedToday' />

id

    var x = document.getElementById("myBtn");

tagname

 var node\_list = document.getElementsByTagName('input');

name

 var node\_list = document.getElementsByName('chkDutyNotExecutedToday');

class

 let box = document.querySelector('.datePersianPicker');

.

<input type="text"

                    className='datePersianPicker'

                    name="dateTaskIsExecute"

                    onClick={showDatePicker}

                    autocomplete="off"

                    value={dateTransfer}

                    onChange={handleChange}

                    placeholder="تاریخ انجام" />

.

پیدا کردن پهنا و ارتفاع المنت

  let box = document.querySelector('.datePersianPicker');

    let width = 0

    if (box != null) {

        width = box.offsetWidth;

        let height = box.offsetHeight;

    }

........

## چک کردن نوع المنت بصورت ز یر

if (node.getAttribute('type') == 'checkbox') {

}

……………..

   function getcheckboxes() {

        var node\_list = document.getElementsByTagName('input');

        var node\_list = document.getElementsByName('chkDutyNotExecutedToday');

        var checkboxes = [];

        for (var i = 0; i < node\_list.length; i++) {

            var node = node\_list[i];

            if (node.getAttribute('type') == 'checkbox') {

                checkboxes.push(node);

            }

        }

        return checkboxes;

    }

…

## Event

    var x = document.getElementById("myBtn");

    x.addEventListener("mouseover", myFunction);

    x.addEventListener("click", mySecondFunction);

    x.addEventListener("mouseout", myThirdFunction);

## نوشتن html از نوع string با js

function App() {

  useEffect(() => {

     ss()

  }, []);

  return (

    <div className="App">

  <div id='fery'></div>

    </div>

  );

}

export default App;

function ss() {

  //This is your fetched html string

  var body = "<div style='text-align:center'>"

  body += "<h2>data from javascript append to id fery</h2>"

  body += "<h1> Test</h1>"

  body += "<table>"

  body += "<tr>"

  body += "<th>1</th>"

  body += "<th>2</th>"

  body += "</tr>"

  body += "<tr>"

  body += "<td>ff</td>"

  body += "<td>ff</td>"

  body += "</tr>"

  body += "</table>"

  body += "</div>"

  //Create temp element holder

  var tag = createElementFromHTML(body)

  document.getElementById('fery').append(tag)

}

function createElementFromHTML(htmlString) {

  var div = document.createElement('div');

  div.innerHTML = htmlString.trim();

  // Change this to div.childNodes to support multiple top-level nodes.

  return div.firstChild;

}

.

# props.children

import React from 'react';

import Alert from './Alert'

import 'bootstrap/dist/css/bootstrap.min.css';

export default class App extends React.Component {

    render() {

        return (

            <div>

                <Alert class="alert alert-success">

                Hello success

                </Alert>

                <Alert class="alert alert-warning">

                    Hello warning

                </Alert>

                <Alert class="alert alert-danger">

                    Hello danger

                </Alert>

            </div>

        );

    }

}

…

import React, { Component } from 'react';

export default class Alert extends Component {

    render() {

        return (

            <div className={this.props.class}>

                {this.props.children}

            </div>

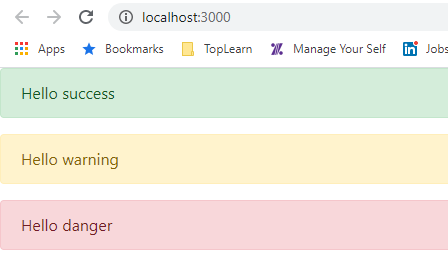
        );

    }

}

.

خروجی :



…………..

# State

## Prevoius state

برای اینکه 5 تا 5 تا زیاد کنی وقتی در حلقه هستش کار نمیکند و باید از فانکشن استفاده کنی بجای x میتوانی هر چیزی بزاری

import React, { useState } from 'react'

const HookCounterTwo = () => {

    const initialCount = 0

    const [count, setCount] = useState(initialCount)

    const increamentFive = () => {

        for (let i = 0; i < 5; i++) {

            // setCount(count + 1)   not work

            setCount(x => x + 1)

        }

    }

    return (<>

        count : {count}

        <button onClick={() => setCount(initialCount)}>Reset</button>

        <button onClick={() => setCount(count + 1)}>Increment</button>

        <button onClick={() => setCount(count - 1)}>Decrement</button>

        <button onClick={increamentFive}>increamentFive</button>

    </>)

}

export default HookCounterTwo

# use effect

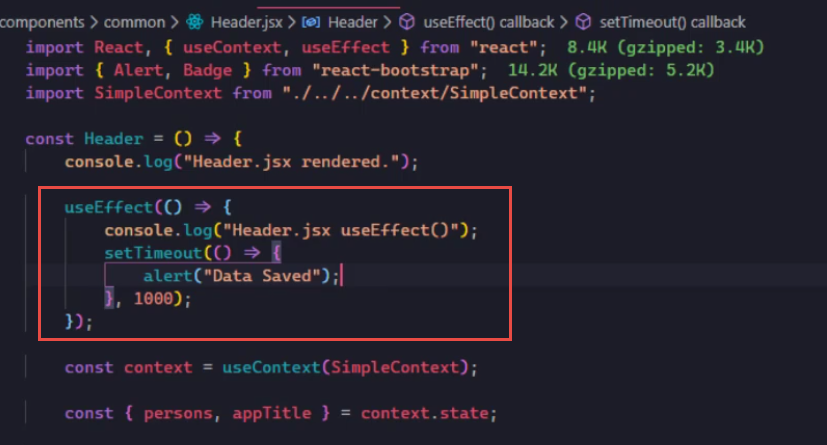
.

انواع lifeCycle

چرخه حیات کامپوننت های تابعی

UseEffect : هم برای چرخه ساخت و هم بروز رسانی و هم تمیزکاری استفاده میشود

## همیشه اجرا میشود



در این حالت useEffect دائما با هر تغییر اجرا میشود برای اینکه کاری کنیم که در زمان خاصی اجرا بشه مثلا زمانی که persons تغییر کرد بیاد اجرا بشه بصورت زیر persons را بعد از کال بک در آرایه قرار میدهیم.

## با شرایط خاص اجرا میشود



## فقط یکبار اجرا میشود

اگر ما بجای ارایه persons یک آرایه خالی مانند مثال زیر بدهیم تنها و تنها یکبار اجرا میشود

که فقط هنگام ساخت کامپوننت اجرا میشود



.....................................................

## حالا برای unmount شدن و یا همان تمیز کاری بصورت زیر استفاده میکنیم

برا اینکه ComponentWillunmount کنیم باید return کنیم یک فانکشن را

اما مشکلی که روش زیر دارد این است که با هر تغییر persons بکبار اجرا میشود برای اینکه فقط یکبار اجرا شود ارایه خالی باید پاس بدهیم

.........

import React, { useContext, useEffect } from "react";

import { Alert, Badge } from "react-bootstrap";

const Header = ({ persons, appTitle }) => {

    useEffect(() => {

        const timer = setTimeout(() => {

            alert("Data Saved");

        }, 1000);

        return () => {

            console.log("Header.jsx Unmount");

            clearTimeout(timer);

        };

    }, []);

    return (

        <div>

        </div>

    );

};

export default Header;

…

مثال دوم

وقتی رو دکمه عدم نمایش کلیک میکنی دیگر کامپوننت اجرا نمیشود و پاک میشود

import React, { useState,useEffect } from 'react'

import HookMouse from './HookMouse'

const MouseContainer=()=>{

const [display, setDisplay] = useState(true)

useEffect(() => {

}, [])

return( <>

<button onClick={()=>setDisplay(!display)}>Toggle Display</button>

{display && <HookMouse/>}

   </>

   )

}

export default MouseContainer

.....

import React, { useEffect, useState } from 'react'

const HookMouse = () => {

    const [x, setX] = useState(0)

    const [y, setY] = useState(0)

    const logMousePosition = e => {

        // console.log(e)

        // console.log("mouse event")

        setX(e.clientX)

        setY(e.clientY)

    }

    useEffect(() => {

        console.log('useEffected Called')

        window.addEventListener('mousemove', logMousePosition)

        return () => {

            console.log("component unmoounting code")

            window.removeEventListener('mousemove', logMousePosition)

        }

    }, [])

    return (<>

        <p style={{ backgroundColor: 'red' }}>  Mouse \*\*\*  x: {x} -  Y: {y}</p>

    </>)

}

export default HookMouse

…

Use effect with incorect dependency

<https://blog.logrocket.com/solve-react-useeffect-hook-infinite-loop-patterns/>

import React, { useState, useEffect } from 'react'

const IntervalHookCounter = () => {

    const [count, setCount] = useState(0)

    const tick = () => {

        // setCount(count+1)

        setCount(x => x + 1)

    }

    useEffect(() => {

        const interval = setInterval(tick, 1000);

        // return () => {

        //     console.log("clearInterval")

        //     clearInterval(interval)

        // }

    }, [])

    return (<>

        {count}

    </>)

}

export default IntervalHookCounter

## What causes infinite loops and how to solve them

1- Passing no dependencies in a dependency array

If your useEffect function does not contain any dependencies, an infinite loop will occur.

For example, look at the following code:

function App() {

const [count, setCount] = useState(0); //initial value of this

useEffect(() => {

setCount((count) => count + 1); //increment this Hook

}); //no dependency array.

return (

<div className="App">

<p> value of count: {count} </p>

</div>

);

}

useEffect by default triggers on every update cycle if there are no dependencies. As a result, the app here will execute the setCount function upon every render. So, this causes an infinite

What causes this issue?

Let’s break down our issue step by step:

1. On the first render, React checks the value of count. Here, since count is 0, the program executes the useEffect function
2. Later on, useEffect invokes the setCount method and updates the value of the count Hook
3. After that, React re-renders the UI to display the updated value of count
4. Furthermore, since useEffect runs on every render cycle, it re-invokes the setCount function
5. Since the above steps occur on every render, this causes your app to crash

How to fix this issue

To mitigate this problem, we have to use a dependency array. This tells React to call useEffect only if a particular value updates.

As the next step, append a blank array as a dependency like so:

useEffect(() => {

setCount((count) => count + 1);

}, []); //empty array as second argument.

This tells React to execute the setCount function on the first mount.

.

2. Using a function as a dependency

…

.

## Multiple useeffect

import React, { useState,useEffect } from 'react'

import HookMouse from './HookMouse'

const MouseContainer=()=>{

const [display, setDisplay] = useState(true)

useEffect(() => {

    console.log("initial MouseContainer")

}, [])

    useEffect(() => {

        console.log("display MouseContainer")

    }, [display])

return( <>

<button onClick={()=>setDisplay(!display)}>Toggle Display</button>

{display && <HookMouse/>}

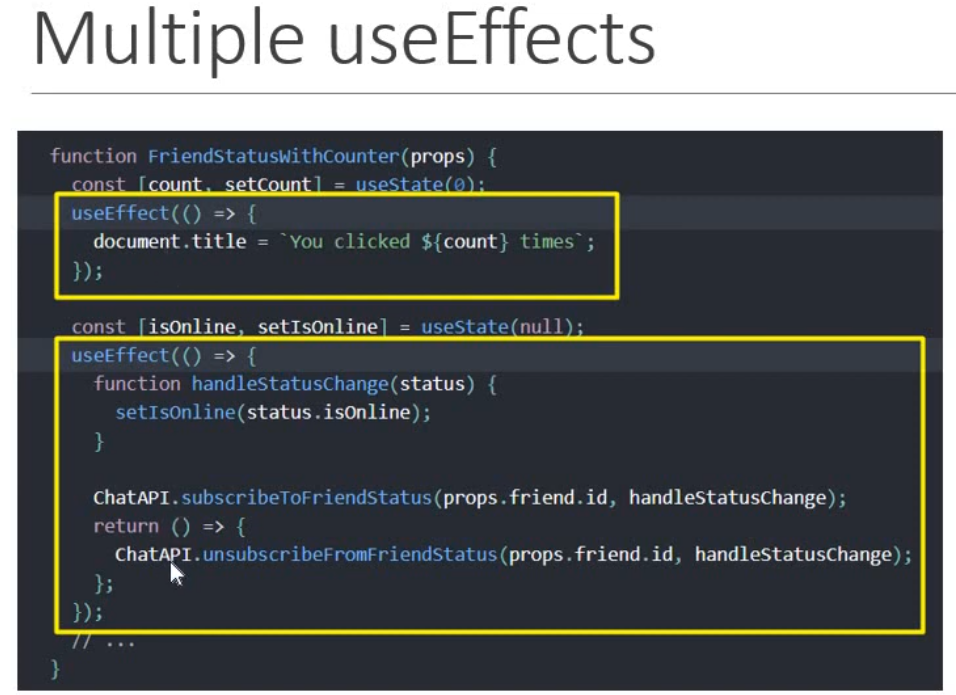
   </>

   )

}

export default MouseContainer

.



.

# Fetch Data -axios

یکی از مزیت های axios این است که از promise ها پشتیبانی میکند

قل میده این کار را انجام بده اگر نشد خطا میدهد

از جهنم calback جلوگیری میکند

.........

ایتدا axois را نصب کنید

Npm install axios

import axios from 'axios'

import React, { useState, useEffect } from 'react'

const DataFetching = () => {

    const [posts, setPosts] = useState([])

    useEffect(() => {

        axios.get("https://jsonplaceholder.typicode.com/posts")

            .then(res => {

                console.log(res)

                setPosts(res.data)

            })

            .catch(err => {

                console.log(err)

            })

    })

    return (

        <>

            {posts.map(post =>

                <p key={post.id}>{post.title}</p>

            )}

        </>

    )

}

export default DataFetching

گرفتن دیتا بصورت تکی با id

import axios from 'axios'

import React, { useState, useEffect } from 'react'

const DataFetchingById = (props) => {

    const [id, setId] = useState(1)

    const [post, setPost] = useState({})

    useEffect(() => {

        debugger

        console.log('DataFetchingById Called')

        axios.get(`https://jsonplaceholder.typicode.com/posts/${props.id}`)

            .then(res => {

                console.log(res)

                setPost(res.data)

            })

            .catch(err => {

                console.log(err)

            })

    })

    return (<>

        <p>{post.title}</p>

    </>)

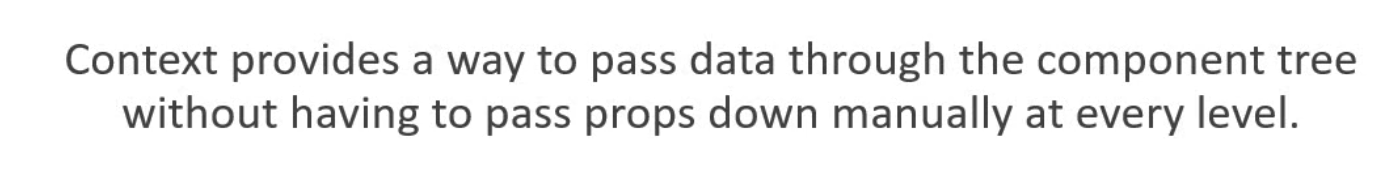
}

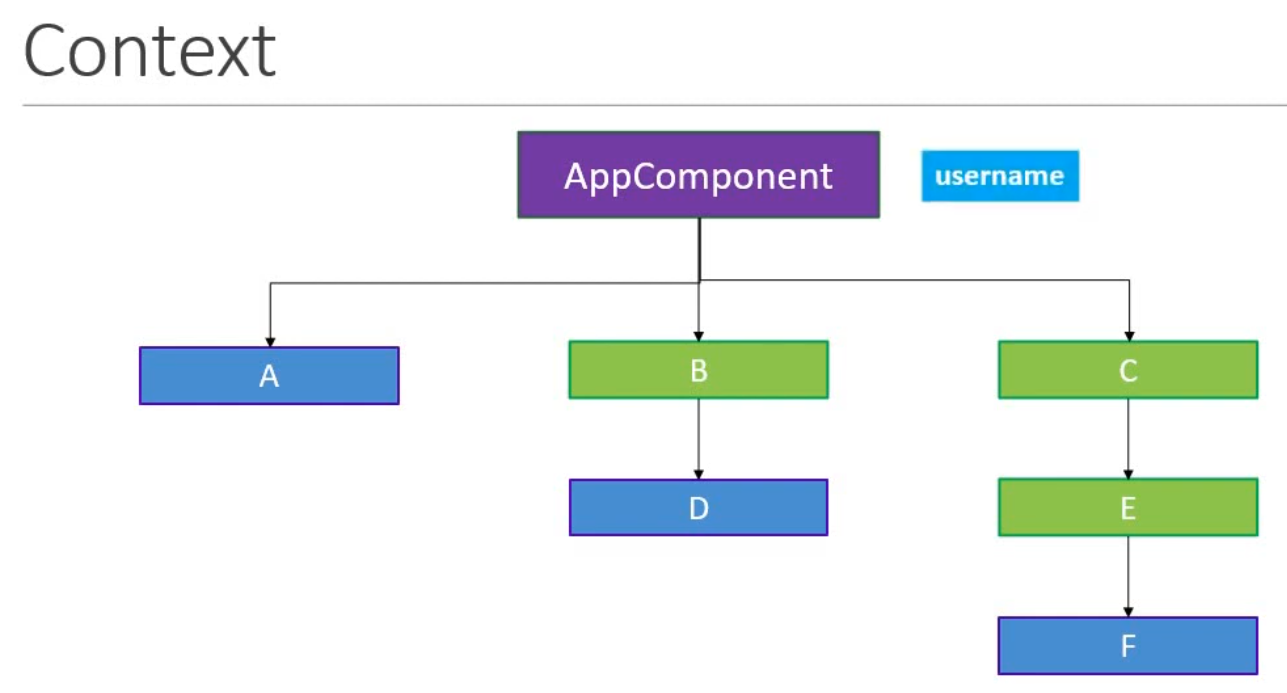
export default DataFetchingById

.

# useContext

فرض کنید بخواهیم username را در کامپوننت های زیر انتشار بدهیم اگر بخواهیم بصورت props اتشار بدهیم کامپوننت E , C اصلا نیازی به username ندارند





.

## createContext و Provider

ابتدا آنرا تعریف میکنیم

* باید بالاتر از تعریف کامپوننت تعریف شود
* باید export شود
* باید کامپوننت های دیگر در دل context provider قرار بگیرند

import React, { useState, useEffect } from 'react'

import ComponentC from './ComponentC'

export const UserContext = React.createContext()

export const ChanelContext = React.createContext()

const MainComponent = () => {

    const [isShow, setIsShow] = useState(false)

    useEffect(() => {

        console.log('useEffected Called')

    }, [])

    return (

        <>

            <UserContext.Provider value={'username'}>

                <ChanelContext.Provider value={'chanel Data'}>

                    <ComponentC />

                </ChanelContext.Provider>

            </UserContext.Provider>

        </>

    )

}

export default MainComponent

ComponentC

import React, { useState, useEffect } from 'react'

import ComponentE from './ComponentE'

const ComponentC = () => {

    const [isShow, setIsShow] = useState(false)

    useEffect(() => {

        console.log('useEffected Called')

    }, [])

    return (

        <>

        <ComponentE/>

        </>

    )

}

export default ComponentC

ComponentE

import React, { useState, useEffect } from 'react'

import ComponentF from './ComponentF'

const ComponentE = () => {

    const [isShow, setIsShow] = useState(false)

    useEffect(() => {

        console.log('useEffected Called')

    }, [])

    return (

        <>

            <ComponentF />

        </>

    )

}

export default ComponentE

## useContext

و در اینجا ازش استفاده میکنیم

import React, { useState, useEffect, useContext } from 'react'

import { UserContext, ChanelContext } from '../Context/MainComponent'

const ComponentF = () => {

    const [isShow, setIsShow] = useState(false)

    useEffect(() => {

        console.log('useEffected Called')

    }, [])

    const user = useContext(UserContext)

    const channel = useContext(ChanelContext)

    return (

        <>

            <p>component F</p>

            <p>{user} - {channel}</p>

        </>

    )

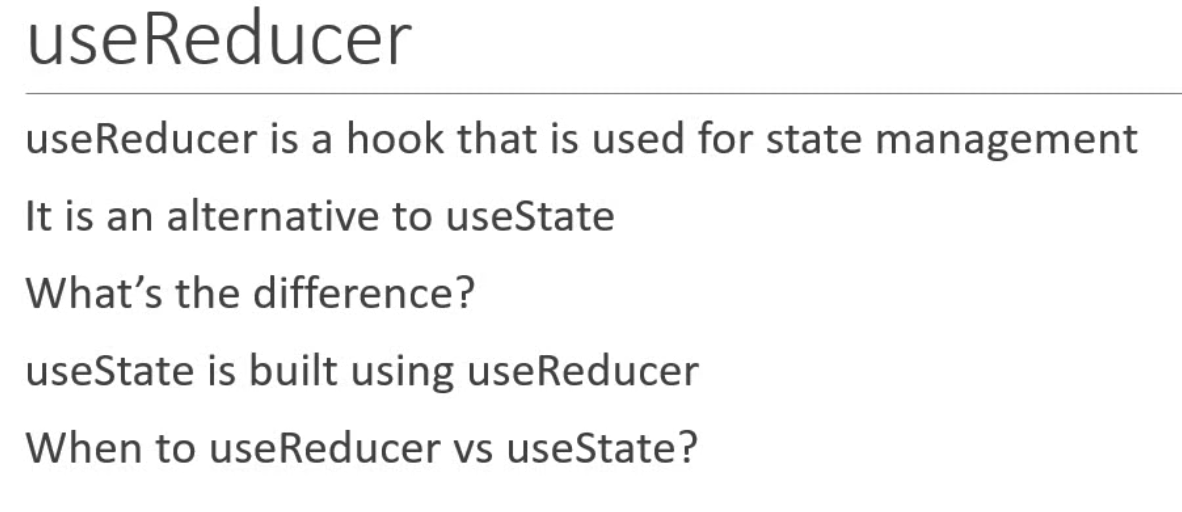
}

export default ComponentF

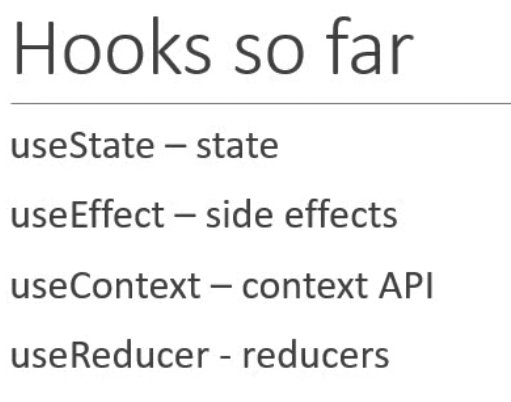
.

# userReducer

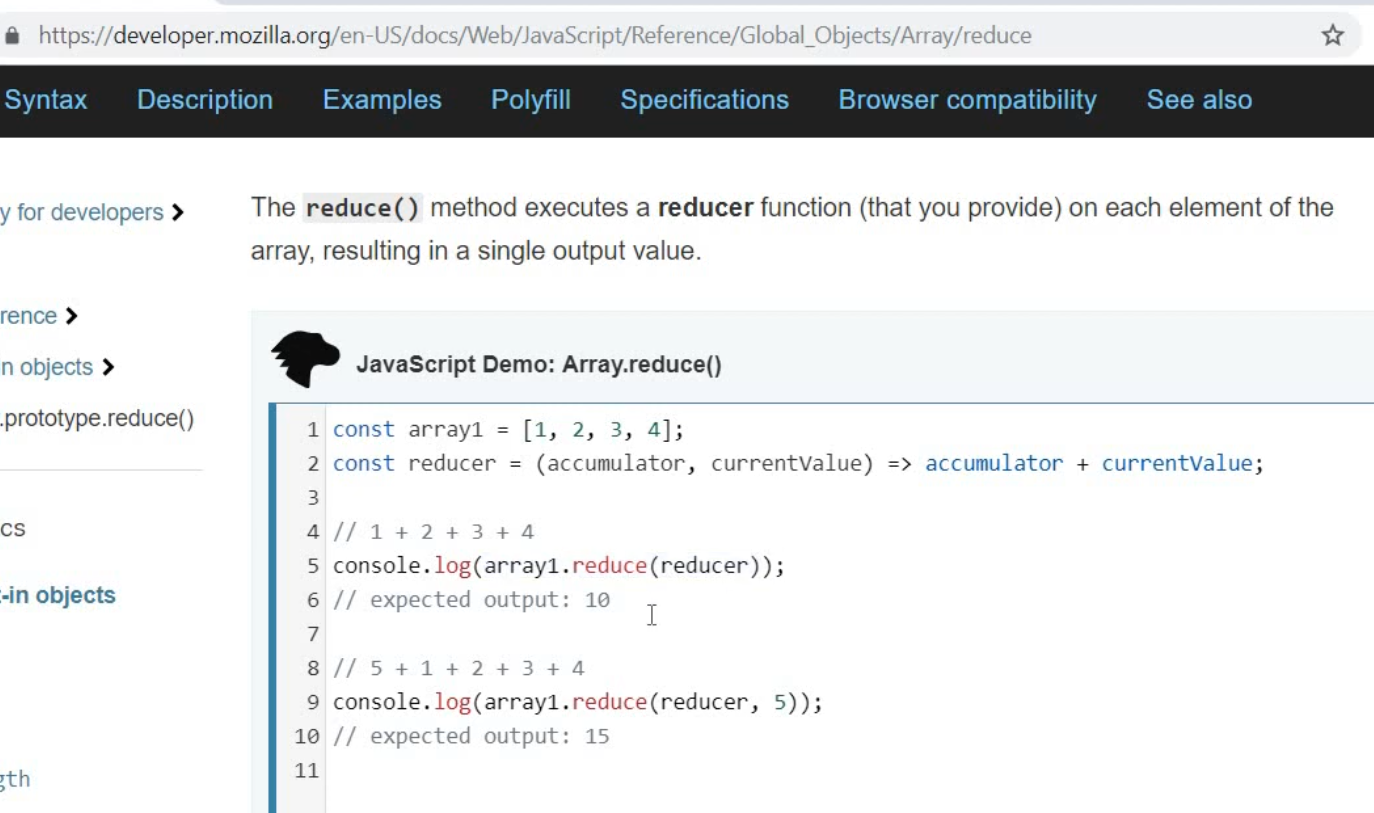
ی



.

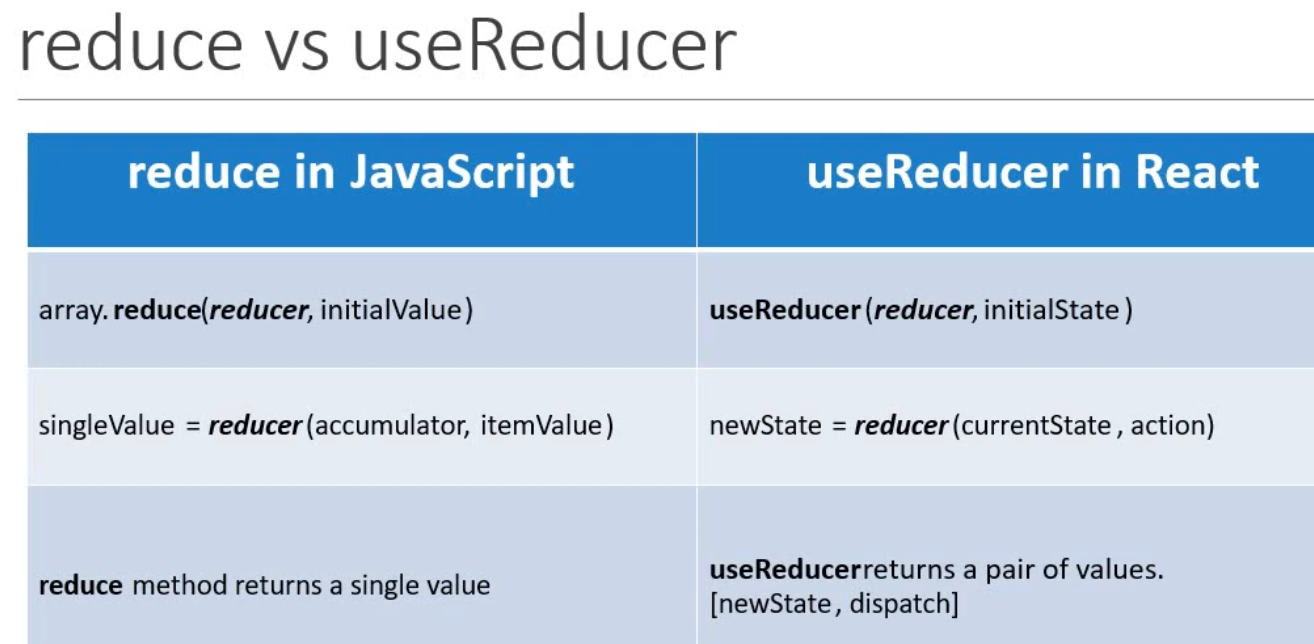


.

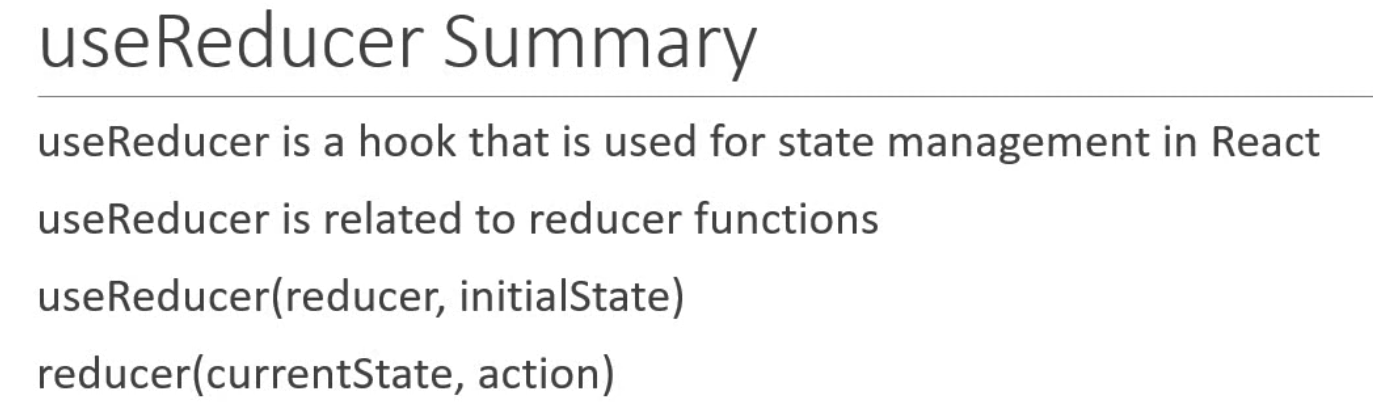


.

.



.



.

## بخش اول

import React, { useReducer } from 'react'

const initialState = 0

const reducer = (state, action) => {

    switch (action) {

        case 'increament':

            return state + 1

        case 'decreament':

            return state - 1

        case 'reset':

            return initialState

        default:

            return state

    }

}

const CounterOne = () => {

    const [count, dispatch] = useReducer(reducer, initialState)

    return (

        <>

            <p>count : {count}</p>

            <button onClick={() => dispatch('increament')}>Increament</button>

            <button onClick={() => dispatch('decreament')}>Decreament</button>

            <button onClick={() => dispatch('reset')}>Reset</button>

        </>

    )

}

export default CounterOne

........

## بخش دوم

import React, { useReducer } from 'react'

const initialState = {

    firstCounter: 0,

    secondCounter: 10

}

const reducer = (state, action) => {

    switch (action.type) {

        case 'increament':

            return { ...state, firstCounter: state.firstCounter + action.value }

        case 'decreament':

            return { ...state, firstCounter: state.firstCounter - action.value }

        case 'increament2':

            return { ...state, secondCounter: state.secondCounter + action.value }

        case 'decreament2':

            return { ...state, secondCounter: state.secondCounter - action.value }

        case 'reset':

            return initialState

        default:

            return state

    }

}

const CounterTwo = () => {

    const [count, dispatch] = useReducer(reducer, initialState)

    return (

        <>

            <p>first Counter  : {count.firstCounter}</p>

            <p>second Counter  : {count.secondCounter}</p>

            <button

                onClick={() => dispatch({ type: 'increament', value: 1 })}

            >

                Increament

            </button>

            <button

                onClick={() => dispatch({ type: 'decreament', value: 1 })}

            >

                Decreament

            </button>

            <button

                onClick={() => dispatch({ type: 'increament', value: 5 })}

            >

                Increament 5

            </button>

            <button

                onClick={() => dispatch({ type: 'decreament', value: 5 })}>Decreament 5

            </button>

            <div>

                <button

                    onClick={() => dispatch({ type: 'increament2', value: 1 })}

                >

                    Increament Counter

                </button>

                <button

                    onClick={() => dispatch({ type: 'decreament2', value: 1 })}

                >

                    Decreament Counter 2

                </button>

            </div>

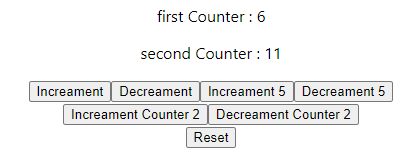
            <button onClick={() => dispatch({ type: 'reset' })}>Reset</button>

        </>

    )

}

export default CounterTwo



.

## بخش سوم

در کد زیر همانطور که مشاهده مینمایید از یک reducer استفاده شده است ولی انگار دو متد متفاوت نوشته ایم

import React, { useReducer } from 'react'

const initialState = 0

const reducer = (state, action) => {

    switch (action) {

        case 'increament':

            return state + 1

        case 'decreament':

            return state - 1

        case 'reset':

            return initialState

        default:

            return state

    }

}

const CounterThree = () => {

    const [count, dispatch] = useReducer(reducer, initialState)

    const [countTwo, dispatchTwo] = useReducer(reducer, initialState)

    return (

        <>

            <p>count : {count}</p>

            <button onClick={() => dispatch('increament')}>Increament</button>

            <button onClick={() => dispatch('decreament')}>Decreament</button>

            <button onClick={() => dispatch('reset')}>Reset</button>

            <hr/>

            <p>count Two : {countTwo}</p>

            <button onClick={() => dispatchTwo('increament')}>Increament</button>

            <button onClick={() => dispatchTwo('decreament')}>Decreament</button>

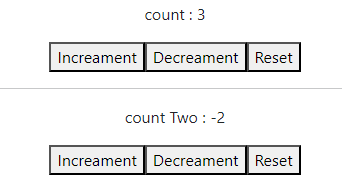
            <button onClick={() => dispatchTwo('reset')}>Reset</button>

        </>

    )

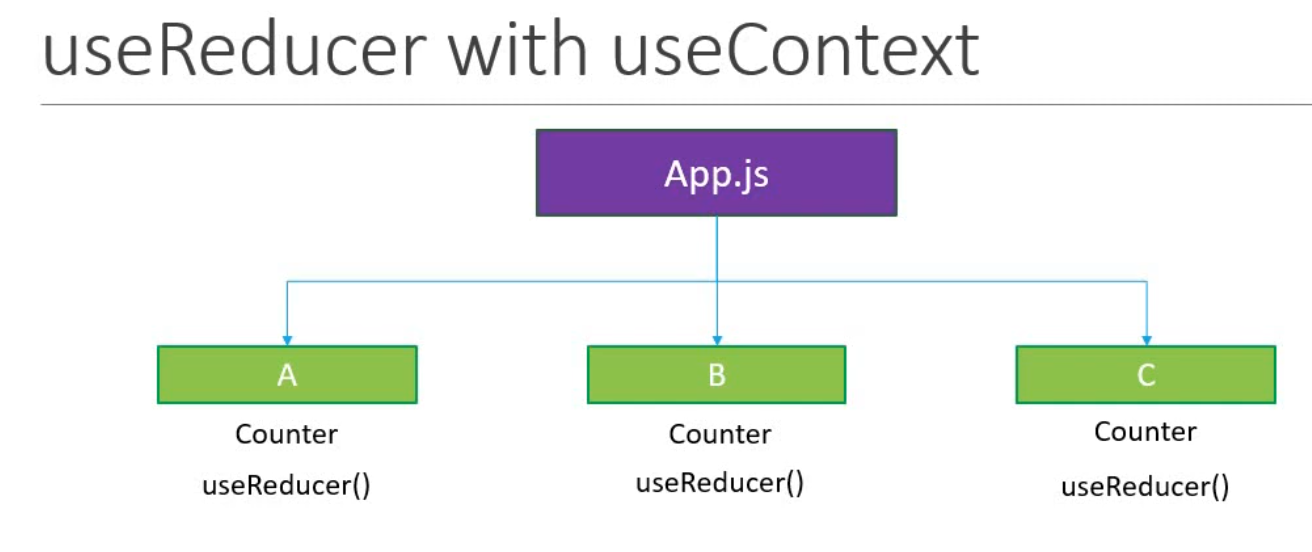
}

export default CounterThree

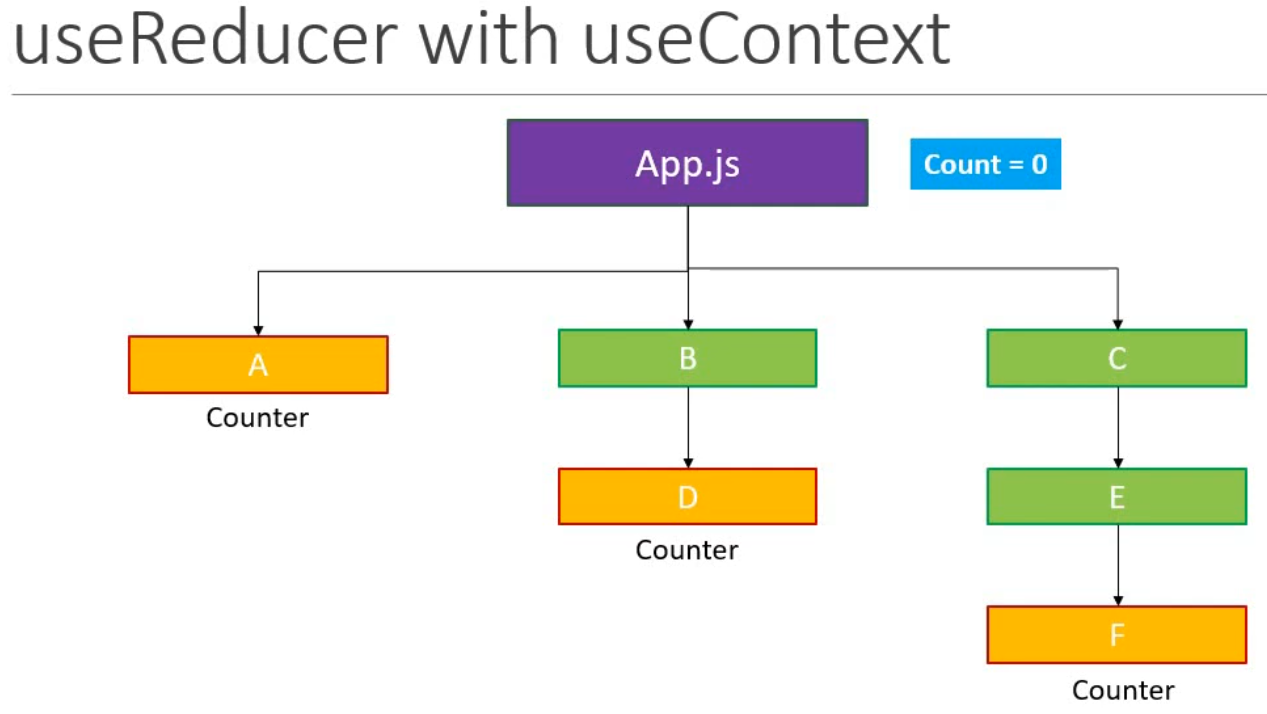


.

## UseReducer with UseContext



.



.

import React, { useEffect, useReducer } from "react";

import ComponentA from './AnotherComponent/UseReducerWithUseContext/ComponentA';

import ComponentB from './AnotherComponent/UseReducerWithUseContext/ComponentB';

import ComponentC from './AnotherComponent/UseReducerWithUseContext/ComponentC';

export const CountContextNew = React.createContext()

const initialState = 0

const reducer = (state, action) => {

  switch (action) {

    case 'increament':

      return state + 1

    case 'decreament':

      return state - 1

    case 'reset':

      return initialState

    default:

      return state

  }

}

function App() {

  useEffect(() => {

    console.log('App Called')

  }, []);

  const [count, dispatch] = useReducer(reducer, initialState)

  return (

    <div className="App">

      <CountContextNew.Provider value={{countState:count,countDispatch:dispatch}}>

        Count : {count}

        <ComponentA />

        <ComponentB />

        <ComponentC />

      </CountContextNew.Provider>

    </div>

  );

}

export default App;

.

import React, { useState, useEffect, useContext } from 'react'

import { CountContextNew } from '../../App'

const ComponentA = () => {

    const [isShow, setIsShow] = useState(false)

    useEffect(() => {

        console.log('ComponentA Called')

    }, [])

    const countContextNew = useContext(CountContextNew)

    return (

        <>

            <h1>Component A {countContextNew.countState}</h1>

            <button onClick={() => countContextNew.countDispatch('increament')}>Increament</button>

            <button onClick={() => countContextNew.countDispatch('decreament')}>Decreament</button>

            <button onClick={() => countContextNew.countDispatch('reset')}>Reset</button>

        </>

    )

}

export default ComponentA

.

import React, { useState, useEffect, useContext } from 'react'

import { CountContextNew } from '../../App'

const ComponentD=()=>{

const [isShow, setIsShow] = useState(false)

useEffect(() => {

    console.log('ComponentD Called')

}, [])

    const countContextNew = useContext(CountContextNew)

return(

<>

        <h1>Component D {countContextNew.countState}</h1>

        <button onClick={() => countContextNew.countDispatch('increament')}>Increament</button>

        <button onClick={() => countContextNew.countDispatch('decreament')}>Decreament</button>

        <button onClick={() => countContextNew.countDispatch('reset')}>Reset</button>

</>

)

}

export default ComponentD

.

import React, { useState, useEffect, useContext } from 'react'

import { CountContextNew } from '../../App'

const ComponentF = () => {

    const [isShow, setIsShow] = useState(false)

    useEffect(() => {

        console.log('ComponentF Called')

    }, [])

    const countContextNew = useContext(CountContextNew)

    return (

        <>

            <h1>Component F {countContextNew.countState}</h1>

            <button onClick={() => countContextNew.countDispatch('increament')}>Increament</button>

            <button onClick={() => countContextNew.countDispatch('decreament')}>Decreament</button>

            <button onClick={() => countContextNew.countDispatch('reset')}>Reset</button>

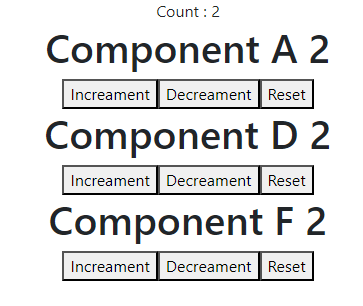
        </>

    )

}

export default ComponentF

.



.

## FetchingData Use Reducer

به صورت معمول برای نمایش دیتا به روش زیر عمل میکنیم

import React, { useEffect, useReducer } from "react";

import DataFetchingOne from './AnotherComponent/ReducerFetchingData/DataFetchingOne';

function App() {

  return (

    <div className="App">

      <DataFetchingOne />

    </div>

  );

}

export default App;

.

import axios from 'axios'

import React, { useState, useEffect } from 'react'

const DataFetchingOne = () => {

    const [loading, setLoading] = useState(true)

    const [error, setError] = useState('')

    const [post, setPost] = useState({})

    useEffect(() => {

        axios.get(`https://jsonplaceholder.typicode.com/posts/1`)

            .then(res => {

                setLoading(false)

                setError('')

                setPost(res.data)

            })

            .catch(err => {

                setLoading(false)

                setError('somthing went wrong! ')

                setPost({})

            })

    }, [])

    return (

        <>

            {loading ? 'loading' : post.title}

            {error ? error : null}

        </>

    )

}

export default DataFetchingOne

برای اینکه بصورت reducer بنویسم به روش زیر عمل میکنیم

import React, { useEffect, useReducer } from "react";

import DataFetchingOne from './AnotherComponent/ReducerFetchingData/DataFetchingOne';

import DataFetchingTwo from "./AnotherComponent/ReducerFetchingData/DataFetchingTwo";

function App() {

  return (

    <div className="App">

      <DataFetchingTwo />

    </div>

  );

}

export default App;

.

import React, { useState, useEffect, useReducer } from 'react'

import axios from 'axios'

const initialState = {

    loading: true,

    error: '',

    post: {}

}

const reducer = (state, action) => {

    switch (action.type) {

        case 'fetchSuccess':

            return {

                loading: false,

                post: action.payload,

                error: ''

            }

        case 'fetchError':

            return {

                loading: false,

                post: {},

                error: 'somthing went wrong? '

            }

        default:

            return state

    }

}

const DataFetchingTwo = () => {

    const [state, dispatch] = useReducer(reducer, initialState)

    useEffect(() => {

        axios.get(`https://jsonplaceholder.typicode.com/posts/1`)

            .then(res => {

                dispatch({ type: 'fetchSuccess', payload: res.data })

            })

            .catch(err => {

                dispatch({ type: 'fetchError' })

            })

    }, [])

    return (

        <>

            {state.loading ? 'loading' : state.post.title}

            {state.error ? state.error : null}

        </>

    )

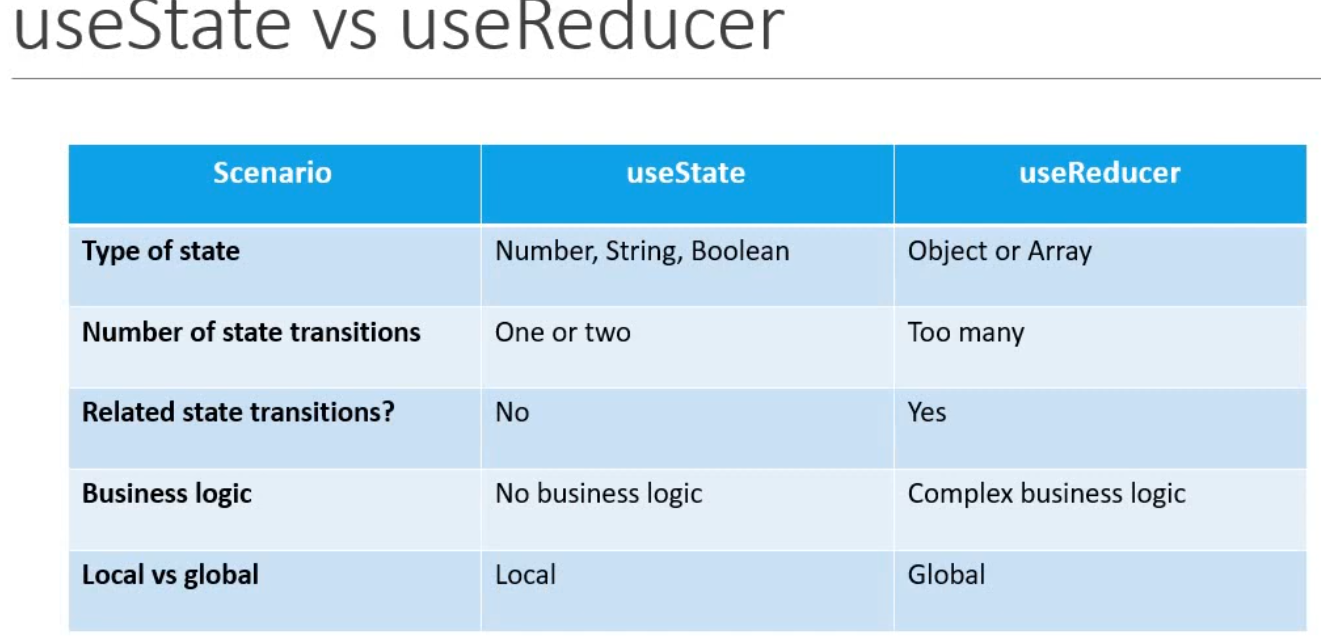
}

export default DataFetchingTwo

..............

# UseState vs useReducer

ی



.

# UseCallback Hook

با این روش میتوانیم بگیم بعد از تغییر همه ی component ها render نشودند و برخی که لازم داریم render شود

import React, { useEffect, useReducer } from "react";

import ParentComponent from "./AnotherComponent/ParentComponent";

function App() {

  return (

    <div className="App" style={{'textAlign':'center'}}>

      <ParentComponent />

    </div>

  );

}

export default App;

.

import React, { useState, useEffect } from 'react'

import Title from './Titlte'

import Button from './Button'

import Count from './Count'

const ParentComponent = () => {

    const [age, setAge] = useState(25)

    const [salary, setSalary] = useState(50000)

    const increamentAge = () => {

        setAge(age + 1)

    }

    const increamentSalary = () => {

        setSalary(salary + 1000)

    }

    return (

        <>

            <Title />

            <Count text='Age' count={age}/>

            <Button handleClick={increamentAge}>Increament Age</Button>

            <Count text='Salary' count={salary} />

            <Button handleClick={increamentSalary}>Increament Salary</Button>

        </>

    )

}

export default ParentComponent

.

import React, { useState, useEffect } from 'react'

const Button = ({ handleClick, children }) => {

    console.log('Rendering Button - ', children)

    return (

        <>

            <button onClick={handleClick}>{children} </button>

        </>

    )

}

export default Button

.

import React, { useState, useEffect } from 'react'

const Count = ({ text, count }) => {

    console.log(`Rendering Count ${text}`)

    return (

        <>

            <p> {text} - {count}</p>

        </>

    )

}

export default Count

.

import React, { useState, useEffect } from 'react'

const Title = () => {

    console.log('Rendering Title')

    return (

        <>

            <h2>useCallback Hook</h2>

        </>

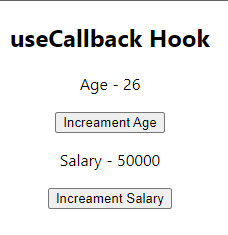
    )

}

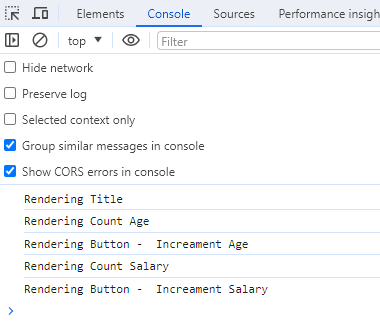
export default Title

.

با کلیک بر روی دکمه ها همانطور که مشاهده مینمایید تمام کامپوننت ها دوباره اجرا میشوند و این اصلا روش خوبی نیست



.



.

برای optimize کردن میتوان از React Memo استفاده کرد یک high order component میباشد که با عث میشود با تغییر props , state دوباره رفرش نشود.

.

export default React.memo(Count)

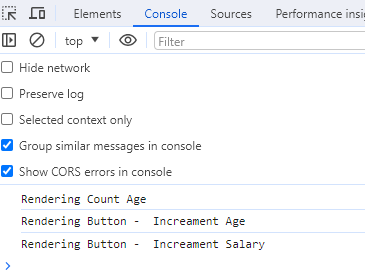
export default React.memo(Button)

export default React.memo(Title)

.

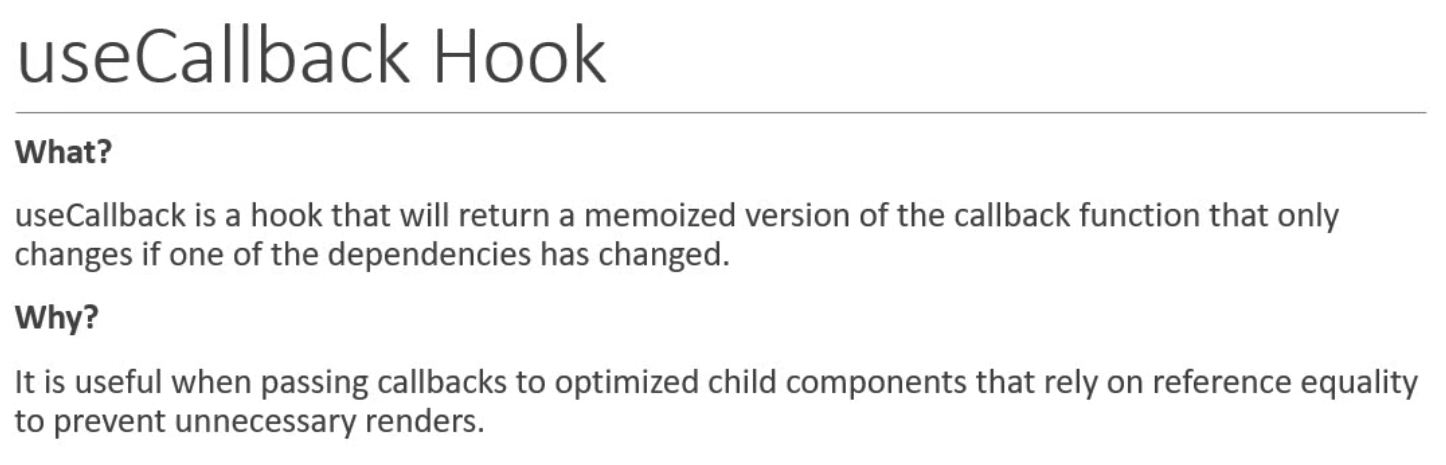
همانطور که میبینید با تغییر state دیگر کامپوننت title رندر میشود

و همچنین component Count چونکه staet مربوط به salary تغییر نکرده دوباره render نشده ولی کامپوننت Button چونکه props ، hadleClick در هر دو مشترک هست دوباره رفرش شده



.

اما چطور بگیم که دکمه دیگر رفرش نشود توسط useCallBack



.

با استفاده از useCallBack بصورت زیر که دارای دو تا پارامتر هستش پارامتر اول فانکشن هستش و پارامتر د.م ارایه ای هستش که میگوید فقط با تغییر این state رفرش شود

import React, { useState, useCallback } from 'react'

import Title from './Titlte'

import Button from './Button'

import Count from './Count'

const ParentComponent = () => {

    const [age, setAge] = useState(25)

    const [salary, setSalary] = useState(50000)

    const increamentAge = useCallback(() => {

        setAge(age + 1) }, [age])

    const increamentSalary = useCallback(() => {

        setSalary(salary + 1000)}, [salary])

    return (

        <>

            <Title />

            <Count text='Age' count={age} />

            <Button handleClick={increamentAge}>Increament Age</Button>

            <Count text='Salary' count={salary} />

            <Button handleClick={increamentSalary}>Increament Salary</Button>

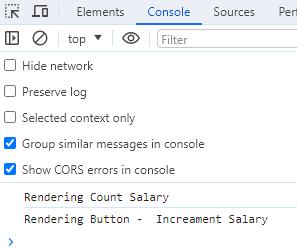
        </>

    )

}

export default ParentComponent

.



.

# useMemo

همانطور که در کد زیر مشاهده مینمایید با کلیک بر روی دکمه دوم نیز باز با کندی یکدونه اضافه میشود

علتش بخاطر این هستش که با تغییر state تمام رندر دوباره اجرا شده و زمانی را دوباره برای رندر کردن span میگذارد

برای اینکه نخواهیم عمل isEven بعد از کلیک بر روی دکمه دوم اتفاق بیفتد باید بگوییم که عمل isEven فقط باید بر روی state ، counterOne اتفاق بیفتد

import React, { useState} from 'react'

const Counter = () => {

    const [counterOne, setCounterOne] = useState(0)

    const [counterTwo, setCounterTwo] = useState(0)

    const incrementOne = () => {

        setCounterOne(counterOne + 1)

    }

    const incrementTwo = () => {

        setCounterTwo(counterTwo + 1)

    }

    const isEven = () => {

        let i = 0

        while (i < 2000000000) i++

        return counterOne % 2 === 0

    }

    return (

        <>

            <div>

                <button onClick={incrementOne}>Count One - {counterOne} </button>

                <span>{isEven() ? 'even' : 'odd'}</span>

            </div>

            <div>

                <button onClick={incrementTwo}>Count Two - {counterTwo} </button>

            </div>

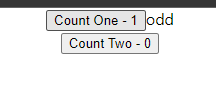
        </>

    )

}

export default Counter

.



همانطور که در کد زیر مشاهده مینمایید دیگر isEven یک متغیر میباشد نه متد

پارامتر دوم useMemo مقدار state را میگیرد که باید روش حساس باشد و اگر تغییر کرد باید اجرا شود

import React, { useMemo, useState } from 'react'

const Counter = () => {

    const [counterOne, setCounterOne] = useState(0)

    const [counterTwo, setCounterTwo] = useState(0)

    const incrementOne = () => {

        setCounterOne(counterOne + 1)

    }

    const incrementTwo = () => {

        setCounterTwo(counterTwo + 1)

    }

    const isEven = useMemo(() => {

        let i = 0

        while (i < 2000000000) i++

        return counterOne % 2 === 0

    }, [counterOne])

    return (

        <>

            <div>

                <button onClick={incrementOne}>Count One - {counterOne} </button>

                <span>{isEven ? 'even' : 'odd'}</span>

            </div>

            <div>

                <button onClick={incrementTwo}>Count Two - {counterTwo} </button>

            </div>

        </>

    )

}

export default Counter

.

فرق بین useMemo , useCallback در این هستش که در useMemo میتوانیم مقدار خروجی را ذخیره کنیم

.

# useRef

دسترسی به node های dom را بصورت مستقیم میدهد

اگر بخواهیم به node مربوط به input عمل focus را بدهیم باید بصورت زیر عمل کنیم

import React, { useState, useEffect, useRef } from 'react'

const FocusInput = () => {

const inputRef=useRef(null)

    useEffect(() => {

        inputRef.current.focus()

    }, [])

    return (

        <>

            <input ref={inputRef} type='text'/>

        </>

    )

}

export default FocusInput

.

# useRef

در این مثال به عنوان متغیر استفاده میکنیم

همانطور که میبینید به interval در render دسترسی نداریم

import React, { useState, useEffect } from 'react'

const HookTimer = () => {

    const [timer, setTimer] = useState(0)

    useEffect(() => {

        const interval = setInterval(() => {

           // setTimer(timer + 1)

             setTimer(x => x + 1)

        }, 1000)

        return () => {

            clearInterval(interval)

        }

    }, [])

    return (

        <>

            Hook Timer - {timer}

            <button onClick={() => clearInterval(interval)}></button>

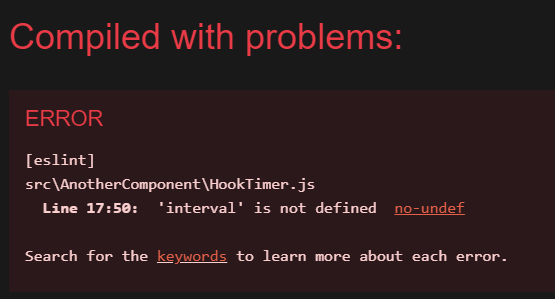
        </>

    )

}

export default HookTimer

.



.

از useRef به عنوان متغیر global استفاده میکنیم که بتوانیم در render بهش دسترسی داشته باشیم

import React, { useState, useEffect, useRef } from 'react'

const HookTimer = () => {

    const [timer, setTimer] = useState(0)

    const intervalRef = useRef()

    useEffect(() => {

        intervalRef.current = setInterval(() => {

            // setTimer(timer + 1)

            setTimer(x => x + 1)

        }, 1000)

        return () => {

            clearInterval(intervalRef.current)

        }

    }, [])

    return (

        <>

            Hook Timer - {timer}

            <button onClick={() => clearInterval(intervalRef.current)}>clearInterval</button>

        </>

    )

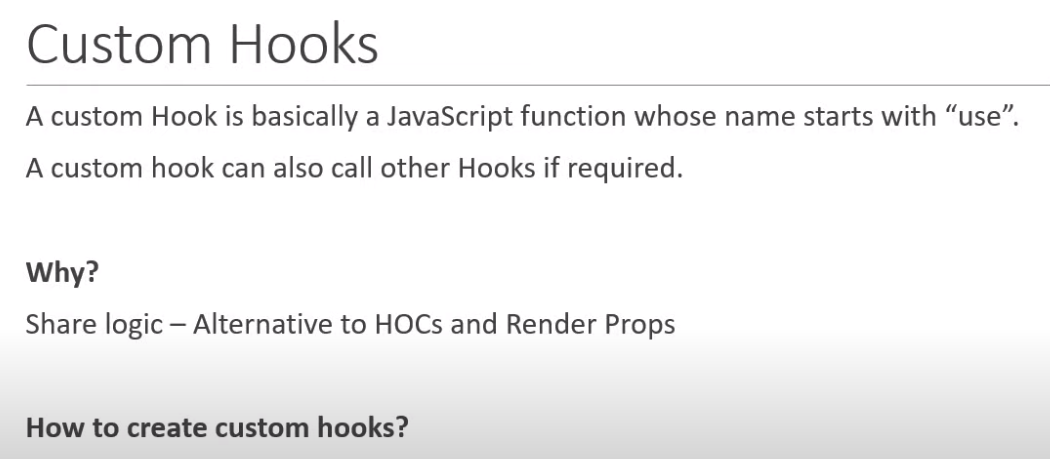
}

export default HookTimer

.

# Custom Hook

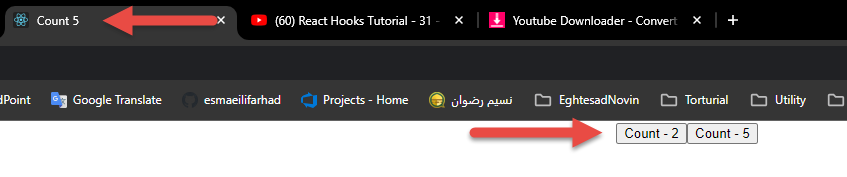
ی



.

## React Hooks Tutorial - 31 - useDocumentTitle Custom Hook

با کلیک بر روی هر دکمه مقدار title عوض میشود مشکلی که این روش دارد یک کد یکسان را در دو کامپوننت تکرار کرده ایم برای حل این مشکل از custom hook استفاده میکنیم



.

import React, { useState, useEffect } from 'react'

const DocTitleOne = () => {

    const [count, setCount] = useState(0)

    useEffect(() => {

        document.title = `Count ${count}`

    }, [count])

    return (

        <>

            <button onClick={() => setCount(count + 1)}>Count - {count}</button>

        </>

    )

}

export default DocTitleOne

.

import React, { useState, useEffect } from 'react'

const DocTitleTwo = () => {

    const [count, setCount] = useState(0)

    useEffect(() => {

        document.title = `Count ${count}`

    }, [count])

    return (

        <>

            <button onClick={() => setCount(count + 1)}>Count - {count}</button>

        </>

    )

}

export default DocTitleTwo

بصورت زیر یک customHook ایجاد میکنیم و در کامپوننت های دیگر ازش استفاده میکنیم

دقت کن render ندارد

import { useEffect } from 'react'

const useDocumentTitle = (count) => {

    useEffect(() => {

        document.title = `Count ${count}`

    }, [count])

}

export default useDocumentTitle

.

import React, { useState, useEffect } from 'react'

import useDocumentTitle from './UseDocumentTitle'

const DocTitleOne = () => {

    const [count, setCount] = useState(0)

    useDocumentTitle(count)

    return (

        <>

            <button onClick={() => setCount(count + 1)}>Count - {count}</button>

        </>

    )

}

export default DocTitleOne

.

import React, { useState, useEffect } from 'react'

import useDocumentTitle from './UseDocumentTitle'

const DocTitleTwo = () => {

    const [count, setCount] = useState(0)

    useDocumentTitle(count)

    return (

        <>

            <button onClick={() => setCount(count + 1)}>Count - {count}</button>

        </>

    )

}

export default DocTitleTwo

.

## مثال دوم

نکته هایی که این CustomHook داره به شرح ذیل است :

1. مقدار initialState بصورت پیشفرض صفر هستش
2. مقدار intialState , Value بصورت داینامیک قرار گرفته شده است
3. خروجی ارایه ای از state ها و متد هاست

import { useState } from 'react'

const useCounter = (initialCount = 0,value) => {

    const [count, setCount] = useState(initialCount)

    const increament = () => {

        setCount(x => x + value)

    }

    const decreament = () => {

        setCount(x => x - value)

    }

    const rest = () => {

        setCount(0)

    }

    return [count, increament, decreament, rest]

}

export default useCounter

به صورت زیر از customHook استفاده شده است

import React, { useState, useEffect } from 'react'

import useCounter from './useCounter'

const CounterOne = () => {

    const [count, increament, decreament, rest] = useCounter(0,1)

    return (

        <>

            <h2>Count = {count}</h2>

            <button onClick={increament}>Increament</button>

            <button onClick={decreament}>decreament</button>

            <button onClick={rest}>rest</button>

        </>

    )

}

export default CounterOne

.

import React, { useState, useEffect } from 'react'

import useCounter from './useCounter'

const CounterTwo = () => {

    const [count, increament, decreament, rest] = useCounter(10,10)

    return (

        <>

            <h2>Count = {count}</h2>

            <button onClick={increament}>Increament</button>

            <button onClick={decreament}>decreament</button>

            <button onClick={rest}>rest</button>

        </>

    )

}

export default CounterTwo

.

## مثال سوم

در این مثال برخی برای عمل بایند کردن دیتا و ست کردن value به state از customHook استفاده شده است

import { useState } from 'react'

const useInput = (initialValue) => {

    const [value, setValue] = useState(initialValue)

    const reset = () => {

        setValue(initialValue)

    }

    const bind = {

        value,

        onChange: e => {

            setValue(e.target.value)

        }

    }

    return [value, bind, reset]

}

export default useInput

.

import React, { useState, useEffect } from 'react'

import useInput from './useInput'

const UserForm = () => {

    const [firstName, bindFirstName, resetFirstName] = useInput('')

    const [lastName, bindLastName, resetLastName] = useInput('')

    const submitHandler = e => {

        //dont refresh page

        e.preventDefault()

        alert(`Hello ${firstName} ${lastName}`)

        resetFirstName()

        resetLastName()

    }

    return (

        <>

            <form onSubmit={submitHandler}>

                <div>

                    <label>First Name</label>

                    <input

                        {...bindFirstName}

                        type='text'

                    />

                </div>

                <div>

                    <label>Last Name</label>

                    <input

                        {...bindLastName}

                        type='text'

                    />

                </div>

                <button>Submit</button>

            </form>

        </>

    )

}

export default UserForm

.

# روش نوشتن javaScript توی React

.

# React CRUD

# react-router-dom

اول باید react-router-dom را نصب کنیم

import ListEmployee from './AnotherComponent/CRUD/ListEmployee';

import 'bootstrap/dist/css/bootstrap.min.css'

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import CreateEmployee from './AnotherComponent/CRUD/CreateEmployee';

import EditEmployee from './AnotherComponent/CRUD/EditEmployee';

function App() {

  return (

    <div className="App">

      <Router>

        <Routes>

          <Route exact path='/' element={<ListEmployee />} />

          <Route path='/create' element={<CreateEmployee />} />

          <Route path='/edit' element={<EditEmployee/>} />

        </Routes>

      </Router>

    </div>

  );

}

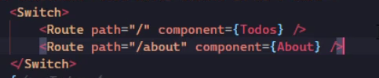
export default App;

..........

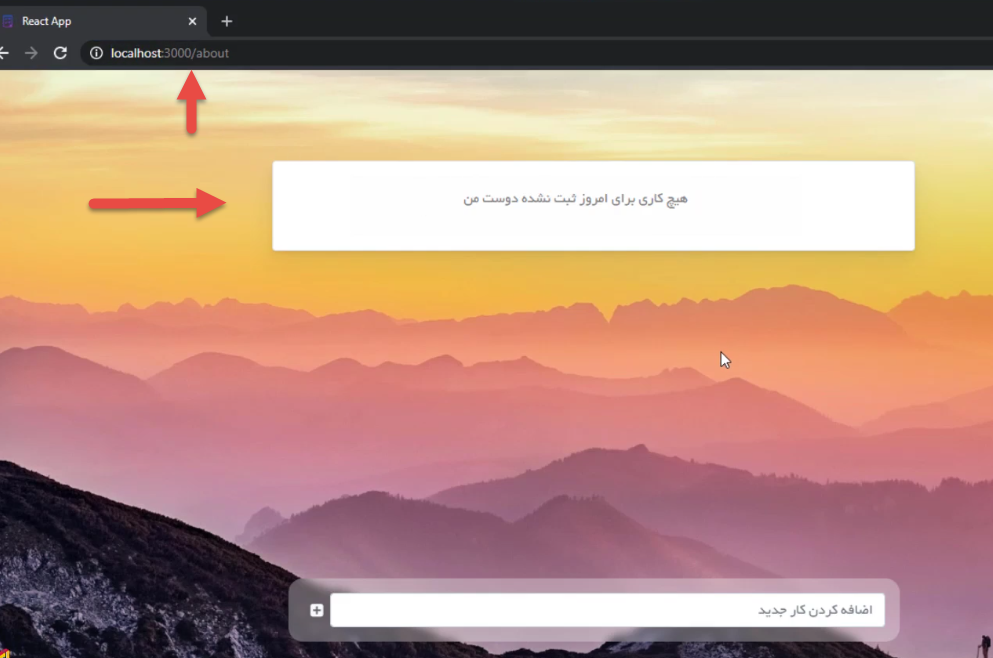
#### Switch

نکته خیلی مهم

Switch اولین مسیری که مطابقت کند را نمایش میدهد

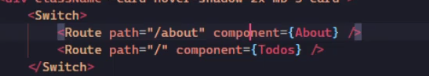


دقت کن چونکه در مسیر / قرار داده شده با اینکه ما آدرس /about را خواسته ایم اما دارد todo را نمایش میدهد



یک روش برای حل مشکل بالا این است که مسیر / خالی را اخرسر قرار دهیم

.



#### exact

اما روش بهتر استفاده از exact است

Exact : دقیق : یعنی دقیقا همان باشد

                        <Switch>

                            <Route path="/" exact component={Todos} />

                            <Route path="/about" component={About} />

                        </Switch>

## ListEmployee

نکته خیلی مهم روش ارسال دیتا از link به کامپوننت در مثال زیر همانطور که مشاهده میکنید ما یک object دیتا را از link به کامپوننت EditEmployee ارسال میکنیم و با استفاده از useLocation در کامپوننت EditEmployee دیتا را دریافت میکنیم

<Link to={`/edit`} state={{ id: item.id, name: item.Name }} >

 <button className='btn btn-success' >Edit</button>

</Link>

نکته مهم دوم پس از عمل delete با استفاده از useNavigate به صفحه اصلی باز میگردیم

let history = useNavigate()

    const handleDelete = (id) => {

        var index = DataEmployee.map(function (e) {

            return e.id

        }).indexOf(id)

        DataEmployee.splice(index, 1)

        history('/')

    }

import React, { useState, useEffect } from 'react'

import DataEmployee from './DataEmployee'

import { useNavigate, Link } from 'react-router-dom'

const ListEmployee = () => {

    let history = useNavigate()

    const handleDelete = (id) => {

        var index = DataEmployee.map(function (e) {

            return e.id

        }).indexOf(id)

        DataEmployee.splice(index, 1)

        history('/')

    }

    return (

        <>

            <Link to="/create">

                <button className='btn btn-info' style={{ size: 'lg' }}>Create</button>

            </Link>

            <br />

            <table className='table table-bordered'>

                <tr>

                    <th>Name</th>

                    <th>Age</th>

                    <th>Operation</th>

                </tr>

                {

                    DataEmployee && DataEmployee.length > 0

                        ?

                        DataEmployee.map((item) => {

                            return (

                                <tr key={item.id}>

                                    <td>

                                        {item.Name}

                                    </td>

                                    <td>

                                        {item.Age}

                                    </td>

                                    <td>

                                        <button className='btn btn-danger' onClick={() => handleDelete(item.id)}>Delete{item.id}</button>

                                        <span> | </span>

                                        <Link to={`/edit`} state={{ id: item.id, name: item.Name }} >

                                            <button className='btn btn-success' >Edit</button>

                                        </Link>

                                    </td>

                                </tr>

                            )

                        })

                        :

                        "No Data"

                }

            </table>

        </>

    )

}

export default ListEmployee

## CreateEmployee

import React, { useState, useEffect } from 'react'

import { Form, useNavigate } from 'react-router-dom'

import DataEmployee from './DataEmployee'

const CreateEmployee = () => {

    const [name, setName] = useState('')

    const [age, setAge] = useState('')

    let history = useNavigate()

    useEffect(() => {

    }, [])

    const handleSubmit = (e) => {

        e.preventDefault();

        let uniqueId = Math.floor(Math.random() \* 1000000);

        let a = name, b = age

        DataEmployee.push({ id: uniqueId, Name: a, Age: b })

        history("/")

    }

    return (

        <>

            <div className=''>

                <div className='form-group'>

                    <input type='text' className='form-control' placeholder='enter name' required onChange={(e) => setName(e.target.value)} />

                </div>

                <div className='form-group'>

                    <input type='number' className='form-control' placeholder='enter age' required onChange={(e) => setAge(e.target.value)} />

                </div>

                <button onClick={(e) => handleSubmit(e)} type='submit'>Submit</button>

            </div>

        </>

    )

}

export default CreateEmployee

## EditEmployee

import React, { useState, useEffect } from 'react'

import { useLocation, useNavigate } from 'react-router-dom'

import DataEmployee from './DataEmployee'

const EditEmployee = (props) => {

    const [name, setName] = useState("")

    const [age, setAge] = useState("")

    const [id, setId] = useState("")

    // [oldData, setOldData] = useState({})

    const location = useLocation()

    let history = useNavigate();

    var index = DataEmployee.map(function (e) {

        return e.id

    }).indexOf(id)

    const handleSubmit = (e) => {

        e.preventDefault();

        const id = location.state.id

        const old = DataEmployee.find(x => x.id == id)

        old.Name = name

        old.Age = age

        history("/")

    }

    useEffect(() => {

        const id = location.state.id

        const old = DataEmployee.find(x => x.id == id)

        setName(old.Name)

        setAge(old.Age)

        debugger

    }, [])

    return (

        <>

            <p>ویرایش اطلاعات {location.state.name}</p>

            <p>EditEmployee</p>

            <div className=''>

                <div className='form-group'>

                    <input type='text' value={name} className='form-control' placeholder='enter name' required onChange={(e) => setName(e.target.value)} />

                </div>

                <div className='form-group'>

                    <input type='number' value={age} className='form-control' placeholder='enter age' required onChange={(e) => setAge(e.target.value)} />

                </div>

                <button onClick={(e) => handleSubmit(e)} type='submit'>Submit</button>

            </div>

        </>

    )

}

export default EditEmployee

.

.............

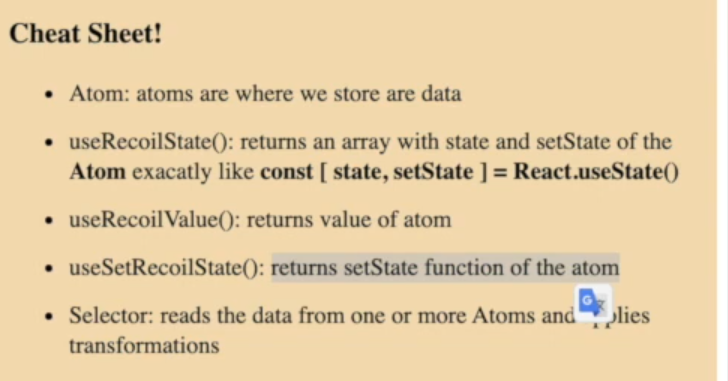
# Recoil

ایجاد یک Global State

هر قسمت از اطلاعات در یک object ذخیره میشوند که به آن اتم میگویند

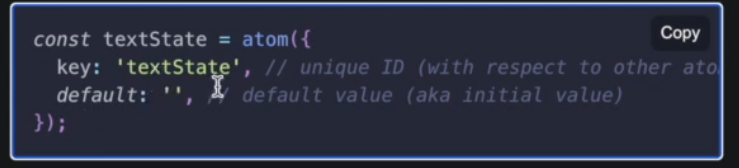
اتم را سعی میکنیم کوچکترین جز در نظر بگیریم

.



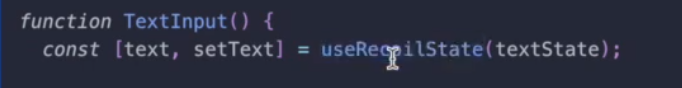
.

برای ساخت اتم بصورت زیر عمل مکینیم



.

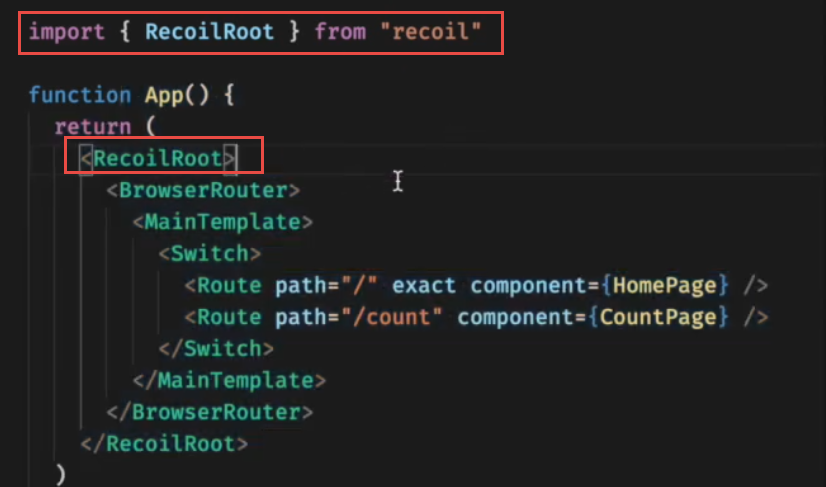
برای خواندن مقدار و نوشتن مقدار اتم بصورت زیر useRecoilState استفاده میکنیم





..........

مثال

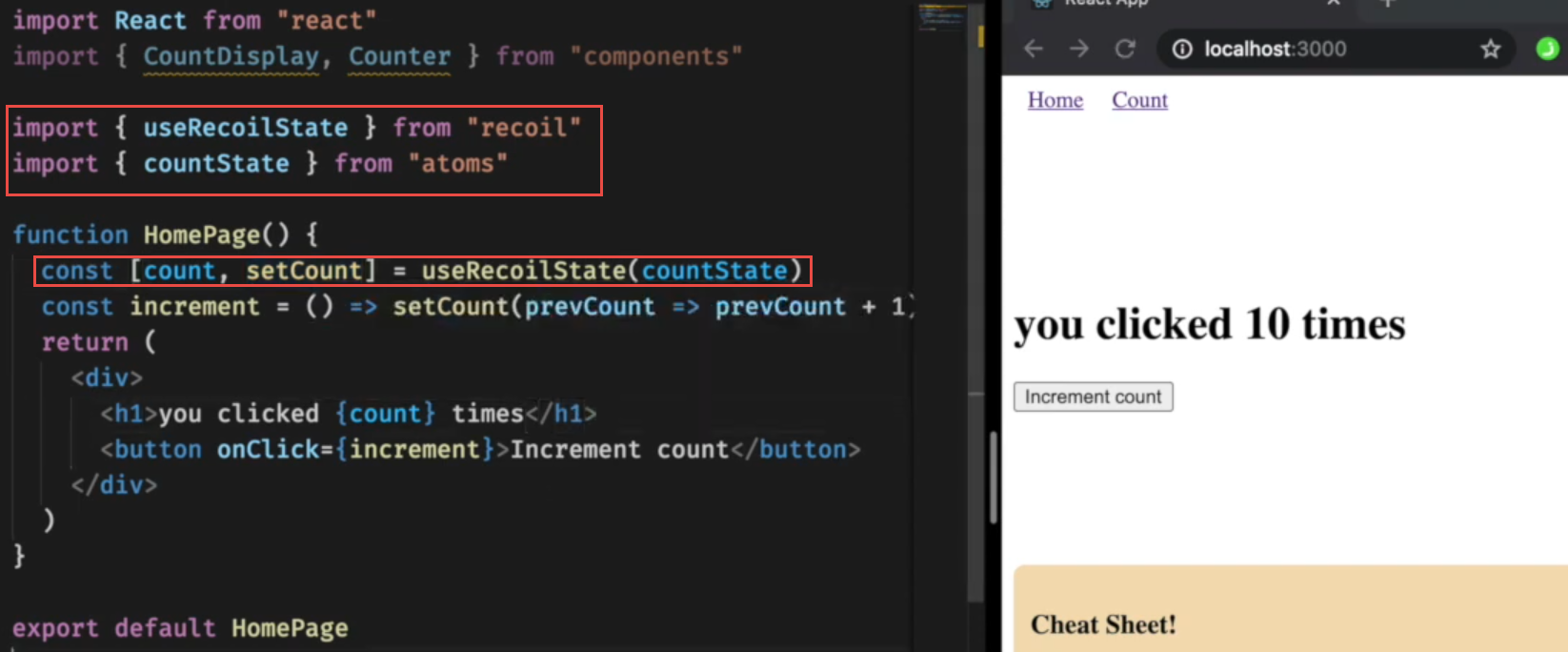


ساخت اتم



.

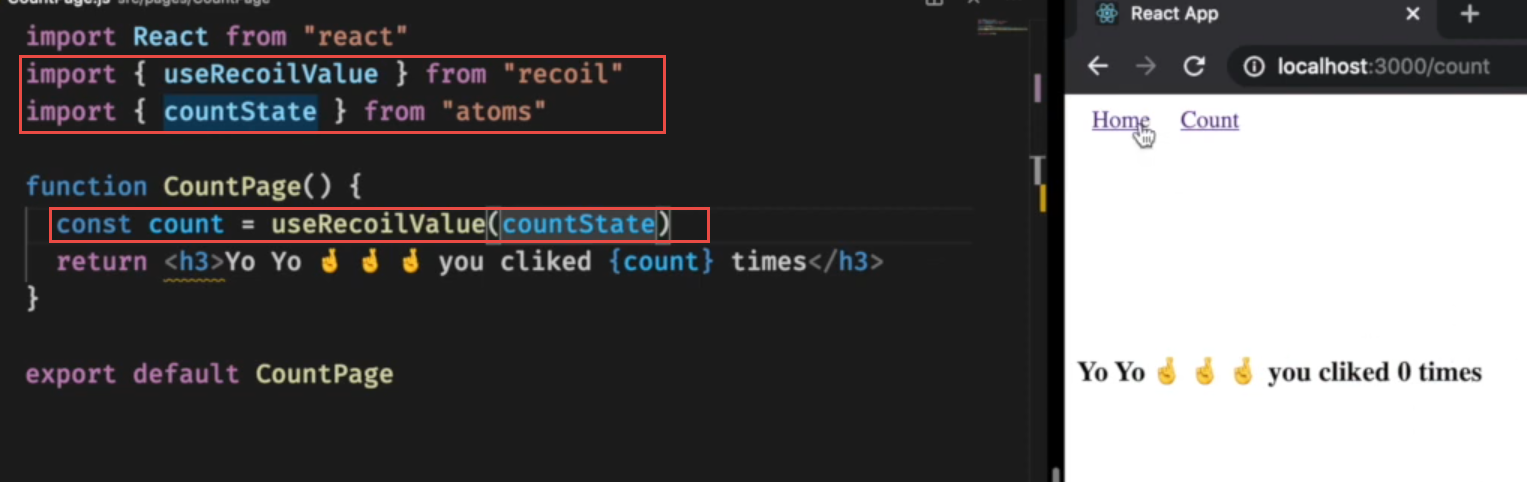
و به صورت زیر مقدار را میخوانیم و مینویسیم



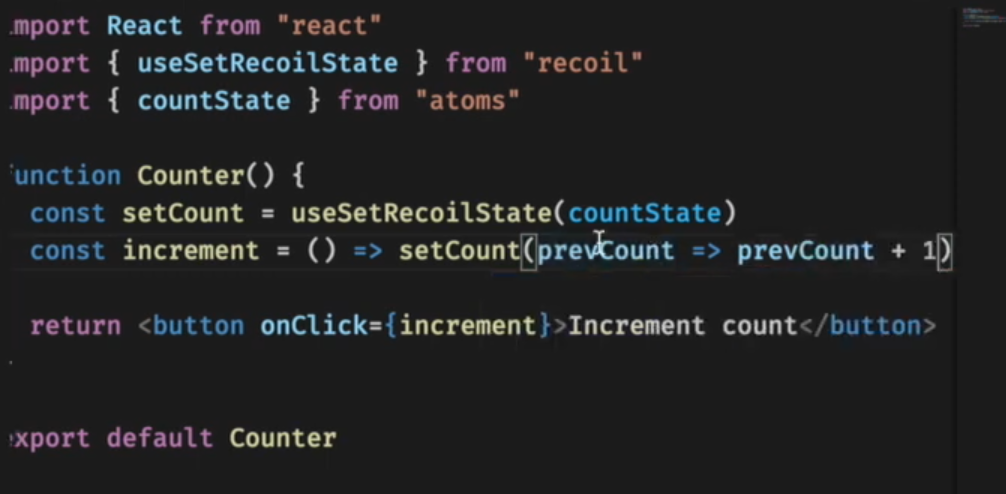
.

و در کامپوننت دیگر فقط مقدار را از اتم میخوانیم

useRecoilValue فقط مقدار را به ما میدهد دیگر امکان نوشتن را نمیدهد ولی useRecoilState میشد نوشت و خواند



...



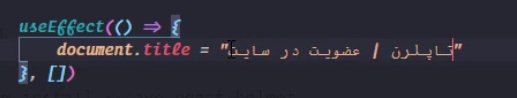
...

# ReactHelmet

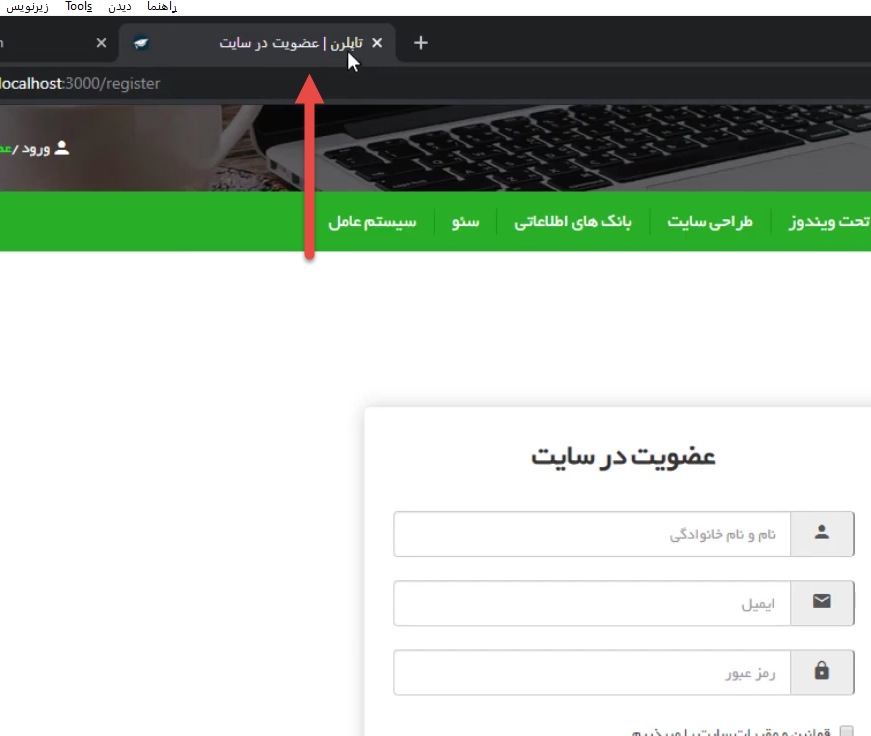
باعث میشود عنوان تب عوض بشود

البته با js هم میتوان اینکار را کرد برای اینکار باید در useeffect برای بار اول که اجرا میشود بصورت زیر عمل کرد

از useEffect استفاده میکنیم چون میخواهیم فقط یکبار اتفاق بیفته آرایه خالی بهش میدهیم



.



بصورت زیر نصب میشود

Npm I react-helmet

.............

import Helmet from "react-helmet";

const MainLayout = props => {

    const { pathname } = props.location;

    return (

        <Fragment>

            <Helmet>

                <title>خودآموز تاپلرن</title>

            </Helmet>

.

# Preloader

برای اینکه وقتی منتظر دیتا هستی چیزی را نمایش بدهد تا بفهمیم منتظر سرور هستیم

فعلا اموزشی نداریم

# Redux Toolkit

همه موارد که لازم است با redux کار کنیم در یک library جمع شده است

<https://redux-toolkit.js.org/tutorials/quick-start>

.

# React Router

## روش نصب

Npm I react-router-dom@6

.

## BrowserRouter

سپس باید BrowserRouter را ایجاد کنیم

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { BrowserRouter } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <BrowserRouter>

      <App />

    </BrowserRouter>

  </React.StrictMode>

);

reportWebVitals();

## Routes , Route

import logo from './logo.svg';

import './App.css';

import { Route, Routes } from 'react-router-dom';

import Home from './Components/Home';

import About from './Components/About';

function App() {

  return (

    <>

    <Routes>

      <Route path='/' element={<Home/>}/>

      <Route path='about' element={<About/>}/>

    </Routes>

    </>

  );

}

export default App;

## link

import React, { useState, useEffect } from 'react'

import { Link } from 'react-router-dom'

const Navbar = () => {

    return (

        <>

            <nav>

                <Link to='/'>Home</Link>

                <Link to='about'>About</Link>

            </nav>

        </>

    )

}

export default Navbar

.

import logo from './logo.svg';

import './App.css';

import { Route, Routes } from 'react-router-dom';

import Home from './Components/Home';

import About from './Components/About';

import Navbar from './Components/Navbar';

function App() {

  return (

    <>

    <Navbar/>

    <Routes>

      <Route path='/' element={<Home/>}/>

      <Route path='about' element={<About/>}/>

    </Routes>

    </>

  );

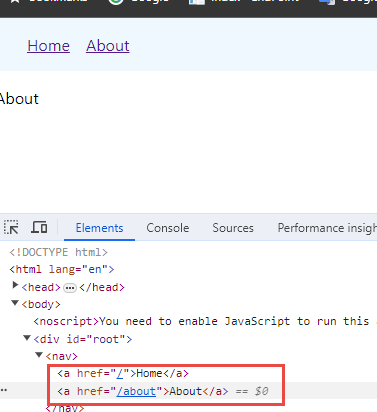
}

export default App;

.

## active link

در link کلاس اکتیو مشخص نیست ولی در NavLink مشخص است

.

.

import React, { useState, useEffect } from 'react'

import { Link, NavLink } from 'react-router-dom'

const Navbar = () => {

    return (

        <>

            <nav>

                {/\* <Link to='/'>Home</Link>

                <Link to='about'>About</Link> \*/}

                <NavLink to='/'>Home</NavLink>

                <NavLink to='about'>About</NavLink>

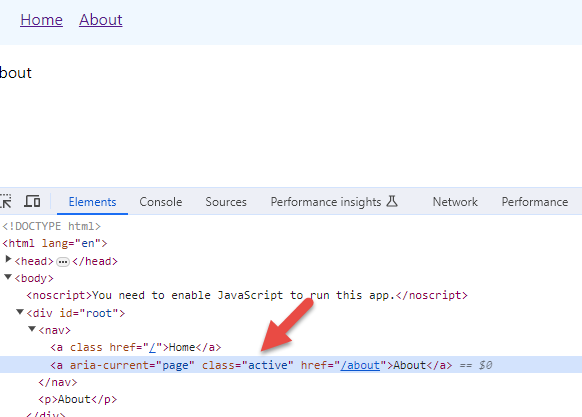
            </nav>

        </>

    )

}

export default Navbar



.

و میتوانیم در فایل css تغییرات زیر را بدهیم تا لینک فعال تغییر کند

nav a.active{

  text-decoration: none;

  font-weight: bold;

}

روش دوم تغییر لینک فعال

import React, { useState, useEffect } from 'react'

import { Link, NavLink } from 'react-router-dom'

const Navbar = () => {

    const navLinkStyle=({isActive})=>{

        return {

            fontWeight:isActive?'bold':'normal',

            textDecoration:isActive?'none':'underline'

        }

    }

    return (

        <>

            <nav>

                {/\* <Link to='/'>Home</Link>

                <Link to='about'>About</Link> \*/}

                <NavLink style={navLinkStyle}  to='/'>Home</NavLink>

                <NavLink style={navLinkStyle}  to='about'>About</NavLink>

            </nav>

        </>

    )

}

export default Navbar

.

## useNavigate

.



.

import logo from './logo.svg';

import './App.css';

import { Route, Routes } from 'react-router-dom';

import Home from './Components/Home';

import About from './Components/About';

import Navbar from './Components/Navbar';

import OrderSummary from './Components/OrderSummary';

function App() {

  return (

    <>

    <Navbar/>

    <Routes>

      <Route path='/' element={<Home/>}/>

      <Route path='about' element={<About/>}/>

      <Route path='order-summary' element={<OrderSummary/>}/>

    </Routes>

    </>

  );

}

export default App;

Replcae:true باعث میشود که به محل اول بازگردد.

import React, { useState, useEffect } from 'react'

import { useNavigate } from 'react-router-dom'

const Home = () => {

const navigate= useNavigate()

    return (

        <>

            <p>Home</p>

            <button onClick={()=>navigate('order-summary',{replace:true})}>place order</button>

        </>

    )

}

export default Home

.

import React, { useState, useEffect } from 'react'

import { useNavigate } from 'react-router-dom'

const OrderSummary = () => {

const navigate=useNavigate()

    return (

        <>

            <p>Order confirmed!</p>

            <button onClick={()=>navigate(-1)}>go back</button>

        </>

    )

}

export default OrderSummary

.

## useNavigate

نکته مهم دوم پس از عمل delete با استفاده از useNavigate به صفحه اصلی باز میگردیم

let history = useNavigate()

    const handleDelete = (id) => {

        var index = DataEmployee.map(function (e) {

            return e.id

        }).indexOf(id)

        DataEmployee.splice(index, 1)

        history('/')

    }

.

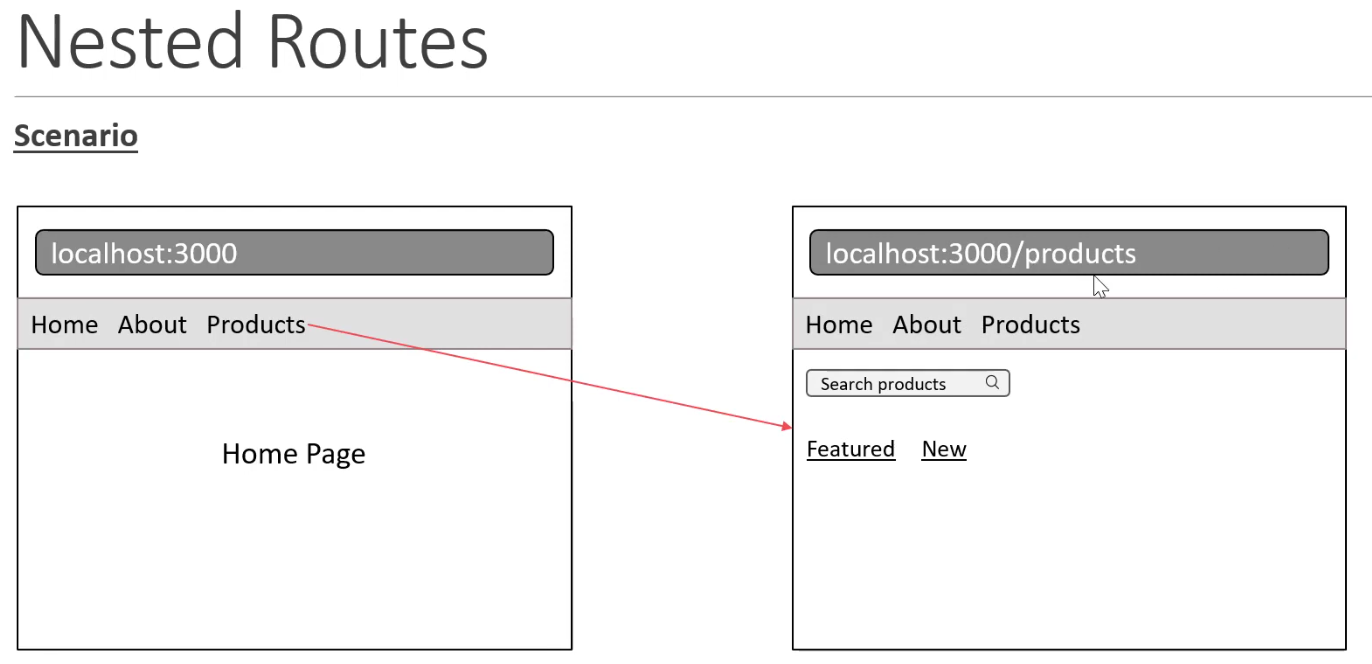
## No Match Route

ی

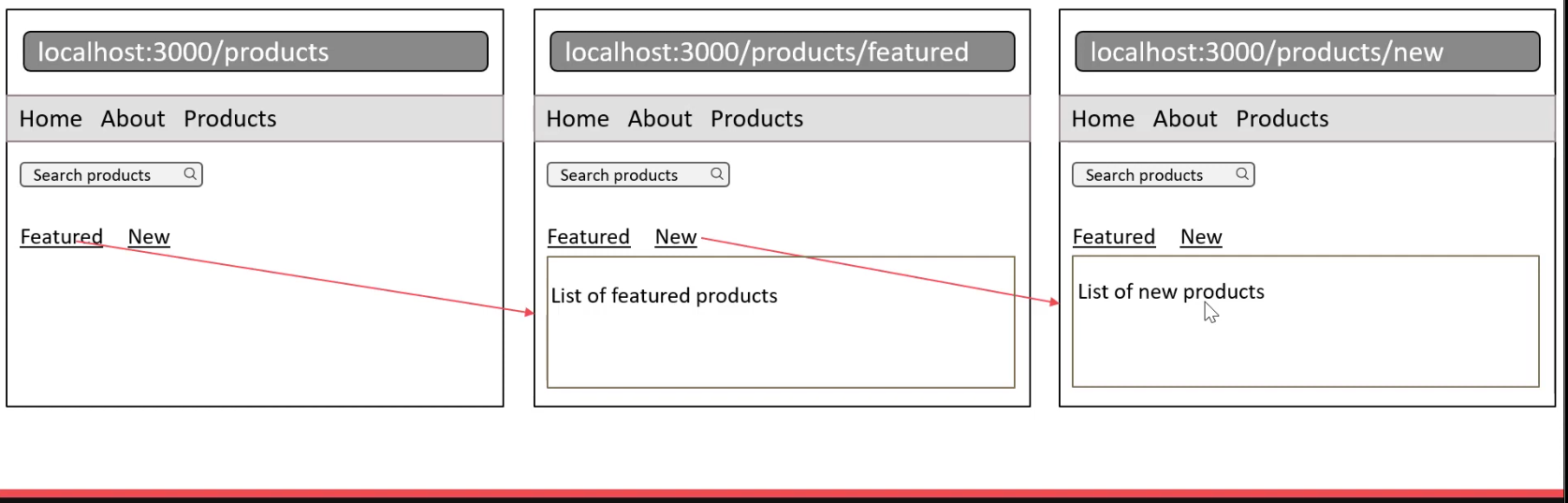
        <Route path='\*' element={<NoMatch />} />

## Nested Routes

ی



.



ی

 <Routes>

        <Route path='/' element={<Home />} />

        <Route path='about' element={<About />} />

        <Route path='order-summary' element={<OrderSummary />} />

        <Route path='products' element={<Products />}>

          <Route path='featured' element={<FeatureProducts />} />

          <Route path='new' element={<NewProducts />} />

        </Route>

        <Route path='\*' element={<NoMatch />} />

 </Routes>

.

به outlet دقت کن برای رندر شدن child ها اضافه میشود و در پدر نوشته میشود مثلا در اینجاد در کامپوننتن پدر یعنی products کلمه outlet نوشته شده تا فرزند ها در این قسمت رندر شوند

import React, { useState, useEffect } from 'react'

import { Link, Outlet } from 'react-router-dom'

const Products = () => {

    return (

        <>

            <input type='search' placeholder='search products' />

            <nav>

                <Link to='featured'>Featured</Link>

                <Link to='new'>New</Link>

            </nav>

            <Outlet/>

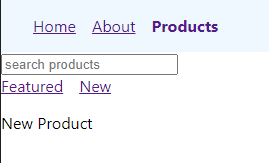
        </>

    )

}

export default Products

.



.

## Index Route

برای اینکه در آدرس پدر فرزند را هم نمایش دهیم بصورت زیر عمل میکنیم

  <Routes>

        <Route path='/' element={<Home />} />

        <Route path='about' element={<About />} />

        <Route path='order-summary' element={<OrderSummary />} />

        <Route path='products' element={<Products />}>

          <Route index element={<Home />} />

          <Route path='featured' element={<FeatureProducts />} />

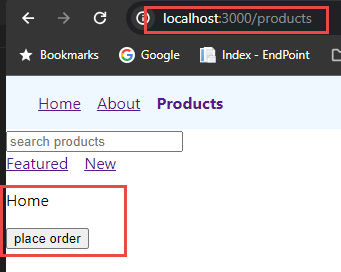
          <Route path='new' element={<NewProducts />} />

        </Route>

        <Route path='\*' element={<NoMatch />} />

      </Routes>

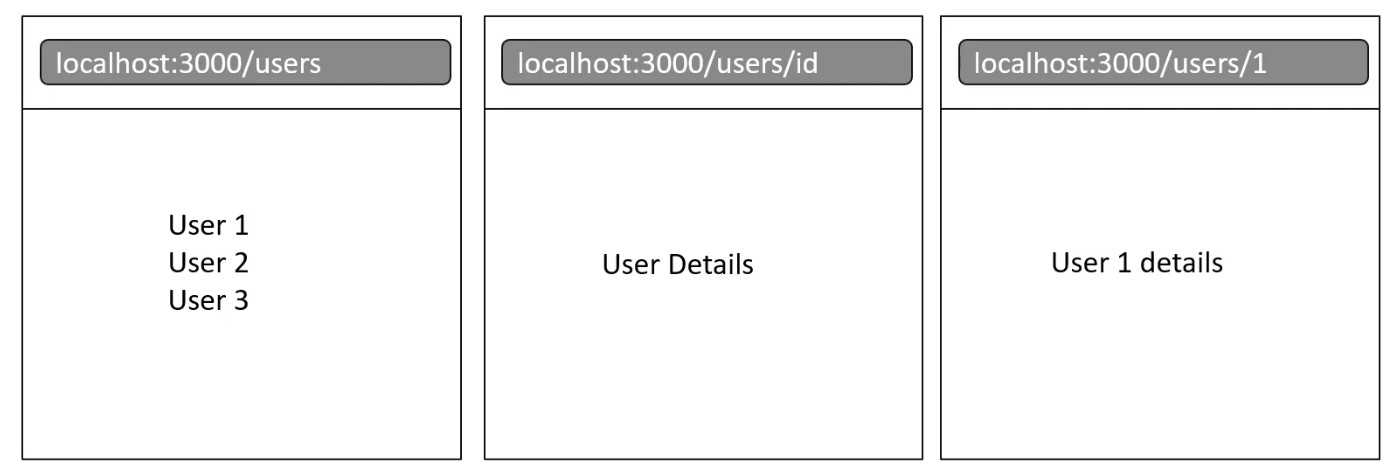
.



.

## Dynamic routes

ی



.

 <Routes>

        <Route path='/' element={<Home />} />

        <Route path='users' element={<Users />} />

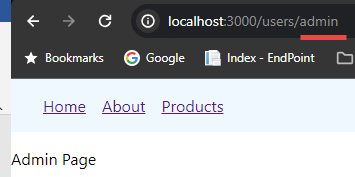
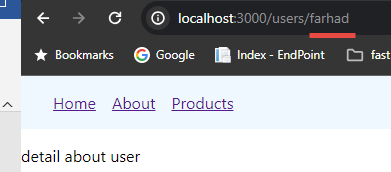
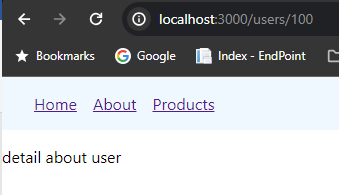
        <Route path='users/:userId' element={<UserDetails />} />

        <Route path='users/Admin' element={<Admin />} />

        <Route path='\*' element={<NoMatch />} />

      </Routes>

.



.

دقت کن وقتی route فرزند میشود path='users/Admin' علامت اسلش / برداشته میشود path='Admin'

 <Route path='users' element={<Users />}>

          <Route path=':userId' element={<UserDetails />} />

          <Route path='Admin' element={<Admin />} />

 </Route>

Outlet برای این هستش که کامپوننت ها فرزند تو دل پدر لود شوند

import React, { useState, useEffect } from 'react'

import { Outlet } from 'react-router-dom'

const Users = () => {

    return (

        <>

            <h2>User 1</h2>

            <h2>User 2</h2>

            <h2>User 3</h2>

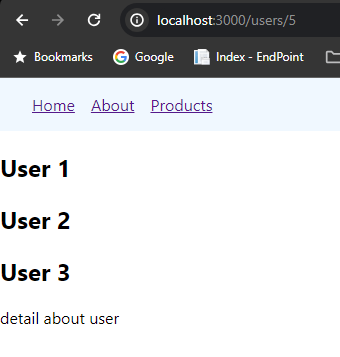
            <Outlet/>

        </>

    )

}

export default Users



.

## useParams

.

          <Route path=':userId' element={<UserDetails />} />

به صورت زیر میتوانیم مقدار userId را دریافت کنیم

.

import React, { useState, useEffect } from 'react'

import { useParams } from 'react-router-dom'

const UserDetails = () => {

    const params = useParams()

    const userId = params.userId

    return (

        <>

            <p>detail about user {userId}</p>

        </>

    )

}

export default UserDetails

. .

## useLocation

نکته خیلی مهم روش ارسال دیتا از link به کامپوننت در مثال زیر همانطور که مشاهده میکنید ما یک object دیتا را از link به کامپوننت ارسال میکنیم و با استفاده از useLocation در کامپوننت دیتا را دریافت میکنیم

  <Link to='featured' state={{ id: 1, lastName: 'اسماعیلی', name: 'فرهاد' }} >Featured</Link>

روش دریافت state

import React, { useState, useEffect } from 'react'

import { useLocation } from 'react-router-dom'

const FeatureProducts = () => {

    const location = useLocation()

    return (

        <>

            <p>List Of Feeature Products {location.state.lastName}  {location.state.name}</p>

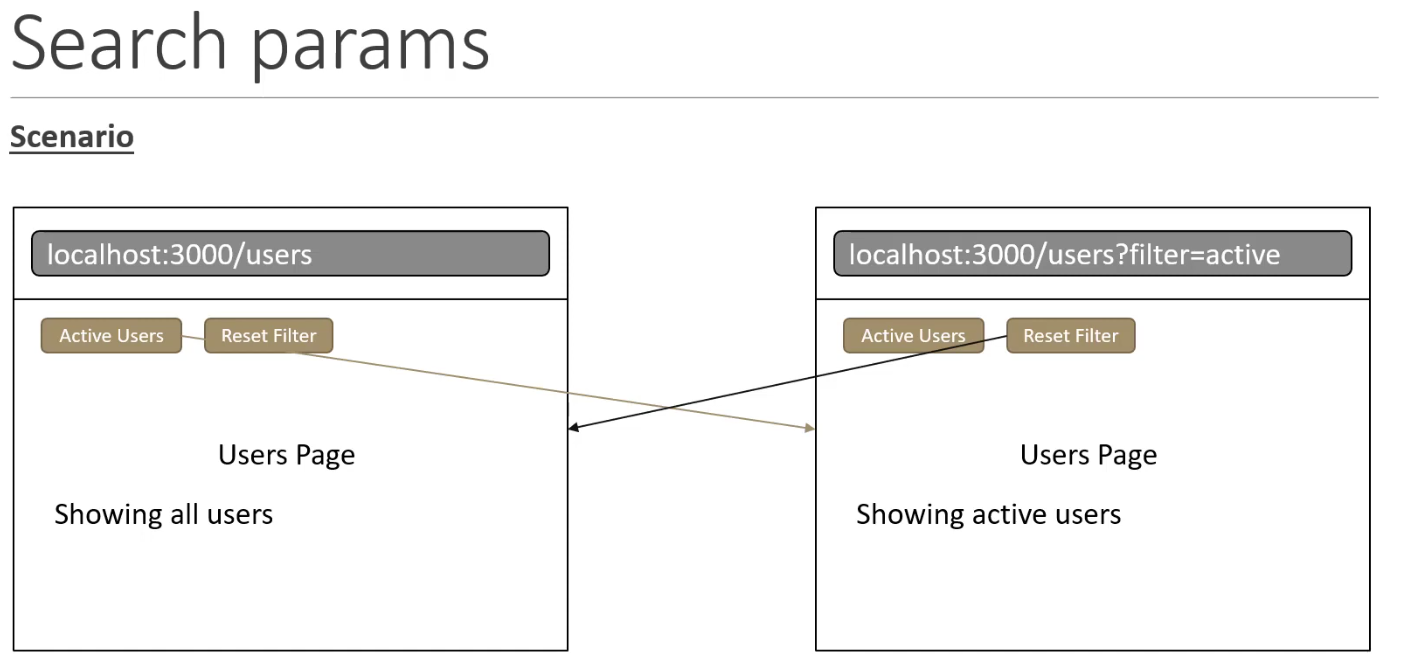
        </>

    )

}

export default FeatureProducts

## Search params



.

import React, { useState, useEffect } from 'react'

import { Outlet, useSearchParams } from 'react-router-dom'

const Users = () => {

const [searchParams,setSearchParams]=useSearchParams()

const showActiveUsers=searchParams.get('filter')==='active'

    return (

        <>

            <h2>User 1</h2>

            <h2>User 2</h2>

            <h2>User 3</h2>

            <Outlet />

            <div>

                <button onClick={()=>setSearchParams({filter:'active'})}>Active User</button>

                <button onClick={()=>setSearchParams()}>Reset Filter</button>

            </div>

            {

                showActiveUsers?<h2>show active users</h2>:<h2>showing all users</h2>

            }

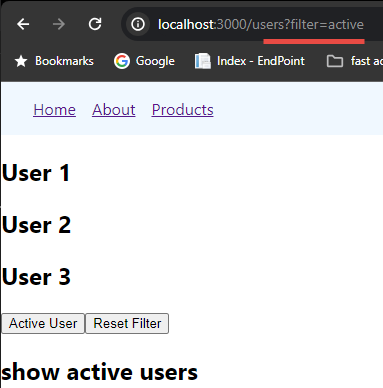
        </>

    )

}

export default Users

.



## Relative links

 <Routes>

        <Route path='/' element={<Home />} />

        <Route path='products' element={<Products />}>

          <Route index element={<Home />} />

          <Route path='featured' element={<FeatureProducts />} />

          <Route path='new' element={<NewProducts />} />

        </Route>

        <Route path='\*' element={<NoMatch />} />

      </Routes>

.

                <Link to='/products/new'>New</Link>

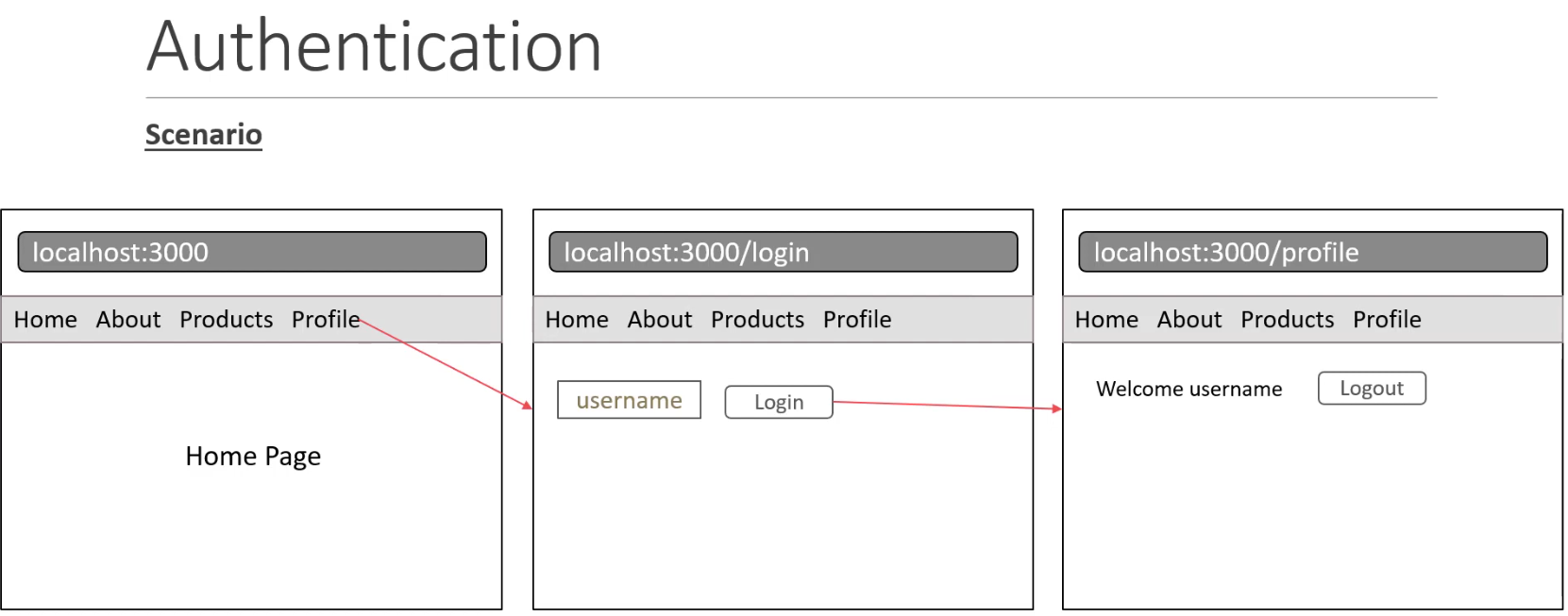
میتوان بصورت زیر لینک را خلاصه تر کرد واین میشه relative لینک

                <Link to='new'>New</Link>

....

## Authentication

ب



### App

1. کامپوننت AuthProvider را اول مینوسیم که اطلاعات آن را در زیر مجموعه های خودش Share کنیم به کمک Context که در AuthProvider تعریف کرده ایم
2. کامپوننت Profile قبل از اینکه بخواد اجرا بشه کامپوننت RequireAuth اجرا میشود

function App() {

  return (

    <>

      <AuthProvider>

        <Navbar />

        <Routes>

          <Route path='/' element={<Home />} />

          <Route path='about' element={<About />} />

          <Route path='order-summary' element={<OrderSummary />} />

          <Route path='products' element={<Products />}>

            <Route index element={<Home />} />

            <Route path='featured' element={<FeatureProducts />} />

            <Route path='new' element={<NewProducts />} />

          </Route>

          <Route path='users' element={<Users />}>

            <Route path=':userId' element={<UserDetails />} />

            <Route path='Admin' element={<Admin />} />

          </Route>

          <Route path='profile' element={<RequireAuth><Profile /></RequireAuth>} />

          <Route path='login' element={<Login />} />

          <Route path='\*' element={<NoMatch />} />

        </Routes>

      </AuthProvider>

    </>

  );

}

export default App;

### AuthProvider

1. ابتدا یک context تعریف کرده و user login logout را در بین همه مشترک کرده . و همه در app همه زیر مجموعه این authProvider هستند
2. هر کامپوننتی که بخواد به user , login , logout دسترسی داشته باشه باید useAuth را صدا بزنه

import React, { useState,  createContext, useContext } from 'react'

const AuthContext = createContext(null)

export const AuthProvider = ({ children }) => {

    const [user, setUser] = useState(null)

    const login = (user) => {

        setUser(user)

    }

    const logout = () => {

        setUser(null)

    }

    return ( <AuthContext.Provider value={{ user, login, logout }}>

        {children}

        </AuthContext.Provider>

    )

}

export const useAuth=()=>{

    return useContext(AuthContext)

}

### RequireAuth

1. اطلاعات منویی که روش کلیک شده در صورتی که لاگین نشده باشد به کامپوننت login به کمک state ارسال میکند

import React, { useState, useEffect } from 'react'

import { useAuth } from './Auth'

import { Navigate, useLocation } from 'react-router-dom'

const RequireAuth = ({ children }) => {

    const auth = useAuth()

    const location = useLocation()

    if (!auth.user) {

        return <Navigate to='/login' state={{ path: location.pathname }} />

    }

    return children

}

export default RequireAuth

.

### Navbar

1. اگر user وجود نداست منو login را بهش نمایش میدهیم

import React, { useState, useEffect } from 'react'

import { Link, NavLink } from 'react-router-dom'

import { useAuth } from './Auth'

const Navbar = () => {

    const navLinkStyle = ({ isActive }) => {

        return {

            fontWeight: isActive ? 'bold' : 'normal',

            textDecoration: isActive ? 'none' : 'underline'

        }

    }

    const auth = useAuth()

    return (

        <>

            <nav className='primary-nav'>

                <NavLink style={navLinkStyle} to='/'>Home</NavLink>

                <NavLink style={navLinkStyle} to='about'>About</NavLink>

                <NavLink style={navLinkStyle} to='products'>Products</NavLink>

                <NavLink style={navLinkStyle} to='profile'>Profile</NavLink>

                {!auth.user && (

                    <NavLink style={navLinkStyle} to='login'>Login</NavLink>

                )}

            </nav>ّ

        </>

    )

}

export default Navbar

### Login

1. Location.state را از [RequireAuth](#_RequireAuth) میگیریم.
2. Replcae:true باعث میشود وقتی بر روی go back مرورگر کلیک کردیم دیگر وارد صفحه لاگین نشود.

import React, { useState, useEffect } from 'react'

import { useAuth } from './Auth'

import { useLocation, useNavigate } from 'react-router-dom'

const Login = () => {

    const [user, setUser] = useState('')

    const auth = useAuth()

    const navigate = useNavigate()

    const location = useLocation()

    const redirectPath = location.state?.path || '/'

    const handleLogin = () => {

        auth.login(user)

        navigate(redirectPath, { replace: true })

    }

    return (

        <>

            <label>

                Username:{''}

                <input type='text' onChange={(e) => setUser(e.target.value)} />

            </label>

            <button onClick={handleLogin}>Login</button>

        </>

    )

}

export default Login

### Profile

import React, { useState, useEffect } from 'react'

import { useAuth } from './Auth'

import { useNavigate } from 'react-router-dom'

const Profile = () => {

    const auth = useAuth()

    const navigate=useNavigate()

    const handleLogout=()=>{

        auth.logout()

        navigate('/')

    }

    return (

        <>

            <p>welcome {auth.user}</p>

            <button onClick={handleLogout}>logout</button>

        </>

    )

}

export default Profile

.