link: null
title: 珠峰架构师成长计划
description: webpack.config.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=160 sentences=710, words=4111

1. 初始化项目

- 项目预览 (http://img.zhufengpeixun.cn/tree4.html)

1.1 创建项目

```
mkdir zhufeng_prepare_tree
cd zhufeng_prepare_tree
cnpm init -y
touch .gitignore
```

1.2 安装依赖

- @types (https://github.com/DefinitelyTyped/DefinitelyTyped)开头的包都是typeScript的声明文件，可以进入 `node_modules/@types/XX/index.d.ts`进行查看

```
cnpm i react @types/react react-dom @types/react-dom -S
cnpm i webpack webpack-cli webpack-dev-server  -D
cnpm i typescript ts-loader source-map-loader style-loader css-loader  less-loader  less file-loader url-loader html-webpack-plugin -D
cnpm i jest @types/jest ts-jest jest-junit enzyme @types/enzyme
enzyme-adapter-react-16 @types/enzyme-adapter-react-16 -D
cnpm i axios express qs @types/qs -D
```

模块名 使用方式 react React is a JavaScript library for creating user interfaces. react-dom This package serves as the entry point to the DOM and server renderers for React. It is intended to be paired with the generic React package, which is shipped as react to npm. webpack webpack is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset. webpack-cli The official CLI of webpack webpack-dev-server Use webpack with a development server that provides live reloading. This should be used for development only. typescript TypeScript is a language for application-scale JavaScript. ts-loader This is the TypeScript loader for webpack. source-map-loader Extracts source maps from existing source files (from their sourceMappingURL). style-loader Inject CSS into the DOM. css-loader The css-loader interprets @import and url() like import/require() and will resolve them. less-loader A Less loader for webpack. Compiles Less to CSS. less This is the JavaScript, official, stable version of Less. file-loader The file-loader resolves import/require() on a file into a url and emits the file into the output directory. url-loader A loader for webpack which transforms files into base64 URIs. html-webpack-plugin Plugin that simplifies creation of HTML files to serve your bundles jest
jest (https://jestjs.io/)

is a delightful JavaScript Testing Framework with a focus on simplicity. jest-junit A Jest reporter that creates compatible junit xml files ts-jest ts-jest is a TypeScript preprocessor with source map support for Jest that lets you use Jest to test projects written in TypeScript. enzyme JavaScript Testing utilities for React enzyme-adapter-react-16 Enzyme is a JavaScript Testing utility for React that makes it easier to test your React Components' output. You can also manipulate, traverse, and in some ways simulate runtime given the output.

1.3 支持typescript

- 首先需要生成一个 `tsconfig.json`文件来告诉 `ts-loader`如何编译代码TypeScript代码

```
tsc --init
```

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "jsx": "react",
    "outDir": "./dist",
    "rootDir": "./src",
    "noImplicitAny":true,
    "esModuleInterop": true
  },
  "include": [
    "./src/**/*",
    "./typings/**/*"
  ]
}
```

参数 含义 target 转换成es5 module 代码规范 jsx react模式会生成React.createElement，在使用前不需要再进行转换操作了，输出文件的扩展名为.js outDir 指定输出目录 rootDir 指定根目录 sourceMap 把 ts 文件编译成 js 文件的时候，同时生成对应的sourceMap文件 noImplicitAny 如果为true的话，TypeScript 编译器无法推断出类型时，它仍然会生成 JS文件，但是它也会报告一个错误 esModuleInterop 是否转译common.js模块 include 需要编译的目录

1.4 webpack.config.js

webpack.config.js

```
const webpack = require('webpack');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const path = require('path');
module.exports = {
    mode: 'development',
    entry: "./src/index.tsx",
    output: {
        path: path.join(__dirname, 'dist')
    },
    devtool: "source-map",
    devServer: {
        hot: true,
        contentBase: path.join(__dirname, 'dist'),
        historyApiFallback: {
            index: './index.html'
        }
    },
    resolve: {
        extensions: [".ts", ".tsx", ".js", ".json"]
    },

    module: {
        rules: [{
            test: /\.tsx?$/,
            loader: "ts-loader"
        },
        {
            enforce: "pre",
            test: /\.tsx$/,
            loader: "source-map-loader"
        },
        {
            test: /\.less$/,
            use: ['style-loader', 'css-loader', 'less-loader']
        },
        {
            test: /\.(jpg|png|gif|svg)$/,
            loader: "url-loader"
        }
        ]
    },

    plugins: [
        new HtmlWebpackPlugin({
            template: './src/index.html'
        }),
        new webpack.HotModuleReplacementPlugin()
    ],
};
```

1.5 src\index.tsx

src\index.tsx

```
console.log('hello');
```

1.6 src\index.html

src\index.html

```
<body>
    <div id="root">div>
body>
```

1.7 package.json

package.json

```
  "scripts": {
    "build": "webpack",
    "dev": "webpack-dev-server",
    "test": "jest"
  }
```

2. 单元测试

2.1 jest.config.js

jest.config.js

```
module.exports = {
    verbose: true,
    clearMocks: true,
    collectCoverage: true,
    reporters: ["default", "jest-junit"],
    moduleFileExtensions: ['js', 'jsx', 'ts', 'tsx'],
    moduleDirectories: ['node_modules'],
    transform: {
        '^.+\\.tsx?{{content}}#x27;: 'ts-jest',
    },
    moduleNameMapper: {
        "\\.(jpg|jpeg|png|gif||svg|ttf|woff|woff2){{content}}quot;:
            "/test/__mocks__/file-mock.js",
        "\\.(css|less|sass|scss){{content}}quot;:
            "/test/__mocks__/object-mock.js"
    },
    testRegex: '(/__tests__/.*|(\\.|/)(test|spec))\\.(jsx|tsx){{content}}#x27;,
    moduleFileExtensions: ['ts', 'tsx', 'js', 'jsx', 'json', 'node'],
    setupFilesAfterEnv: ['/test/setupTests.tsx']
}
```

2.2 src\utils.tsx

src\utils.tsx

```ts
export function sum(a: number, b: number): number {
    return a + b;
}
```

2.3 utils.spec.tsx

src__tests__\utils.spec.tsx

```ts
import { sum } from '../utils';
describe('sum', () => {
    test('1+1', () => {
        expect(sum(1, 1)).toBe(2);
    });
    test('2+2', () => {
        expect(sum(2, 2)).toBe(4);
    });
});
```

2.4 mocks\file-mock.js

test__mocks__\file-mock.js

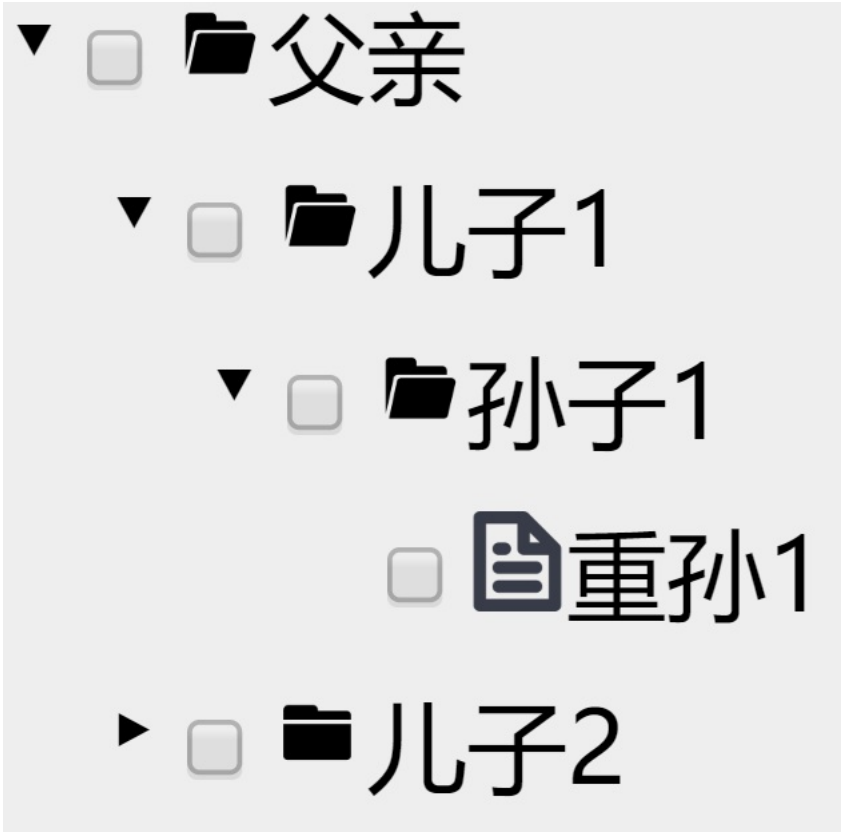
```ts
module.exports = 'file-stub';
```

2.5 object-mock.js

test__mocks__\object-mock.js

```ts
module.exports = {}
```

3. 创建树型菜单



3.1 src\index.tsx

src\index.tsx

```ts
import React from 'react';
import ReactDOM from 'react-dom';
import Tree from './components/tree';
import data from './data';

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

3.2 src\data.tsx

src\data.tsx

```
import { TreeData } from './typings';
const data: TreeData = {
    name: '父亲',
    key: '1',
    type: 'folder',
    collapsed: false,
    children: [
        {
            name: '儿子1',
            key: '1-1',
            type: 'folder',
            collapsed: false,
            children: [
                {
                    name: '孙子1',
                    key: '1-1-1',
                    type: 'folder',
                    collapsed: false,
                    children: [
                        {
                            name: '重孙1',
                            key: '1-1-1-1',
                            type: 'file',
                            collapsed: false,
                            children: []
                        }
                    ]
                }
            ]
        },
        {
            name: '儿子2',
            key: '1-2',
            type: 'folder',
            collapsed: true
        }
    ]
}
export default data;
```

3.3 typings.tsx

src\typings.tsx

```
export interface TreeData {
    name: string;
    key: string;
    type: string;
    collapsed?: boolean;
    children?: Array;
    parent?: TreeData;
}
```

3.4 components\tree.tsx

src\components\tree.tsx

```
import React from 'react';
import './index.less';
import { TreeData } from '../typings';
interface Props {
    data: TreeData
}
class Tree extends React.Component {
    render() {
        return (

                Tree

        )
    }
}
export default Tree;
```

3.5 index.less

src\components\index.less

```
.tree{
    position: fixed;
    left:0;
    top:0;
    bottom:0;
    width:80%;
    overflow-x: hidden;
    overflow-y: auto;
    background-color: #EEE;
}
```

4. 渲染树型结构

4.1 src\components\tree.tsx

src\components\tree.tsx

```
import React from 'react';
import './index.less';
import { TreeData } from '../typings';
+import TreeNode from './tree-node';
interface Props {
    data: TreeData
}
+interface KeyToNodeMap {
+    [key: string]: TreeData
+}
+interface State {
+    data: TreeData
+}
class Tree extends React.Component {
+    data: TreeData;
+    keyToNodeMap: KeyToNodeMap;
+    constructor(props: Props) {
+        super(props);
+        this.data = props.data;
+        this.state={ data: this.props.data };
+        this.buildKeyMap();
+    }
+    buildKeyMap = () => {
+        let data = this.data;
+        this.keyToNodeMap = {};
+        this.keyToNodeMap[data.key] = data;
+        if (data.children && data.children.length > 0) {
+            this.walk(data.children, data);
+        }
+    }
+    walk = (children: Array, parent: TreeData): void => {
+        children.map((item: TreeData) => {
+            item.parent = parent;
+            this.keyToNodeMap[item.key] = item;
+            if (item.children && item.children.length > 0) {
+                this.walk(item.children, item);
+            }
+        });
+    }
    render() {
        return (

+
+
+                        data={this.props.data}
+                    />
+


        )
    }
}
export default Tree;
```

4.2 index.less

src\components\index.less

```
.tree{
    position: fixed;
    left:0;
    top:0;
    bottom:0;
    width:80%;
    overflow-x: hidden;
    overflow-y: auto;
    background-color: #EEE;
+    .tree-nodes{
+        position: relative;
+        overflow:hidden;
+        .tree-node{
+            .inner{
+                color:#000;
+                font-size:20px;
+                position: relative;
+                cursor:pointer;
+                padding-left:10px;
+                .content{
+                    display: inline-block;
+                    width:100%;
+                    padding:4px 5px;
+                }
+            }
+            .children{
+                padding-left: 20px;
+            }
+        }
+    }
}
```

4.3 components\tree-node.tsx

src\components\tree-node.tsx

```
import React from 'react';
import { TreeData } from '../typings';
interface Props {
    data: TreeData
}
class TreeNode extends React.Component<Props> {
    constructor(props: Props) {
        super(props);
    }
    render() {
        let { data: { name, children } } = this.props;
        return (
            <div className="tree-node">
                <div className="inner">
                    <span className="content">{name}span>
                div>
                {
                    (children && children.length > 0) && (
                        <div className="children">
                            {
                                children.map((item: TreeData) => (
                                    <TreeNode key={item.key} data={item} />
                                ))
                            }
                        div>
                    )
                }
            div>
        )
    }
}
export default TreeNode;
```

5. 打开关闭功能

- icons.zip (http://img.zhufengpeixun.cn/icons.zip)

5.1 components\tree.tsx

src\components\tree.tsx

```
import React from 'react';
import './index.less';
import { TreeData } from '../typings';
import TreeNode from './tree-node';
interface Props {
    data: TreeData
}
interface KeyToNodeMap {
    [key: string]: TreeData
}
interface State {
    data: TreeData
}
class Tree extends React.Component {
    data: TreeData;
    keyToNodeMap: KeyToNodeMap;
    constructor(props: Props) {
        super(props);
        this.data = props.data;
        this.state={ data: this.props.data };
        this.buildKeyMap();
    }
    buildKeyMap = () => {
        let data = this.data;
        this.keyToNodeMap = {};
        this.keyToNodeMap[data.key] = data;
        if (data.children && data.children.length > 0) {
            this.walk(data.children, data);
        }
    }
    walk = (children: Array, parent: TreeData): void => {
        children.map((item: TreeData) => {
            item.parent = parent;
            this.keyToNodeMap[item.key] = item;
            if (item.children && item.children.length > 0) {
                this.walk(item.children, item);
            }
        });
    }
+    onCollapse = (key: string) => {
+        let data = this.keyToNodeMap[key];
+        if (data) {
+            data.collapsed = !data.collapsed;
+            data.children = data.children || [];
+            this.setState({ data: this.state.data });
+        }
+    }
    render() {
        return (

                    +                       onCollapse={this.onCollapse}
                    />

        )
    }
}
export default Tree;
```

5.2 tree-node.tsx

src\components\tree-node.tsx

```
import React from 'react';
import { TreeData } from '../typings';
+import file from '../assets/file.png';
+import closedFolder from '../assets/closed-folder.png';
+import openedFolder from '../assets/opened-folder.png';
interface Props {
    data: TreeData,
+    onCollapse: any
}
class TreeNode extends React.Component {
    constructor(props: Props) {
        super(props);
    }
    render() {
+        let { data: { name, children, collapsed = false, key } } = this.props;
+        let caret, icon;
+        if (children) {
+            if (children.length > 0) {
+                caret = (
+
+                        onClick={() => this.props.onCollapse(key)}
+                    />
+                )
+                icon = collapsed ? closedFolder : openedFolder;
+            } else {
+                caret = null;
+                icon = file;
+            }
+        } else {
+            caret = (
+
+                    onClick={() => this.props.onCollapse(key)}
+                />
+            )
+            icon = closedFolder;
+        }
        return (

+                    {caret}
+
+
+                        {name}
+


                {
+                    (children && children.length > 0 && !collapsed) && (

                            {
                                children.map((item: TreeData) => (
                                    +                                        onCollapse={this.props.onCollapse}
                                        key={item.key}
                                        data={item} />
                                ))
                            }

                    )
                }

        )
    }
}
export default TreeNode;
```

5.3 components\index.less

src\components\index.less

```
.tree{
    position: fixed;
    left:0;
    top:0;
    bottom:0;
    width:80%;
    overflow-x: hidden;
    overflow-y: auto;
    background-color: #EEE;
    .tree-nodes{
        position: relative;
        overflow:hidden;
        .tree-node{
            .inner{
                color:#000;
                font-size:20px;
                position: relative;
                cursor:pointer;
                padding-left:10px;
+                .collapse {
+                    position: absolute;
+                    left: 0;
+                    cursor: pointer;
+                }
+                .caret-right:before {
+                    content: '\25B8';
+                }
+                .caret-down:before {
+                    content: '\25BE';
+                }
                .content{
                    display: inline-block;
                    width:100%;
                    padding:4px 5px;
                }
            }
            .children{
                padding-left: 20px;
            }
        }

    }
}
```

5.4 images.d.ts

typings\images.d.ts

```
declare module '*.svg';
declare module '*.png';
declare module '*.jpg';
declare module '*.jpeg';
declare module '*.gif';
declare module '*.bmp';
declare module '*.tiff';
```

6. 全选和全消功能

6.1 src\typings.tsx

src\typings.tsx

```
export interface TreeData {
    name: string;
    key: string;
    type: string;
    collapsed: boolean;
    children?: Array;
    parent?: TreeData;
    checked?: boolean;
}
```

6.2 components\tree.tsx

src\components\tree.tsx

```
import React from 'react';
import './index.less';
import { TreeData } from '../typings';
import TreeNode from './tree-node';
interface Props {
    data: TreeData
}
interface KeyToNodeMap {
    [key: string]: TreeData
}
interface State {
    data: TreeData
}
class Tree extends React.Component {
    data: TreeData;
    keyToNodeMap: KeyToNodeMap;
    constructor(props: Props) {
        super(props);
        this.data = props.data;
        this.state={ data: this.props.data };
        this.buildKeyMap();
    }
    buildKeyMap = () => {
        let data = this.data;
        this.keyToNodeMap = {};
        this.keyToNodeMap[data.key] = data;
        if (data.children && data.children.length > 0) {
            this.walk(data.children, data);
        }
    }
    walk = (children: Array, parent: TreeData): void => {
        children.map((item: TreeData) => {
            item.parent = parent;
            this.keyToNodeMap[item.key] = item;
            if (item.children && item.children.length > 0) {
                this.walk(item.children, item);
            }
        });
    }
    onCollapse = (key: string) => {
        let data = this.keyToNodeMap[key];
        if (data) {
            data.collapsed = !data.collapsed;
            data.children = data.children || [];
            this.setState({ data: this.state.data });
        }
    }
+    onCheck = (key: string) => {
+        let data: TreeData = this.keyToNodeMap[key];
+        if (data) {
+            data.checked = !data.checked;
+            if (data.checked) {
+                this.checkChildren(data.children, true);
+                this.checkParentCheckAll(data.parent);
+            } else {
+                this.checkChildren(data.children, false);
+                this.checkParent(data.parent, false);
+            }
+            this.setState({ data: this.state.data });
+        }
+    }
+    checkParentCheckAll = (parent: TreeData) => {
+        while (parent) {
+            parent.checked = parent.children.every(item => item.checked);
+            parent = parent.parent;
+        }
+    }
+    checkParent = (parent: TreeData, checked: boolean) => {
+        while (parent) {
+            parent.checked = checked;
+            parent = parent.parent;
+        }
+    }
+    checkChildren = (children: Array = [], checked: boolean) => {
+        children.forEach((item: TreeData) => {
+            item.checked = checked;
+            this.checkChildren(item.children, checked);
+        });
+    }
    render() {
        return (

                    +                       onCheck={this.onCheck}
                    />

        )
    }
}
export default Tree;
```

6.3 tree-node.tsx

src\components\tree-node.tsx

```
import React from 'react';
import { TreeData } from '../typings';
import file from '../assets/file.png';
import closedFolder from '../assets/closed-folder.png';
import openedFolder from '../assets/opened-folder.png';
interface Props {
    data: TreeData,
    onCollapse: any,
+    onCheck: any
}
class TreeNode extends React.Component {
    constructor(props: Props) {
        super(props);
    }
    render() {
+        let { data: { name, children, collapsed = false, key, checked = false } } = this.props;
        let caret, icon;
        if (children) {
            if (children.length > 0) {
                caret = (
                     this.props.onCollapse(key)}
                    />
                )
                icon = collapsed ? closedFolder : openedFolder;
            } else {
                caret = null;
                icon = file;
            }
        } else {
            caret = (
                 this.props.onCollapse(key)}
                />
            )
            icon = closedFolder;
        }
        return (

                    {caret}

+                         this.props.onCheck(key)} />

                        {name}

                {
                    (children && children.length > 0 && !collapsed) && (

                            {
                                children.map((item: TreeData) => (
                                    +                                        onCheck={this.props.onCheck}
                                        key={item.key}
                                        data={item} />
                                ))
                            }

                    )
                }

        )
    }
}
export default TreeNode;
```

7. 动态加载数据

7.1 typings.tsx

src\typings.tsx

```
export interface TreeData {
    name: string;
    key: string;
    type: string;
    collapsed: boolean;
    children?: Array;
    parent?: TreeData;
    checked?: boolean;
+   loading?: boolean;
}
```

7.2 tree.tsx

src\components\tree.tsx

```
import React from 'react';
import './index.less';
import { TreeData } from '../typings';
import TreeNode from './tree-node';
+import { getChildren } from '../api';
interface Props {
    data: TreeData
}
interface KeyToNodeMap {
    [key: string]: TreeData
}
interface State {
    data: TreeData
}
class Tree extends React.Component {
    data: TreeData;
    keyToNodeMap: KeyToNodeMap;
    constructor(props: Props) {
        super(props);
        this.data = props.data;
        this.state={ data: this.props.data };
        this.buildKeyMap();
    }
    buildKeyMap = () => {
        let data = this.data;
        this.keyToNodeMap = {};
        this.keyToNodeMap[data.key] = data;
        if (data.children && data.children.length > 0) {
            this.walk(data.children, data);
        }
    }
    walk = (children: Array, parent: TreeData): void => {
        children.map((item: TreeData) => {
            item.parent = parent;
            this.keyToNodeMap[item.key] = item;
            if (item.children && item.children.length > 0) {
                this.walk(item.children, item);
            }
        });
    }
+    onCollapse = async (key: string) => {
        let data = this.keyToNodeMap[key];
        if (data) {
+            let { children } = data;
+            if (!children) {
+                data.loading = true;
+                this.setState({ data: this.state.data });
+                let result = await getChildren(data);
+                if (result.code == 0) {
+                    data.children = result.data;
+                    data.collapsed = false;
+                    data.loading = false;
+                    this.buildKeyMap();
+                    this.setState({ data: this.state.data });
+                } else {
+                    alert('加载失败');
+                }
+            } else {
+                data.collapsed = !data.collapsed;
+                this.setState({ data: this.state.data });
+            }
        }
    }
    onCheck = (key: string) => {
        let data: TreeData = this.keyToNodeMap[key];
        if (data) {
            data.checked = !data.checked;
            if (data.checked) {
                this.checkChildren(data.children, true);
                this.checkParentCheckAll(data.parent);
            } else {
                this.checkChildren(data.children, false);
                this.checkParent(data.parent, false);
            }
            this.setState({ data: this.state.data });
        }
    }
    checkParentCheckAll = (parent: TreeData) => {
        while (parent) {
            parent.checked = parent.children.every(item => item.checked);
            parent = parent.parent;
        }
    }
    checkParent = (parent: TreeData, checked: boolean) => {
        while (parent) {
            parent.checked = checked;
            parent = parent.parent;
        }
    }
    checkChildren = (children: Array = [], checked: boolean) => {
        children.forEach((item: TreeData) => {
            item.checked = checked;
            this.checkChildren(item.children, checked);
        });
    }
    render() {
        return (

        )
    }
}
export default Tree;
```

7.3 tree-node.tsx

src\components\tree-node.tsx

```
import React from 'react';
import { TreeData } from '../typings';
import file from '../assets/file.png';
import closedFolder from '../assets/closed-folder.png';
import openedFolder from '../assets/opened-folder.png';
+import loadingSrc from '../assets/loading.gif';
interface Props {
    data: TreeData,
    onCollapse: any,
    onCheck: any
}
class TreeNode extends React.Component {
    constructor(props: Props) {
        super(props);
    }
    render() {
+        let { data: { name, children, collapsed = false, key, checked = false,loading } } = this.props;
        let caret, icon;
        if (children) {
            if (children.length > 0) {
                caret = (
                     this.props.onCollapse(key)}
                    />
                )
                icon = collapsed ? closedFolder : openedFolder;
            } else {
                caret = null;
                icon = file;
            }
        } else {
+            caret = (
                loading?: this.props.onCollapse(key)}
                />
            )
            icon = closedFolder;
        }
        return (

                    {caret}

                         this.props.onCheck(key)} />

                        {name}

                {
                    (children && children.length > 0 && !collapsed) && (

                            {
                                children.map((item: TreeData) => (

                                ))
                            }

                    )
                }

        )
    }
}
export default TreeNode;
```

7.4 src\api.tsx

src\api.tsx

```
import axios from 'axios';
import qs from 'qs';
axios.defaults.baseURL = 'http://localhost:3000';
export const getChildren = (data: any) => {
    return axios.get(`/getChildren?${qs.stringify({ key: data.key, name: data.name })}`).then(res => res.data).catch(function (error) {
        console.log(error);
    });
}
```

7.5 api.js

api.js

```
let express = require('express');
let app = express();
app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    if (req.method === 'OPTIONS') {
        return res.sendStatus(200);
    }
    next();
});
app.get('/getChildren', (req, res) => {
    let data = req.query;
    setTimeout(function () {
        res.json({
            code: 0,
            data: [
                {
                    name: data.name + '的儿子1',
                    key: `${data.key}-1`,
                    type: 'folder',
                    collapsed: true
                },
                {
                    name: data.name + '的儿子2',
                    key: `${data.key}-2`,
                    type: 'folder',
                    collapsed: true
                }
            ]
        });
    }, 2000)

});
app.listen(3000, () => {
    console.log(`接口服务器在${3000}上启动`);
});
```

8. 拖动排序

8.1 components\tree.tsx

src\components\tree.tsx

```
import React from 'react';
import './index.less';
import { TreeData } from '../typings';
import TreeNode from './tree-node';
import { getChildren } from '../api';
interface Props {
    data: TreeData;
}
interface KeyToNodeMap {
    [key: string]: TreeData
}
interface State {
    data: TreeData;
+    fromNode?: TreeData;
}
class Tree extends React.Component {
    data: TreeData;
    keyToNodeMap: KeyToNodeMap;
    constructor(props: Props) {
        super(props);
        this.state = { data: this.props.data };
        this.data = props.data;
        this.buildKeyMap();
    }
    buildKeyMap = () => {
        let data = this.data;
        this.keyToNodeMap = {};
        this.keyToNodeMap[data.key] = data;
        if (data.children && data.children.length > 0) {
            this.walk(data.children, data);
        }
        this.setState({ data: this.state.data });
    }
    walk = (children: Array, parent: TreeData): void => {
        children.map((item: TreeData) => {
            item.parent = parent;
            this.keyToNodeMap[item.key] = item;
            if (item.children && item.children.length > 0) {
                this.walk(item.children, item);
            }
        });
    }
    onCollapse = async (key: string) => {
        let data = this.keyToNodeMap[key];
        if (data) {
            let { children } = data;
            if (!children) {
                data.loading = true;
                this.setState({ data: this.state.data });
                let result = await getChildren(data);
                if (result.code == 0) {
                    data.children = result.data;
                    data.collapsed = false;
                    data.loading = false;
                    this.buildKeyMap();
                } else {
                    alert('加载失败');
                }
            } else {
                data.collapsed = !data.collapsed;
                this.setState({ data: this.state.data });
```

```
            }
        }
    }
    onCheck = (key: string) => {
        let data: TreeData = this.keyToNodeMap[key];
        if (data) {
            data.checked = !data.checked;
            if (data.checked) {
                this.checkChildren(data.children, true);
                this.checkParentCheckAll(data.parent);
            } else {
                this.checkChildren(data.children, false);
                this.checkParent(data.parent, false);
            }
            this.setState({ data: this.state.data });
        }
    }
    checkParentCheckAll = (parent: TreeData) => {
        while (parent) {
            parent.checked = parent.children.every(item => item.checked);
            parent = parent.parent;
        }
    }
    checkParent = (parent: TreeData, checked: boolean) => {
        while (parent) {
            parent.checked = checked;
            parent = parent.parent;
        }
    }
    checkChildren = (children: Array = [], checked: boolean) => {
        children.forEach((item: TreeData) => {
            item.checked = checked;
            this.checkChildren(item.children, checked);
        });
    }
+    setFromNode = (fromNode: TreeData) => {
+        this.setState({ ...this.state, fromNode });
+    }
+    onMove = (toNode: TreeData) => {
+        let fromNode = this.state.fromNode;
+        let fromChildren = fromNode.parent.children, toChildren = toNode.parent.children;
+        let fromIndex = fromChildren.findIndex((item: TreeData) => item === fromNode);
+        let toIndex = toChildren.findIndex(item => item === toNode);
+        fromChildren.splice(fromIndex, 1, toNode);
+        toChildren.splice(toIndex, 1, fromNode);
+        this.buildKeyMap();
+    }
    render() {
        return (

                    +                       setFromNode={this.setFromNode}
+                        onMove={this.onMove}
                    />

        )
    }
}
export default Tree;
```

8.2 tree-node.tsx

src\components\tree-node.tsx

```
import React from 'react';
import { TreeData } from '../typings';
import file from '../assets/file.png';
import closedFolder from '../assets/closed-folder.png';
import openedFolder from '../assets/opened-folder.png';
import loadingSrc from '../assets/loading.gif';
interface Props {
    data: TreeData,
    onCollapse: any,
    onCheck: any;
+   setFromNode: any;
+   onMove: any
}
class TreeNode extends React.Component {
    treeNodeRef: React.RefObject;
    constructor(props: Props) {
        super(props);
+        this.treeNodeRef = React.createRef();
    }
+    componentDidMount() {
+        this.treeNodeRef.current.addEventListener('dragstart', (event: DragEvent): void => {
+            this.props.setFromNode(this.props.data);
+            event.stopPropagation();
+        }, false);//useCapture=false
+        this.treeNodeRef.current.addEventListener('dragenter', (event: DragEvent) => {
+            event.preventDefault();
+            event.stopPropagation();
+        }, false);
+        this.treeNodeRef.current.addEventListener('dragover', (event: DragEvent) => {
+            event.preventDefault();
+            event.stopPropagation();
+        }, false);
+        this.treeNodeRef.current.addEventListener('drop', (event: DragEvent) => {
+            event.preventDefault();
+            this.props.onMove(this.props.data);
+            event.stopPropagation();
+        }, false);
+    }
    render() {
        let { data: { name, children, collapsed = false, key, checked = false, loading } } = this.props;
        let caret, icon;
        if (children) {
            if (children.length > 0) {
                caret = (
                     this.props.onCollapse(key)}
                    />
                )
                icon = collapsed ? closedFolder : openedFolder;
            } else {
                caret = null;
                icon = file;
            }
        } else {
            caret = (
                loading ?  :  this.props.onCollapse(key)}
                />
            )
            icon = closedFolder;
        }
        return (

                    {caret}

                         this.props.onCheck(key)} />

                        {name}

                {
                    (children && children.length > 0 && !collapsed) && (

                            {
                                children.map((item: TreeData) => (
                                    +                                        setFromNode={this.props.setFromNode}
+                                            onMove={this.props.onMove}
                                           data={item} />
                                ))
                            }

                    )
                }

        )
    }
}
export default TreeNode;
```