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=114 sentences=202, words=1422

1.React RouterV6新特性

- 官网 (https://reactrouter.com/docs/en/v6)
- Switch重命名为Routes
- Route的新特性
- 嵌套路由变得更简单
- 用useNavigate代替useHistory
- 新钩子useRoutes代替react-router-config

1.1 Switch重命名为Routes

- component/render被element替代
- Routes和Route基于当前位置在 React Router中渲染某些内容的主要方法
- 您可以把Route考虑为一种类似于if语句的路由，如果其路径与当前URL匹配，则呈现其元素
- Route的caseSensitive属性属性确定是否应以区分大小写的方式进行匹配(默认为false)

1.1.1 src\index.js

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Routes, Link } from 'react-router-dom';
import Home from './components/Home';
import User from './components/User';
import Profile from './components/Profile';
ReactDOM.render(
  <Router>
    <ul>
      <li><Link to="/">HomeLink>li>
      <li><Link to="/user">UserLink>li>
      <li><Link to="/profile">ProfileLink>li>
    ul>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/user" element={<User />} />
      <Route path="/profile" element={<Profile />} />
    Routes>
  Router>,
  document.getElementById('root')
);
```

1.2 嵌套路由变得更简单

- Route children 已更改为接受子路由
- 比Route exact和Route strict更简单的匹配规则
- Route path 路径层次更清晰

1.2.1 src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Routes, Link } from 'react-router-dom';
import Home from './components/Home';
import User from './components/User';
import Profile from './components/Profile';
ReactDOM.render(

      Home
      User
      Profile

      } />
+     } />
      } />

  ,
  document.getElementById('root')
);
```

1.2.2 User.js

src\components\User.js

```
import React from 'react';
+import { Route, Routes, Link } from 'react-router-dom';
+import UserAdd from './UserAdd';
+import UserDetail from './UserDetail';
+import UserList from './UserList';
function User() {
    return (
+
+
+                添加用户
+                用户列表
+
+
+                } />
+                } />
+                } />
+
+
+   )
}
export default User;
```

1.2.3 UserAdd.js

src\components\UserAdd.js

```
import React from 'react';
function UserAdd() {
    return (
        <div>UserAdddiv>
    )
}
export default UserAdd;
```

1.2.4 UserDetail.js

src\components\UserDetail.js

```
import React from 'react';
function UserDetail() {
    return (
        <div>UserDetaildiv>
    )
}
export default UserDetail;
```

1.2.5 UserList.js

src\components\UserList.js

```
import React from 'react';
function UserList() {
    return (
        <div>UserListdiv>
    )
}
export default UserList;
```

1.3 Outlet

- 应在父Route元素中使用Outlet(插座)来渲染其子Route元素
- 这允许在渲染子Route时显示嵌套UI
- 如果父路由完全匹配，则会呈现子路由，如果没有路由，则不会呈现任何内容
- *1.3.1 src\index.js #*

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router, Route, Routes, Link } from 'react-router-dom';
import Home from './components/Home';
import User from './components/User';
import Profile from './components/Profile';
import UserAdd from './components/UserAdd';
import UserDetail from './components/UserDetail';
import UserList from './components/UserList';
ReactDOM.render(

      Home
      User
      Profile

      } />
+     } >
+       } />
+       } />
+       } />
+
      } />

  ,
  document.getElementById('root')
);
```

** 1.3.2 User.js #**

src\components\User.js

```
import React from 'react';
+import { Link, Outlet } from 'react-router-dom';
function User() {
    return (

                添加用户
                用户列表

+


    )
}
export default User;
```

1.4 用useNavigate代替useHistory

- useNavigate钩子返回一个函数，该函数允许您以编程方式进行导航

** 1.4.1 Home.js #**

src\components\Home.js

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
function Home() {
+   let navigate = useNavigate();
+   function navigateToUser() {
+       navigate('/user');
+   };
+   return (
+
+           Home
+           跳转到/user
+
+   )
}
export default Home;
```

1.5 5. 新钩子useRoutes代替react-router-config

- useRoutes钩子在功能上等同于Routes，但它使用JavaScript对象而不是Route元素来定义路由
- 这些对象与普通Route元素具有相同的属性，但它们不需要JSX

** 1.5.1 src\index.js #**

```
import React from 'react';
import ReactDOM from 'react-dom';
+import { BrowserRouter as Router, Route, Routes, Link, useRoutes } from 'react-router-dom';
+import routes from './routes';
+function App() {
+  return useRoutes(routes);
+}
ReactDOM.render(

      Home
      User
      Profile


+
  ,
  document.getElementById('root')
);
```

** 1.5.2 src\routes.js #**

src\routes.js

```
import Home from './components/Home';
import User from './components/User';
import Profile from './components/Profile';
import UserAdd from './components/UserAdd';
import UserDetail from './components/UserDetail';
import UserList from './components/UserList';
import NotFound from './components/NotFound';
const routes = [
    { path: '/', element: <Home /> },
    { path: '/profile', element: <Profile /> },
    {
        path: 'user',
        element: <User />,
        children: [
            { path: 'add', element: <UserAdd /> },
            { path: 'list', element: <UserList /> },
            { path: 'detail/:id', element: <UserDetail /> }
        ]
    },

    { path: '*', element: <NotFound /> }
];
export default routes;
```

1.6 Redirect 标签删除

- 解决方案：新版的路由需要引入标签
- Navigate元素在渲染时更改当前位置

```
import { HashRouter as Router, Route, Routes, Navigate } from 'react-router-dom'

        }/>
        }/>
        } />
```

2. React路由原理

- 不同的路径渲染不同的组件

- 有两种实现方式
    - HashRouter:利用hash实现路由切换
    - BrowserRouter:实现h5 Api实现路由的切换

2.1 HashRouter

- 利用hash实现路由切换

public\index.html

```
Document

    #root{
        border:1px solid red;
    }

    /a
    /b

    window.addEventListener('hashchange',()=>{
        console.log(window.location.hash);
        let pathname = window.location.hash.slice(1);//把最前面的那个#删除
        root.innerHTML = pathname;
    });
```

2.2 BrowserRouter

- 利用h5 Api实现路由的切换

** 2.2.1 history #**

- HTML5规范给我们提供了一个history (https://developer.mozilla.org/zh-CN/docs/Web/API/Window/history)接口
- HTML5 History API包括2个方法： `history.pushState()`和 `history.replaceState()`，和1个事件 `window.onpopstate`

2.2.1.1 pushState

- history.pushState(stateObject, title, url)，包括三个参数
    - 第一个参数用于存储该url对应的状态对象，该对象可在onpopstate事件中获取，也可在history对象中获取
    - 第二个参数是标题，目前浏览器并未实现
    - 第三个参数则是设定的url
- pushState函数向浏览器的历史堆栈压入一个url为设定值的记录，并改变历史堆栈的当前指针至栈顶

2.2.1.2 onpopstate

- 该事件是window的属性
- 该事件会在调用浏览器的前进、后退以及执行 `history.forward`、 `history.back`、和 `history.go`触发，因为这些操作有一个共性，即修改了历史堆栈的当前指针
- 在不改变document的前提下，一旦当前指针改变则会触发 `onpopstate`事件

2.2.1.4 案例

- 浏览器针对每个页面维护一个 `History`栈,执行 `pushState`函数可压入设定的 `url`至栈顶,同时修改当前指针
- 当执行 `back`和 `forward`操作时，history栈大小并不会改变（history.length不变），仅仅移动当前指针的位置
- 若当前指针在history栈的中间位置(非栈顶)，此时执行pushState会在指针当前的位置添加此条目,并成为新的栈顶

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>browsertitle>
head>

<body>
    <div id="root">div>
    <ul>
        <li><a onclick="pushA()">/aa>li>
        <li><a onclick="pushB()">/ba>li>
        <li><a onclick="pushC()">/ca>li>
        <li><a onclick="forward()">forwarda>li>
        <li><a onclick="back()">backa>li>
        <li><a onclick="go(-1)">go-1=backa>li>
        <li><a onclick="go(1)">go+1=forwarda>li>
    ul>
    <script>
        let history = window.history;
        (function (history) {
            let oldPushState = history.pushState;
            history.pushState = function (state, title, pathname) {
                let result = oldPushState.apply(history, arguments);
                if (typeof window.onpushstate === 'function') {
                    window.onpushstate(new CustomEvent('pushstate', { detail: { pathname, state } }));
                }
            }
        })(history);
        window.onpushstate = (event) => {
            console.log(event);
            root.innerHTML = window.location.pathname;
        }
        window.onpopstate = (event) => {
            console.log(event);
            root.innerHTML = window.location.pathname;
        }
        function pushA() {
            history.pushState({ name: 'a' }, null, '/a');
        }
        function pushB() {
            history.pushState({ name: 'b' }, null, '/b');
        }
        function pushC() {
            history.pushState({ name: 'c' }, null, '/c');
        }
        function forward() {
            history.forward();
        }
        function back() {
            history.back();
        }
        function go(step) {
            history.go(step);
        }
    script>
body>
html>
```

3.使用基本路由

- https://create-react-app.dev (https://create-react-app.dev)
- https://reactrouter.com/docs/en/v6 (https://reactrouter.com/docs/en/v6)
- https://github.com/remix-run/react-router (https://github.com/remix-run/react-router)

3.1 安装

```
npm i react-router-dom --save
```

3.2 src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { HashRouter, BrowserRouter, Routes, Route } from 'react-router-dom';
import Home from './components/Home';
import User from './components/User';
import Profile from './components/Profile';
ReactDOM.render(
  <HashRouter>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/user" element={<User />} />
      <Route path="/profile" element={<Profile />} />
    Routes>
  HashRouter>
  , document.getElementById('root'));
```

3.3 Home.js

src\components\Home.js

```
import React from 'react';
function Home(props) {
    console.log(props);
    return (
        <div>Homediv>
    )
}
export default Home;
```

3.4 User.js

src\components\User.js

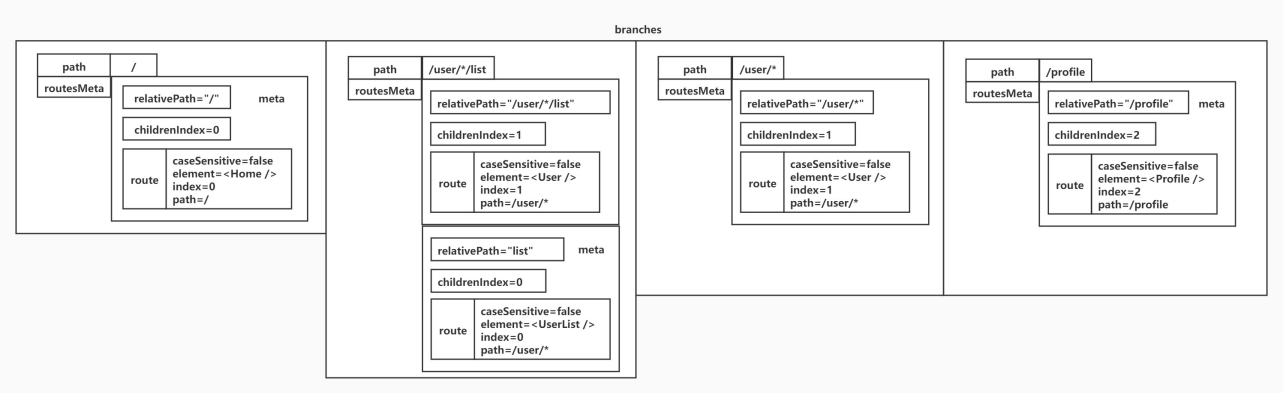
```
import React from 'react';
function User() {
    return (
        <div>Userdiv>
    )
}
export default User;
```

3.5 Profile.js

src\components\Profile.js

```
import React from 'react';
function Profile() {
    return (
        <div>Profilediv>
    )
}
export default Profile;
```

4.实现基本路由



4.1 react-router-dom\index.js

src\react-router-dom\index.js

```
import React from 'react'
import { Router } from '../react-router';
import { createHashHistory, createBrowserHistory } from "history";
export * from '../react-router';

export function HashRouter({ children }) {
    let historyRef = React.useRef();
    if (historyRef.current == null) {
        historyRef.current = createHashHistory();
    }
    let history = historyRef.current;
    let [state, setState] = React.useState({
        action: history.action,
        location: history.location
    });
    React.useLayoutEffect(() => history.listen(setState), [history]);
    return (
        <Router
            children={children}
            location={state.location}
            navigationType={state.action}
            navigator={history}
        />
    );
}

export function BrowserRouter({ children }) {
    let historyRef = React.useRef();
    if (historyRef.current == null) {
        historyRef.current = createBrowserHistory();
    }
    let history = historyRef.current;
    let [state, setState] = React.useState({
        action: history.action,
        location: history.location
    });
    React.useLayoutEffect(() => history.listen(setState), [history]);
    return (
        <Router
            children={children}
            location={state.location}
            navigationType={state.action}
            navigator={history}
        />
    );
}
```

4.2 src\react-router\index.js

src\react-router\index.js

```
import React from 'react';

const NavigationContext = React.createContext({});

const LocationContext = React.createContext({});

const RouteContext = React.createContext({});

export {
    NavigationContext,
    LocationContext,
    RouteContext
};
export function Router({ children, location, navigator }) {
    let navigationContext = React.useMemo(
        () => ({ navigator }),
        [navigator]
    );
    return (
        <NavigationContext.Provider value={navigationContext}>
            <LocationContext.Provider
                children={children}
                value={{ location }}
            />
        NavigationContext.Provider>
    );
}
export function Routes({ children }) {
    return useRoutes(createRoutesFromChildren(children));
}
export function useLocation() {
    return React.useContext(LocationContext).location;
}
export function useRoutes(routes) {
    let location = useLocation();
    let pathname = location.pathname || "/";
    for (let i = 0; i < routes.length; i++) {
        let { path, element } = routes[i];
        let match = matchPath(path, pathname);
        if (match) {
            return element;
        }
    }
    return null;
}

export function createRoutesFromChildren(children) {
    let routes = [];
    React.Children.forEach(children, element => {
        let route = {
            path: element.props.path,
            element: element.props.element
        };
        routes.push(route);
    });
    return routes;
}

export function Route(props) {

}
function compilePath(path) {
    let source = "^" + path;
    source += "{{content}}quot;;
    let matcher = new RegExp(source);
    return matcher;
}
export function matchPath(path, pathname) {
    let matcher = compilePath(path);
    let match = pathname.match(matcher);
    if (!match) return null;
    return match;
}
```