link: null
title: 珠峰架构师成长计划
description: src\index.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=58 sentences=249, words=1617

1.初始化项目

```
create-react-app zhufeng-keepalive
cd zhufeng-keepalive
npm install react-router-dom keepalive-react-component --save
npm start
```

2.跑通路由

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Link, Switch } from 'react-router-dom'
import Home from './components/Home';
import UserList from './components/UserList';
import UserAdd from './components/UserAdd';
const App = () => {
  return (
    <Router  >
      <ul>
        <li><Link to="/">首页Link>li>
        <li><Link to="/list">用户列表Link>li>
        <li><Link to="/add">添加用户Link>li>
      ul>
      <Switch>
        <Route path={'/'} component={Home} exact />
        <Route path={'/list'} component={UserList} />
        <Route path={'/add'} component={UserAdd} />
      Switch>
    Router>
  )
}
ReactDOM.render(<App/>, document.getElementById('root'));
```

src\components\Home.js

```
import React from 'react';
const Home = (props) => {
    return (
        <div>
            <button >重置UserAddbutton>
            <button >重置UserListbutton>
        div>
    )
}
export default Home;
```

src\components\UserAdd.js

```
import React from 'react';
const UserAdd = ()=>{
    let [number,setNumber]=React.useState(0);
    return (
        <div>
            用户名:<input/>
            <hr/>
            <button onClick={()=>setNumber(number=>number+1)}>{number}button>
        div>
    )
}
export default UserAdd;
```

src\components\UserList.js

```
import React from 'react';
import {Link} from 'react-router-dom'
const UserList = (props)=>{
    let users = new Array(100).fill(0);
    return (
        <ul style={{height:'200px',overflow:'scroll'}}>
            {
                users.map((item,index)=>(
                    <li key={index}><Link to={`/detail/${index}`}>{index}Link>li>
                ))
            }
        ul>
    )
}
export default UserList;
```

3.实现keep-alive

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Link, Switch } from 'react-router-dom'
import Home from './components/Home';
import UserList from './components/UserList';
import UserAdd from './components/UserAdd';
+import { KeepAliveProvider, withKeepAlive } from './keepalive-react-component';
+let KeepAliveHome = withKeepAlive(Home, { cacheId: 'Home'});
+let KeepAliveUserList = withKeepAlive(UserList, { cacheId: 'UserList'});
+let KeepAliveUserAdd = withKeepAlive(UserAdd, { cacheId: 'UserAdd' });
const App = () => {
  return (

+

          首页
          用户列表
          添加用户

+
+
+

+


  )
}
ReactDOM.render(, document.getElementById('root'));
```

src\keepalive-react-component\index.js

```
export {default as KeepAliveProvider} from './KeepAliveProvider';
export {default as withKeepAlive} from './withKeepAlive';
```

src\keepalive-react-component\cache-types.js

```
export const CREATE = 'CREATE';
export const CREATED = 'CREATED';
export const ACTIVE = 'ACTIVE';
```

src\keepalive-react-component\cacheReducer.js

```
import *  as cacheTypes from './cache-types';
function cacheReducer(cacheStates = {}, { type, payload }) {
    switch (type) {
        case cacheTypes.CREATE:
            return { ...cacheStates,
                [payload.cacheId]: {
                    cacheId:payload.cacheId,
                    element:payload.element,
                    status:cacheTypes.CREATE
                } };
        case cacheTypes.CREATED:
            return { ...cacheStates,
                [payload.cacheId]: {
                    ...cacheStates[payload.cacheId],
                    doms:payload.doms,
                    status:cacheTypes.CREATED
                } };
        case cacheTypes.ACTIVE:
            return { ...cacheStates,
                [payload.cacheId]: {
                    ...cacheStates[payload.cacheId],
                    status:cacheTypes.ACTIVE
                } };
        default:
            return cacheStates;
    }
}
export default cacheReducer;
```

src\keepalive-react-component\CacheContext.js

```
import React from 'react';
const CacheContext = React.createContext();
export default CacheContext;
```

src\keepalive-react-component\KeepAliveProvider.js

```
import React, { useReducer, useCallback } from "react";
import CacheContext from './CacheContext';
import cacheReducer from './cacheReducer';
import * as cacheTypes from './cache-types';
function KeepAliveProvider(props) {
    let [cacheStates, dispatch] = useReducer(cacheReducer, {});
    const mount = useCallback(({ cacheId, element }) => {
        if(!cacheStates[cacheId]){
            dispatch({ type: cacheTypes.CREATE, payload: { cacheId, element } });
        }
    }, [cacheStates]);
    return (
        <CacheContext.Provider value={{ mount, cacheStates, dispatch }}>
            {props.children}
            {Object.values(cacheStates).map(({ cacheId, element }) => (
                <div
                    id={`cache_${cacheId}`}
                    key={cacheId}
                    ref={(dom) => {
                        let cacheState = cacheStates[cacheId];
                        if (dom && (!cacheState.doms)) {
                            let doms = Array.from(dom.childNodes);
                            dispatch({ type: cacheTypes.CREATED, payload: { cacheId, doms } });
                        }
                    }}
                >{element}div>
            ))}
        CacheContext.Provider>
    );
}
export default KeepAliveProvider;
```

src\keepalive-react-component\withKeepAlive.js

```
import React, { useContext, useRef,useEffect } from "react";
import CacheContext from './CacheContext';
function withKeepAlive(OldComponent, { cacheId = window.location.pathname }) {
    return function (props) {
        const {mount, cacheStates,dispatch } = useContext(CacheContext);
        const ref = useRef(null);
        useEffect(() => {
            let cacheState = cacheStates[cacheId];
            if(cacheState&&cacheState.doms){
                 let doms = cacheState.doms;
                 doms.forEach(dom=>ref.current.appendChild(dom));
            }else{
                mount({ cacheId, element: <OldComponent {...props} dispatch={dispatch}/> })
            }
        }, [cacheStates, dispatch, mount, props]);
        return <div id={`keepalive_${cacheId}`} ref={ref} />;
    }
}
export default withKeepAlive;
```

4.保持滚动状态

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Link, Switch } from 'react-router-dom'
import Home from './components/Home';
import UserList from './components/UserList';
import UserAdd from './components/UserAdd';
import { KeepAliveProvider, withKeepAlive } from './keepalive-react-component';
let KeepAliveHome = withKeepAlive(Home, { cacheId: 'Home'});
+let KeepAliveUserList = withKeepAlive(UserList, { cacheId: 'UserList',scroll:true});
let KeepAliveUserAdd = withKeepAlive(UserAdd, { cacheId: 'UserAdd' });
const App = () => {
  return (

          首页
          用户列表
          添加用户


  )
}
ReactDOM.render(, document.getElementById('root'));
```

src\keepalive-react-component\cacheReducer.js

```
import *  as cacheTypes from './cache-types';
function cacheReducer(cacheStates = {}, { type, payload }) {
    switch (type) {
        case cacheTypes.CREATE:
            return { ...cacheStates,
                [payload.cacheId]: {
+                   scrolls:{},
                    cacheId:payload.cacheId,
                    element:payload.element,
                    status:cacheTypes.CREATE
                } };
        case cacheTypes.CREATED:
            return { ...cacheStates,
                [payload.cacheId]: {
                    ...cacheStates[payload.cacheId],
                    doms:payload.doms,
                    status:cacheTypes.CREATED
                } };
        case cacheTypes.ACTIVE:
            return { ...cacheStates,
                [payload.cacheId]: {
                    ...cacheStates[payload.cacheId],
                    status:cacheTypes.ACTIVE
                } };
        default:
            return cacheStates;
    }
}
export default cacheReducer;
```

src\keepalive-react-component\KeepAliveProvider.js

```
import React, { useReducer, useCallback } from "react";
import CacheContext from './CacheContext';
import cacheReducer from './cacheReducer';
import * as cacheTypes from './cache-types';
function KeepAliveProvider(props) {
    let [cacheStates, dispatch] = useReducer(cacheReducer, {});
    const mount = useCallback(({ cacheId, element }) => {
        if(!cacheStates[cacheId]){
            dispatch({ type: cacheTypes.CREATE, payload: { cacheId, element } });
        }
    }, [cacheStates]);
+   let handleScroll = useCallback((cacheId, {target}) => {
+       if(cacheStates[cacheId]){
+           let scrolls = cacheStates[cacheId].scrolls;
+           scrolls[target] = target.scrollTop;
+       }
+   }, [cacheStates]);
    return (
+
            {props.children}
            {Object.values(cacheStates).map(({ cacheId, element }) => (
                 {
                        let cacheState = cacheStates[cacheId];
                        if (dom && (!cacheState.doms)) {
                            let doms = Array.from(dom.childNodes);
                            dispatch({ type: cacheTypes.CREATED, payload: { cacheId, doms } });
                        }
                    }}
                >{element}
            ))}

    );
}
export default KeepAliveProvider;
```

src\keepalive-react-component\withKeepAlive.js

```
import React, { useContext, useRef,useEffect } from "react";
import CacheContext from './CacheContext';
+function withKeepAlive(OldComponent, { cacheId = window.location.pathname,scroll=false }) {
    return function (props) {
+       const {mount, cacheStates,dispatch,handleScroll } = useContext(CacheContext);
        const ref = useRef(null);
+       useEffect(()=>{
+           if(scroll){
+               ref.current.addEventListener('scroll', handleScroll.bind(null, cacheId),true);
+           }
+       },[handleScroll]);
        useEffect(() => {
            let cacheState = cacheStates[cacheId];
            if(cacheState&&cacheState.doms){
                let doms = cacheState.doms;
                doms.forEach(dom=>ref.current.appendChild(dom));
+               if(scroll){
+                  doms.forEach(dom=>{
+                      if (cacheState.scrolls[dom])
+                        dom.scrollTop = cacheState.scrolls[dom];
+                  });
+                }
            }else{
                mount({ cacheId, element:  })
            }
        }, [cacheStates, dispatch, mount, props]);
        return ;
    }
}
export default withKeepAlive;
```

5.销毁缓存

src\components\Home.js

```
import React from 'react';
const Home = (props) => {
    return (

+            props.dispatch({ type: 'DESTROY', payload: { cacheId: 'UserAdd' } })}>重置UserAdd
+            props.dispatch({ type: 'DESTROY', payload: { cacheId: 'UserList' } })}>重置UserList

    )
}
export default Home;
```

src\keepalive-react-component\cache-types.js

```
export const CREATE = 'CREATE';        //创建
export const CREATED = 'CREATED';      //创建成功
export const ACTIVE = 'ACTIVE';        //激活
+export const DESTROY = 'DESTROY';     //销毁
```

src\keepalive-react-component\cacheReducer.js

```
import *  as cacheTypes from './cache-types';
function cacheReducer(cacheStates = {}, { type, payload }) {
    switch (type) {
        case cacheTypes.CREATE:
            return { ...cacheStates,
                [payload.cacheId]: {
                    scrolls:{},
                    cacheId:payload.cacheId,
                    element:payload.element,
                    status:cacheTypes.CREATE
                } };
        case cacheTypes.CREATED:
            return { ...cacheStates,
                [payload.cacheId]: {
                    ...cacheStates[payload.cacheId],
                    doms:payload.doms,
                    status:cacheTypes.CREATED
                } };
        case cacheTypes.ACTIVE:
            return { ...cacheStates,
                [payload.cacheId]: {
                    ...cacheStates[payload.cacheId],
                    status:cacheTypes.ACTIVE
                } };
+       case cacheTypes.DESTROY:
+           return { ...cacheStates,
+               [payload.cacheId]:{
+                   ...cacheStates[payload.cacheId],
+                   status:cacheTypes.DESTROY
+               }};
        default:
            return cacheStates;
    }
}
export default cacheReducer;
```

src\keepalive-react-component\KeepAliveProvider.js

```
import React, { useReducer, useCallback } from "react";
import CacheContext from './CacheContext';
import cacheReducer from './cacheReducer';
import * as cacheTypes from './cache-types';
function KeepAliveProvider(props) {
    let [cacheStates, dispatch] = useReducer(cacheReducer, {});
    const mount = useCallback(({ cacheId, element }) => {
+      if(cacheStates[cacheId]){
+          let cacheState = cacheStates[cacheId];
+          if(cacheState.status === cacheTypes.DESTROY){
+              let doms = cacheState.doms;
+              doms.forEach(dom=>dom.parentNode.removeChild(dom));
+              dispatch({ type: cacheTypes.CREATE, payload: { cacheId, element } });
+          }
+      }else{
            dispatch({ type: cacheTypes.CREATE, payload: { cacheId, element } });
+      }
    }, [cacheStates]);
    let handleScroll = useCallback((cacheId, {target}) => {
        if(cacheStates[cacheId]){
            let scrolls = cacheStates[cacheId].scrolls;
            scrolls[target] = target.scrollTop;
        }
    }, [cacheStates]);
    return (

            {props.children}
+           {Object.values(cacheStates).filter(cacheState=>cacheState.status!==cacheTypes.DESTROY).map(({ cacheId, element }) => (
                {
                        let cacheState = cacheStates[cacheId];
+                       if (dom && (!cacheState.doms || cacheState.status === cacheTypes.DESTROY) ) {
                            let doms = Array.from(dom.childNodes);
                            dispatch({ type: cacheTypes.CREATED, payload: { cacheId, doms } });
                        }
                    }}
                >{element}
            ))}

    );
}
export default KeepAliveProvider;
```

src\keepalive-react-component\withKeepAlive.js

```
import React, { useContext, useRef,useEffect } from "react";
import CacheContext from './CacheContext';
+import * as cacheTypes from './cache-types';
function withKeepAlive(OldComponent, { cacheId = window.location.pathname,scroll=false }) {
    return function (props) {
        const {mount, cacheStates,dispatch,handleScroll } = useContext(CacheContext);
        const ref = useRef(null);
        useEffect(()=>{
            if(scroll){
                ref.current.addEventListener('scroll', handleScroll.bind(null, cacheId),true);
            }
        },[handleScroll]);
        useEffect(() => {
            let cacheState = cacheStates[cacheId];
+           if(cacheState&&cacheState.doms && cacheState.status !== cacheTypes.DESTROY){
                let doms = cacheState.doms;
                doms.forEach(dom=>ref.current.appendChild(dom));
                if(scroll){
                   doms.forEach(dom=>{
                       if (cacheState.scrolls[dom])
                         dom.scrollTop = cacheState.scrolls[dom];
                   });
                 }
            }else{
                mount({ cacheId, element:  })
            }
        }, [cacheStates, dispatch, mount, props]);
        return ;
    }
}
export default withKeepAlive;
```