link: null
title: 珠峰架构师成长计划
description: 基本参数
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=266 sentences=527, words=4743

1. 搭建开发环境

```
mkdir zhufeng_typescript_react
cd zhufeng_typescript_react
npm init -y
cd zhufeng_typescript_react
```

2. 生成ts配置文件

- tsconfig-json (http://www.typescriptlang.org/docs/handbook/tsconfig-json.html)
- 编译选项 (http://www.typescriptlang.org/docs/handbook/compiler-options.html)

```
tsc --init
```

基本参数

参数 解释 target 用于指定编译之后的版本目标 module 生成的模块形式：none、commonjs、amd、system、umd、es6、es2015 或 esnext 只有 amd 和 system 能和 outFile 一起使用 target 为 es5 或更低时可用 es6 和 es2015 lib 编译时引入的 ES 功能库，包括：es5 、es6、es7、dom 等。如果未设置，则默认为： target 为 es5 时: ["dom", "es5", "scripthost"] target 为 es6 时: ["dom", "es6", "dom.iterable", "scripthost"] allowJs 是否允许编译JS文件，默认是false，即不编译JS文件 checkJs 是否检查和报告JS文件中的错误，默认是false jsx 指定jsx代码用于的开发环境 `preserve`

指保留JSX语法,扩展名为 `.jsx`

,react-native是指保留jsx语法，扩展名js,react指会编译成ES5语法
详解 (http://www.typescriptlang.org/docs/handbook/jsx.html)

declaration 是否在编译的时候生成相应的 `.d.ts`

声明文件 declarationDir 生成的 .d.ts 文件存放路径,默认与 .ts 文件相同 declarationMap 是否为声明文件.d.ts生成map文件 sourceMap 编译时是否生成 `.map`

文件 outFile 是否将输出文件合并为一个文件，值是一个文件路径名，只有设置 `module`

的值为 `amd` `system`

模块时才支持这个配置 outDir 指定输出文件夹 rootDir 编译文件的根目录，编译器会在根目录查找入口文件 composite 是否编译构建引用项目 removeComments 是否将编译后的文件中的注释删掉 noEmit 不生成编译文件 importHelpers 是否引入 `tslib`

里的辅助工具函数 downlevelIteration 当target为 `ES5` `ES3`

时，为 `for-of` `spread` `destructuring`

中的迭代器提供完全支持 isolatedModules 指定是否将每个文件作为单独的模块，默认为true

严格检查

参数 解释 strict 是否启动所有类型检查 noImplicitAny 不允许默认any类型 strictNullChecks 当设为true时，null和undefined值不能赋值给非这两种类型的值 strictFunctionTypes 是否使用函数参数双向协变检查 strictBindCallApply 是否对bind、call和apply绑定的方法的参数的检测是严格检测的 strictPropertyInitialization 检查类的非undefined属性是否已经在构造函数里初始化 noImplicitThis 不允许 `this`

表达式的值为 `any`

类型的时候 alwaysStrict 指定始终以严格模式检查每个模块

额外检查

参数 解释 noUnusedLocals 检查是否有定义了但是没有使用的变量 noUnusedParameters 检查是否有在函数体中没有使用的参数 noImplicitReturns 检查函数是否有返回值 noFallthroughCasesInSwitch 检查switch中是否有case没有使用break跳出

模块解析检查

参数 解释 moduleResolution 选择模块解析策略，有 `node` `classic`

两种类型,
详细说明 (http://www.typescriptlang.org/docs/handbook/module-resolution.html)

baseUrl 解析非相对模块名称的基本目录 paths 设置模块名到基于 `baseUrl`

的路径映射 rootDirs 可以指定一个路径列表，在构建时编译器会将这个路径列表中的路径中的内容都放到一个文件夹中 typeRoots 指定声明文件或文件夹的路径列表 types 用来指定需要包含的模块 allowSyntheticDefaultImports 允许从没有默认导出的模块中默认导入 esModuleInterop 为导入内容创建命名空间,实现CommonJS和ES模块之间的互相访问 preserveSymlinks 不把符号链接解析为其真实路径

sourcemap检查

参数 解释 sourceRoot 调试器应该找到TypeScript文件而不是源文件位置 mapRoot 调试器找到映射文件而非生成文件的位置，指定map文件的根路径 inlineSourceMap 指定是否将map文件的内容和js文件编译在一个同一个js文件中 inlineSources 是否进一步将.ts文件的内容也包含到输出文件中

试验选项

参数 解释 experimentalDecorators 是否启用实验性的装饰器特性 emitDecoratorMetadata 是否为装饰器提供元数据支持

试验选项

参数 解释 files 配置一个数组列表，里面包含指定文件的相对或绝对路径，编译器在编译的时候只会编译包含在files中列出的文件 include include也可以指定要编译的路径列表，但是和files的区别在于，这里的路径可以是文件夹，也可以是文件 exclude exclude表示要排除的、不编译的文件，他也可以指定一个列表 extends extends可以通过指定一个其他的tsconfig.json文件路径，来继承这个配置文件里的配置 compileOnSave 在我们编辑了项目中文件保存的时候，编辑器会根据 `tsconfig.json`

的配置重新生成文件 references 一个对象数组,指定要引用的项目

3.配置webpack

```
cnpm i typescript webpack webpack-cli webpack-dev-server ts-loader cross-env webpack-merge clean-webpack-plugin html-webpack-plugin -D
```

```
cnpm i babel-loader @babel/core @babel/cli @babel/plugin-proposal-class-properties @babel/plugin-proposal-object-rest-spread @babel/preset-env @babel/preset-typescript -D
```

3.1 package.json

```
  "scripts": {
    "dev": "cross-env NODE_ENV=development webpack-dev-server --config ./config/webpack.dev.js",
    "build": "cross-env NODE_ENV=production webpack --config ./config/webpack.prod.js"
  },
```

3.2 config\webpack.base.js

```
const { CleanWebpackPlugin } = require('clean-webpack-plugin');
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
    entry: "./src/index.tsx",
    output: {
        filename: "main.js"
    },
    resolve: {
        extensions: ['.ts', '.tsx', '.js']
    },
    module: {
        rules: [{
            test: /\.tsx?$/,
            use: 'ts-loader',
            exclude: /node_modules/
        }]
    },

    devServer: {
        contentBase: './dist'
    },
    plugins: [
        new CleanWebpackPlugin({
            cleanOnceBeforeBuildPatterns: ['./dist']
        }),
        new HtmlWebpackPlugin({
            template: './src/index.html'
        })
    ]
}
```

3.3 config\webpack.dev.js

config\webpack.dev.js

```
const { smart } = require('webpack-merge');
const base = require('./webpack.base');
module.exports = smart(base, {
    mode: 'development',
    devtool: 'inline-source-map'
});
```

3.4 config\webpack.prod.js

config\webpack.prod.js

```
const { smart } = require('webpack-merge');
const base = require('./webpack.base');
module.exports = smart(base, {
    mode: 'production',
    devtool: false
});
```

4.配置eslint

4.1 安装

```
cnpm i eslint @typescript-eslint/eslint-plugin @typescript-eslint/parser -D
```

4.2 .eslintrc.json

.eslintrc.json

```
{
    "parser": "@typescript-eslint/parser",
    "plugins": [
        "@typescript-eslint/eslint-plugin"
    ],
    "extends": [
        "plugin:@typescript-eslint/recommended"
    ],
    "rules": {
        "@typescript-eslint/no-unused-vars": "off"
    }
}
```

4.3 package.json

```
  "scripts": {
    "dev": "cross-env NODE_ENV=development webpack-dev-server --config ./config/webpack.dev.js",
    "build": "cross-env NODE_ENV=production webpack --config ./config/webpack.prod.js",
+    "eslint": "eslint src --ext .js,.ts,.tsx"
  },
```

5.配置jest

5.1 安装配置

```
cnpm i jest @types/jest ts-jest -D
npx ts-jest config:init
```

5.2 src\calculator.tsx

src\calculator.tsx

```
function sum(a: number, b: number) {
    return a + b;
}
function minus(a: number, b: number) {
    return a - b;
}
module.exports = {
    sum,
    minus
}
```

5.3 tests\calculator.spec.tsx

tests\calculator.spec.tsx

```
let math = require('../src/calculator');
test('1+1=2', () => {
    expect(math.sum(1, 1)).toBe(2);
});
test('1-1=0', () => {
    expect(math.minus(1, 1)).toBe(0);
});
```

5.4 package.json

package.json

```
  "scripts": {
+    "test": "jest"
  },
```

6.支持react

6.1 安装

```
cnpm i react @types/react react-dom @types/react-dom -S
```

6.2 src\index.tsx

src\index.tsx

```
import React, { ReactElement, DOMElement, DetailedReactHTMLElement } from 'react';
import ReactDOM from 'react-dom';
let root: HTMLElement | null = document.getElementById('root');
interface Props {
    className: string
}
let props: Props = { className: 'title' };
let element: DetailedReactHTMLElement = React.createElement('h1', props, 'hello');
ReactDOM.render(element, root);
```

6.3 typings\react.tsx

src\typings\react.tsx

```
export interface ReactHTML { h1: any }

export type ReactText = string | number;
export type ReactChild = ReactElement | ReactText;
export type ReactNode = ReactChild | boolean | null | undefined;

export interface RefObject {
    readonly current: T | null;
}

export interface ReactElement {
    type: T;
    props: P;
    key: Key | null;
}
export interface DOMElement, T extends Element> extends ReactElement {
    ref: RefObject;
}

export interface DetailedReactHTMLElement, T extends HTMLElement> extends DOMElement {
    type: keyof ReactHTML;
}
export type Key = string | number;

export interface Attributes {
    key?: Key;
}
export interface ClassAttributes extends Attributes {
    ref?: RefObject;
}
export interface DOMAttributes {
    children?: ReactNode;
}
export interface HTMLAttributes extends DOMAttributes {
    className?: string;
}
export interface Element { }
export interface HTMLElement extends Element { }
export declare function createElement<P extends HTMLAttributes<T>, T extends HTMLElement>(
    type: keyof ReactHTML,
    props?: ClassAttributes & P | null,
    ...children: ReactNode[]): DetailedReactHTMLElement<P, T>;
```

7.函数组件声明

7.1 src\index.tsx

src\index.tsx

```
import React, { ReactElement, DOMElement, DetailedReactHTMLElement, FunctionComponentElement } from 'react';
import ReactDOM from 'react-dom';
let root: HTMLElement | null = document.getElementById('root');
+interface Props {
+    className: string;
+}
+function Welcome(props: Props) {
+    return React.createElement('h1', { className: props.className }, 'hello');
+}
+let props: Props = { className: 'title' };
+let element: FunctionComponentElement = React.createElement(Welcome, props);
ReactDOM.render(element, root);
```

7.2 typings\react.tsx

src\typings\react.tsx

```
export interface ReactHTML { h1: any }

export type ReactText = string | number;
export type ReactChild = ReactElement | ReactText;
export type ReactNode = ReactChild | boolean | null | undefined;

export interface RefObject {
    readonly current: T | null;
}
export type JSXElementConstructor =
    | ((props: P) => ReactElement | null)
export interface ReactElement = string | JSXElementConstructor> {
    type: T;
    props: P;
    key: Key | null;
}
export interface DOMElement, T extends Element> extends ReactElement {
    ref: RefObject;
}

export interface DetailedReactHTMLElement, T extends HTMLElement> extends DOMElement {
    type: keyof ReactHTML;
}
export type Key = string | number;

export interface Attributes {
    key?: Key;
}
export interface ClassAttributes extends Attributes {
    ref?: RefObject;
}
export interface DOMAttributes {
    children?: ReactNode;
}
export interface HTMLAttributes extends DOMAttributes {
    className?: string;
}
export interface Element { }
export interface HTMLElement extends Element { }

+type PropsWithChildren = P & { children?: ReactNode };
+interface FunctionComponent {
+    (props: PropsWithChildren): ReactElement | null;
+}
+interface FunctionComponentElement extends ReactElement> {
+    ref?: 'ref' extends keyof P ? P extends { ref?: infer R } ? R : never : never;
+}
+export declare function createElement(
+    type: FunctionComponent,
+    props?: Attributes & P | null,
+    ...children: ReactNode[]): FunctionComponentElement;
export declare function createElement, T extends HTMLElement>(
    type: keyof ReactHTML,
    props?: ClassAttributes & P | null,
    ...children: ReactNode[]): DetailedReactHTMLElement;
```

8.类组件声明

8.1 src\index.tsx

src\index.tsx

```
import React, { ReactElement, DOMElement, DetailedReactHTMLElement, FunctionComponentElement, ComponentClass } from 'react';
import ReactDOM from 'react-dom';
let root: HTMLElement | null = document.getElementById('root');
interface Props {
    className: string;
}
+interface State {
+    id: string
+}
+class Welcome extends React.Component {
+    state = { id: 'id' }
+    render() {
+        return React.createElement('h1', { className: props.className }, 'hello');
+    }
+}

+let props: Props = { className: 'title' };
+let element: ReactElement = React.createElement(Welcome, props);
ReactDOM.render(element, root);
```

8.2 typings\react.tsx

src\typings\react.tsx

```
export interface ReactHTML { h1: any }

export type ReactText = string | number;
export type ReactChild = ReactElement | ReactText;
export type ReactNode = ReactChild | boolean | null | undefined;

export interface RefObject {
    readonly current: T | null;
}
export type JSXElementConstructor =
    | ((props: P) => ReactElement | null)
export interface ReactElement = string | JSXElementConstructor> {
    type: T;
    props: P;
    key: Key | null;
}
export interface DOMElement, T extends Element> extends ReactElement {
    ref: RefObject;
}

export interface DetailedReactHTMLElement, T extends HTMLElement> extends DOMElement {
    type: keyof ReactHTML;
}
export type Key = string | number;

export interface Attributes {
    key?: Key;
}
export interface ClassAttributes extends Attributes {
    ref?: RefObject;
}
export interface DOMAttributes {
    children?: ReactNode;
}
export interface HTMLAttributes extends DOMAttributes {
    className?: string;
}
export interface Element { }
export interface HTMLElement extends Element { }

type PropsWithChildren = P & { children?: ReactNode };
interface FunctionComponent {
    (props: PropsWithChildren): ReactElement | null;
}
interface FunctionComponentElement extends ReactElement> {
    ref?: 'ref' extends keyof P ? P extends { ref?: infer R } ? R : never : never;
}
+type ComponentState = any;
+declare class Component {
+    setState(
+        state: ((prevState: Readonly, props: Readonly) => (Pick | S | null)) | (Pick | S | null),
+        callback?: () => void
+    ): void;
+    render(): ReactNode;
+}

+interface ComponentClass {
+    new(props: P, context?: any): Component;
+}
+export declare function createElement(
+    type: FunctionComponent | ComponentClass | string,
+    props?: Attributes & P | null,
+    ...children: ReactNode[]): ReactElement;
```

9.使用redux

```
cnpm i redux react-redux @types/react-redux redux-logger redux-promise redux-thunk   @types/redux-logger @types/redux-promise -D
```

9.1 src\index.tsx

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
+import Counter1 from './components/Counter1';
+import Counter2 from './components/Counter2';
+import store from './store';
+import { Provider } from 'react-redux';
+ReactDOM.render(
+
+
+
+    , document.getElementById('root'));
```

9.2 src\store\index.tsx

```
import { createStore, Store, AnyAction, applyMiddleware, StoreEnhancer, StoreEnhancerStoreCreator } from 'redux';
import logger from 'redux-logger';
import thunk from 'redux-thunk';
import promise from 'redux-promise';
import reducer from './reducers';
import { CombinedState } from './reducers';

const storeEnhancer: StoreEnhancer = applyMiddleware(promise, thunk, logger);
const storeEnhancerStoreCreator: StoreEnhancerStoreCreator = storeEnhancer(createStore);
const store: Store = storeEnhancerStoreCreator(reducer);
export default store;
```

9.3 store\action-types.tsx

src\store\action-types.tsx

```
export const INCREMENT1 = 'INCREMENT1';
export const DECREMENT1 = 'DECREMENT1';

export const INCREMENT2 = 'INCREMENT2';
export const DECREMENT2 = 'DECREMENT2';
```

9.4 reducers\counter1.tsx

src\store\reducers\counter1.tsx

```
import * as types from '../action-types';
import { AnyAction } from 'redux';
export interface Counter1State {
    number: number
}
let initialState: Counter1State = { number: 0 }

export default function (state: Counter1State = initialState, action: AnyAction): Counter1State {
    switch (action.type) {
        case types.INCREMENT1:
            return { number: state.number + 1 };
        case types.DECREMENT1:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

9.5 reducers\counter2.tsx

src\store\reducers\counter2.tsx

```
import * as types from '../action-types';
import { AnyAction } from 'redux';
export interface Counter2State {
    number: number
}
let initialState: Counter2State = { number: 0 };

export default function (state: Counter2State = initialState, action: AnyAction): Counter2State {
    switch (action.type) {
        case types.INCREMENT2:
            return { number: state.number + 1 };
        case types.DECREMENT2:
            return { number: state.number - 1 };
        default:
            return state;
    }
}
```

9.6 reducers\index.tsx

src\store\reducers\index.tsx

```
import { combineReducers, ReducersMapObject, Reducer, AnyAction } from 'redux';
import counter1, { Counter1State } from './counter1';
import counter2, { Counter2State } from './counter2';
interface Reducers {
    counter1: Counter1State;
    counter2: Counter2State;
}
let reducers: ReducersMapObject = {
    counter1,
    counter2
};
export type CombinedState = {
    [key in keyof typeof reducers]: ReturnType<typeof reducers[key]>
}
let rootReducer: Reducer = combineReducers(reducers);
export default rootReducer;
```

9.7 actions\counter1.tsx

src\store\actions\counter1.tsx

```
import * as types from '../action-types';
import { AnyAction, Dispatch, Store } from 'redux';
const actions = {
    increment1(): AnyAction {
        return { type: types.INCREMENT1 };
    },
    promisePlus(): { type: string, payload: Promise<undefined> } {
        return (
            {
                type: types.INCREMENT1,
                payload: new Promise((resolve) => {
                    setTimeout(() => {
                        resolve();
                    }, 1000);
                })
            }
        )
    },
    thunkPlus(): Function {
        return (dispatch: Dispatch, getState: Store['getState']) => {
            setTimeout(() => {
                dispatch({ type: types.INCREMENT1 });
            }, 1000);
        }
    },
    decrement1(): AnyAction {
        return { type: types.DECREMENT1 };
    }
}
export default actions;
```

9.8 actions\counter2.tsx

src\store\actions\counter2.tsx

```
import * as types from '../action-types';
import { AnyAction } from 'redux';
export default {
    increment2(): AnyAction {
        return { type: types.INCREMENT2 };
    },
    decrement2(): AnyAction {
        return { type: types.DECREMENT2 };
    }
}
```

9.9 src\components\Counter1.tsx

src\components\Counter1.tsx

```
import React, { Component } from 'react';
import actions from '../store/actions/counter1';
import { CombinedState } from '../store/reducers';
import { Counter1State } from '../store/reducers/counter1';
import { connect } from 'react-redux';
type Props = Counter1State & typeof actions;
class Counter1 extends Component<Props> {
    render() {
        let { number, increment1, promisePlus, thunkPlus } = this.props;
        return (
            <div>
                <p>{number}p>
                <button onClick={increment1}>+button>
                <button onClick={promisePlus}>promise+button>
                <button onClick={thunkPlus}>thunk+button>
            div>
        )
    }
}

let mapStateToProps = (state: CombinedState): Counter1State => state.counter1;
export default connect(
    mapStateToProps,
    actions
)(Counter1)
```

9.10 src\components\Counter2.tsx

src\components\Counter2.tsx

```
import React, { Component } from 'react';
import actions from '../store/actions/counter2';
import { CombinedState } from '../store/reducers';
import { Counter2State } from '../store/reducers/counter2';
import { connect } from 'react-redux';
type Props = Counter2State & typeof actions;
class Counter2 extends Component<Props> {
    render() {
        let { number, increment2, decrement2 } = this.props;
        return (
            <div>
                <p>{number}p>
                <button onClick={increment2}>+button>
                <button onClick={decrement2}>-button>
            div>
        )
    }
}
let mapStateToProps = (state: CombinedState): Counter2State => state.counter2;
type ThunkDispatchs = ThunkDispatchamount: number }, AnyAction>;
function mapDispatchToProps(dispatch: ThunkDispatchs) {
    return ({
        thunkPlus: () => dispatch<void>((dispatch: ThunkDispatchs, getState: Store['getState'], extraArgument: { amount: number }): void => {
            setTimeout(() => {
                dispatch({ type: types.INCREMENT1, payload: extraArgument.amount });
            }, 1000);
        })
    })
}
export default connect(
    mapStateToProps,
    actions
)(Counter2)
```

9.11 redux-thunk\index.tsx

src\redux-thunk\index.tsx

```
import { Middleware, Action, AnyAction } from 'redux';
type MiddlewareExt = Middleware & {
    withExtraArgument: typeof createThunkMiddleware
}
export type ThunkAction = (
    dispatch: ThunkDispatch,
    getState: () => S,
    extraArgument: E
) => R;
export interface ThunkDispatch {
    (action: T): T;
    (asyncAction: ThunkAction): R;
}
function createThunkMiddleware(extraArgument?: any): Middleware {
    let middleware: Middleware, S, ThunkDispatch> = ({ dispatch, getState }) => next => action => {
        if (typeof action === 'function') {
            return action(dispatch, getState, extraArgument);
        }

        return next(action);
    };
    return middleware;
}

const thunk: MiddlewareExt = createThunkMiddleware() as MiddlewareExt;
thunk.withExtraArgument = createThunkMiddleware;

export default thunk;
```

10.支持路由

10.1 安装

```
cnpm i react-router-dom @types/react-router-dom  connected-react-router -S
```

10.2 src\index.tsx

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
import { Provider } from 'react-redux';
+import { Route, Link, Redirect, Switch } from 'react-router-dom';
+import { ConnectedRouter } from 'connected-react-router';
+import history from './history';
+import store from './store';
ReactDOM.render(

+
+            Counter1
+            Counter2
+            用户管理
+
+
+
+
+
+
+     , document.getElementById('root'));
```

10.2 src\history.tsx

src\history.tsx

```
import { createHashHistory } from 'history'
let history = createHashHistory();
export default history;
```

10.3 store\index.tsx

src\store\index.tsx

```
import { createStore, Store, AnyAction, applyMiddleware, StoreEnhancer, StoreEnhancerStoreCreator } from 'redux';
import logger from 'redux-logger';
import thunk from 'redux-thunk';
import promise from 'redux-promise';
import reducer from './reducers';
import { CombinedState } from './reducers';
+import { routerMiddleware } from 'connected-react-router';
+import history from '../history';
+const storeEnhancer: StoreEnhancer = applyMiddleware(routerMiddleware(history), promise, thunk, logger);
const storeEnhancerStoreCreator: StoreEnhancerStoreCreator = storeEnhancer(createStore);
const store: Store = storeEnhancerStoreCreator(reducer);
export default store;
```

10.4 reducers\index.tsx

src\store\reducers\index.tsx

```
import { combineReducers, ReducersMapObject, Reducer, AnyAction } from 'redux';
import counter1, { Counter1State } from './counter1';
import counter2, { Counter2State } from './counter2';
+import { LocationState } from 'history';
+import { connectRouter, RouterState } from 'connected-react-router';
import history from '../../history';
interface Reducers {
+    router: RouterState,
    counter1: Counter1State;
    counter2: Counter2State;
}
+let reducers: ReducersMapObject = {
+    router: connectRouter(history),
    counter1,
    counter2
};
export type CombinedState = {
    [key in keyof typeof reducers]: ReturnType
}
let rootReducer: Reducer = combineReducers(reducers);
export default rootReducer;
```

10.5 store\actions\counter1.tsx

src\store\actions\counter1.tsx

```
import * as types from '../action-types';
import { AnyAction, Dispatch, Store } from 'redux';
+import { LocationDescriptorObject } from 'history';
+import { push } from 'connected-react-router';
const actions = {
    increment1(): AnyAction {
        return { type: types.INCREMENT1 };
    },
    promisePlus(): { type: string, payload: Promise } {
        return (
            {
                type: types.INCREMENT1,
                payload: new Promise((resolve) => {
                    setTimeout(() => {
                        resolve();
                    }, 1000);
                })
            }
        )
    },
    thunkPlus(): Function {
        return (dispatch: Dispatch, getState: Store['getState']) => {
            setTimeout(() => {
                dispatch({ type: types.INCREMENT1 });
            }, 1000);
        }
    },
+    goto(locationDescriptorObject: LocationDescriptorObject): AnyAction {
+        return push(locationDescriptorObject);
+    }
}
export default actions;
```

10.6 components\Counter1.tsx

src\components\Counter1.tsx

```
+import React, { Component, PropsWithChildren } from 'react';
import actions from '../store/actions/counter1';
import { CombinedState } from '../store/reducers';
import { Counter1State } from '../store/reducers/counter1';
import { connect } from 'react-redux';
+import { RouteComponentProps } from 'react-router';
+interface IParams { }
+type RouteProps = RouteComponentProps;
+type Props = PropsWithChildren;
class Counter1 extends Component {
    render() {
+        let { number, increment1, promisePlus, thunkPlus, goto } = this.props;
        return (

                {number}
                +
                promise+
                thunk+
+                  goto({ pathname: '/counter2' })}>/counter2

        )
    }
}

let mapStateToProps = (state: CombinedState): Counter1State => state.counter1;
export default connect(
    mapStateToProps,
    actions
)(Counter1)
```

10.7 components\Counter2.tsx

src\components\Counter2.tsx

```
+import React, { Component, PropsWithChildren } from 'react';
import actions from '../store/actions/counter2';
import { CombinedState } from '../store/reducers';
import { Counter2State } from '../store/reducers/counter2';
import { connect } from 'react-redux';
+import { RouteComponentProps } from 'react-router';
+interface IParams { }
+type RouteProps = RouteComponentProps;
+type Props = PropsWithChildren;
class Counter2 extends Component {
    render() {
        let { number, increment2, decrement2 } = this.props;
        return (

                {number}
                +
                -

        )
    }
}
let mapStateToProps = (state: CombinedState): Counter2State => state.counter2;
export default connect(
    mapStateToProps,
    actions
)(Counter2)
```

11.路由和antdesign

11.1 安装依赖

```
cnpm i redux-thunk antd axios react-router-dom connected-react-router  -S
cnpm i style-loader css-loader  @types/react-router-dom -D
```

11.2 config\webpack.base.js

config\webpack.base.js

```
    module: {
        rules: [
+            {
+                test: /\.css?$/,
+                use: [
+                    'style-loader',
+                    'css-loader'
+                ]
+            }
+        ]
    },
```

11.3 src\index.tsx

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import Counter1 from './components/Counter1';
import Counter2 from './components/Counter2';
+import User from './components/User';
import { Provider } from 'react-redux';
import { Route, Link, Redirect, Switch } from 'react-router-dom';
import { ConnectedRouter } from 'connected-react-router';
import history from './history';
import store from './store';
+import { Layout } from 'antd';
+import NavBar from './components/NavBar';
+import 'antd/dist/antd.css';
+const { Content } = Layout;
ReactDOM.render(

+
+
+
+
+
+
+
+
+
+
+

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

11.4 action-types.tsx

src\store\action-types.tsx

```
+ export const SET_USERS = 'SET_USERS';
```

11.5 reducers\index.tsx

src\store\reducers\index.tsx

```
import { combineReducers, ReducersMapObject, Reducer, AnyAction } from 'redux';
import counter1, { Counter1State } from './counter1';
import counter2, { Counter2State } from './counter2';
+import user, { UserState } from './user';
import { LocationState } from 'history';
import { connectRouter, RouterState } from 'connected-react-router';
import history from '../../history';
interface Reducers {
    router: RouterState,
    counter1: Counter1State;
    counter2: Counter2State;
+    user: UserState
}
let reducers: ReducersMapObject = {
    router: connectRouter(history),
    counter1,
    counter2,
+    user
};
export type CombinedState = {
    [key in keyof typeof reducers]: ReturnType
}
let rootReducer: Reducer = combineReducers(reducers);
export default rootReducer;
```

11.6 typings\response.tsx

src\typings\response.tsx

```
export interface User {
    _id: string;
    username: string
}
export interface UserlistResponse {
    code: number;
    data: Array
}
export interface AddUserResponse {
    code: number;
    data: User
}
```

11.7 store\reducers\user.tsx

src\store\reducers\user.tsx

```
import * as types from '../action-types';
import { AnyAction } from 'redux';
import { User } from '../../typings/response';
export interface UserState {
    list: Array
}
let initialState: UserState = {
    list: []
};

export default function (state: UserState = initialState, action: AnyAction): UserState {
    switch (action.type) {
        case types.SET_USERS:
            return { ...state, list: action.payload };
        default:
            return state;
    }
}
```

11.8 store\actions\user.tsx

src\store\actions\user.tsx

```
import * as types from '../action-types';
import { User } from '../../typings/response';
export default {
    setUsers(list: Array) {
        return { type: types.SET_USERS, payload: list };
    }
}
```

11.9 src\api\request.tsx

src\api\request.tsx

```
import axios from 'axios';

const instance = axios.create({
    timeout: 20000,
    baseURL: 'http://localhost:4000'
});
export * from 'axios';
export default instance;
```

src\components\NavBar.tsx

```
import React, { Component, PropsWithChildren } from 'react';
import { Link, withRouter } from 'react-router-dom';
import { Layout, Menu } from 'antd';
import { RouteComponentProps } from 'react-router';
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = PropsWithChildren;
class NavBar extends Component<Props> {
    render() {
        return (
            <Layout.Header>
                <Menu
                    theme="dark"
                    style={{ lineHeight: '64px' }}
                    mode="horizontal"
                    selectedKeys={[this.props.location.pathname]} >
                    <Menu.Item key="/counter1"><Link to="/counter1">Counter1Link>Menu.Item>
                    <Menu.Item key="/counter2"><Link to="/counter2">Counter2Link>Menu.Item>
                    <Menu.Item key="/user"><Link to="/user">用户管理Link>Menu.Item>
                Menu>
            Layout.Header>
        )
    }
}
export default withRouter(NavBar);
```

11.11 components\User.tsx

src\components\User.tsx

```
import React, { Component, PropsWithChildren } from 'react';
import actions from '../store/actions/user';
import { CombinedState } from '../store/reducers';
import { UserState } from '../store/reducers/user';
import { connect } from 'react-redux';
import { RouteComponentProps, Link, Route } from 'react-router-dom';
import { Layout, Menu, Icon } from 'antd';
import UserList from './UserList';
import UserAdd from './UserAdd';
const { Sider, Content } = Layout;
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = PropsWithChildrentypeof actions>;
class User extends Component<Props> {
    render() {
        return (
            <Layout>
                <Sider>
                    <Menu
                        theme="dark"
                        defaultSelectedKeys={['/user/list']}
                        mode="inline"
                    >
                        <Menu.Item key={'/user/add'}>
                            <Link to={'/user/add'}><Icon type={'plus'} />添加用户Link>
                        Menu.Item>;
                        <Menu.Item key={'/user/list'}>
                            <Link to={'/user/list'}><Icon type={'user'} />用户列表Link>
                        Menu.Item>;
                    Menu>
                Sider>
                <Content style={{ padding: '20px' }}>
                    <Route path="/user/list" component={UserList} />
                    <Route path="/user/add" component={UserAdd} />
                Content>
            Layout>
        )
    }
}
let mapStateToProps = (state: CombinedState): UserState => state.user;
export default connect(
    mapStateToProps,
    actions
)(User)
```

11.12 src\components\UserAdd.tsx

src\components\UserAdd.tsx

```
import React, { Component, PropsWithChildren, useState, useCallback } from 'react';
import { Link, withRouter } from 'react-router-dom';
import { Layout, Menu, Form, Input, Button, message } from 'antd';
import { RouteComponentProps } from 'react-router';
import request, { AxiosResponse } from '../api/request';
import { User, AddUserResponse } from '../typings/response';
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = PropsWithChildren;

function UserAdd(props: Props) {
    let [user, setUser] = useState({} as User);
    const handleSubmit = useCallback((event: React.FormEvent) => {
        event.preventDefault();
        request.post('/api/users', user).then((response: AxiosResponse) => {
            let data = response.data;
            if (data.code == 0) {
                props.history.push('/user/list');
            } else {
                message.error('添加用户失败!');
            }
        });
    }, [user]);
    const handleNameChange = useCallback((event: React.ChangeEvent) => {
        setUser({
            ...user,
            username: event.target.value
        });
    }, [user])
    return (
        <Form onSubmit={handleSubmit}>
            <Form.Item>
                <Input
                    placeholder="用户名"
                    style={{ width: 120 }}
                    value={user.username}
                    onChange={handleNameChange}
                />
            Form.Item>
            <Form.Item>
                <Button type="primary" htmlType="submit">添加Button>
            Form.Item>
        Form>
    )
}

export default UserAdd;
```

11.13 src\components\UserList.tsx

src\components\UserList.tsx

```
import React, { PropsWithChildren, useState, useEffect } from 'react';
import { RouteComponentProps } from 'react-router-dom';
import { Table } from 'antd';
import request, { AxiosResponse } from '../api/request';
import actions from '../store/actions/user';
import { User, UserlistResponse } from '../typings/response';
import { CombinedState } from '../store/reducers';
import { UserState } from '../store/reducers/user';
import { connect } from 'react-redux';

interface IParams { }
type RouteProps = RouteComponentProps;
type Props = PropsWithChildrentypeof actions>;
const columns = [
    {
        title: '用户名',
        dataIndex: 'username',
        key: 'username'
    }
];

function UserList(props: Props) {
    let [users, setUsers] = useState<Array>(props.list);
    useEffect(() => {
        (async function () {
            if (users.length == 0) {
                let response: AxiosResponse = await request.get>('/api/users');
                let list = response.data.data;
                setUsers(list);
                props.setUsers(list);
            }
        })()
    }, []);
    return (
        <Table columns={columns} dataSource={users} rowKey={(record: User) => record._id} />
    )
}
let mapStateToProps = (state: CombinedState): UserState => state.user;
export default connect(
    mapStateToProps,
    actions
)(UserList);
```

12. 后台接口

12.1. 初始化项目

```
mkdir C:\vipdata\prepare8\zhufeng_ts_react_api
cd zhufeng_ts_react_api
cnpm init -y
cnpm i @types/node express @types/express body-parser cors @types/cors mongoose @types/mongoose shelljs -S
```

12.2 tsconfig.json

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "lib": [
      "dom",
      "es2015"
    ],
    "outDir": "./dist",
    "strict": true,
    "baseUrl": "./",
    "paths": {
      "*": [
        "*",
        "node_modules/*",
        "typings/*"
      ]
    },
    "esModuleInterop": true,
  }
}
```

12.3 server.ts

server.ts

```
import express, { Express, Request, Response } from 'express';
import bodyParser from 'body-parser';
import cors from 'cors';
import Models from './db';
import config from './config';
import path from 'path';
let app: Express = express();
app.use(
    cors({
        origin: config.origin,
        credentials: true,
        allowedHeaders: "Content-Type,Authorization",
        methods: "GET,HEAD,PUT,PATCH,POST,DELETE,OPTIONS"
    })
);
app.use(express.static(path.resolve(__dirname, 'public')));
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.get('/api/users', async (req: Request, res: Response) => {
    let users = await Models.UserModel.find();
    res.json({
        code: 0,
        data: users
    });
});
app.post('/api/users', async (req: Request, res: Response) => {
    let user = req.body;
    user = await Models.UserModel.create(user);
    res.json({
        code: 0,
        data: user
    });
});
app.listen(4000, () => {
    console.log('服务器在4000端口启动!');
});
```

12.4 config.ts

```
interface IConfig {
    secret: string;
    dbUrl: string;
    origin: Array
}
let config: IConfig = {
    secret: 'zhufengcms',
    dbUrl: "mongodb://localhost:27017/zhufengcms",
    origin: ["http://localhost:8080"]
}

export =  config;
```

12.5 db.ts

db.ts

```
import mongoose, { Schema, Connection, Model } from 'mongoose';
import config from './config';
const conn: Connection = mongoose.createConnection(config.dbUrl, { useNewUrlParser: true, useUnifiedTopology: true });
const UserModel = conn.model('User', new Schema({
    username: { type: String },
}));

export = {
    UserModel
}
```

12.6 shelljs\index.d.ts

typings\shelljs\index.d.ts

```
declare module 'shelljs';
```

12.7 copy.ts

copy.ts

```
import shell from 'shelljs';
shell.cp("-R", "./public/", "./dist/");
```

12.8 package.json

package.json

```
{
"scripts": {
    "dev": "cross-env NODE_ENV=development nodemon -e ts,tsx --exec 'ts-node' ./server.ts",
    "serve": "cross-env NODE_ENV=production  tsc && ts-node copy.ts && nodemon ./dist/server.js"
  }
}
```

13. dva

```
import React from 'react';
import { Dispatch } from 'redux';
import dva, { connect } from 'dva';
import keymaster from 'keymaster';
import { RouterAPI } from 'dva';
import { Router, Route } from 'dva/router';
interface Counter1State {
    number: 0
}
interface Counter2State {
    number: 0
}
interface CombinedState {
    counter1: Counter1State;
    counter2: Counter2State;
}
const app = dva();
const delay = (millseconds: number) => {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            resolve();
        }, millseconds);
    });
}
app.model({
    namespace: 'counter1',
    state: { number: 0 },
    reducers: {//接收老状态，返回新状态
        add(state) { //dispatch({type:'add'});
            return { number: state.number + 1 };
        },
        minus(state) {//dispatch({type:'minus'})
            return { number: state.number - 1 };
        }
    },
    // 延时操作 调用接口  等待
    effects: {
        *asyncAdd(action, { put, call }) { //redux-saga/effects {put,call}
            yield call(delay, 1000);//把100传给delay并调用，yield会等待promise完成
            yield put({ type: 'add' });
        }
    },
    subscriptions: {
        keyboard({ dispatch }) {
            keymaster('space', () => {
                dispatch({ type: 'add' });
            });
        },
        changeTitle({ history }) {
            setTimeout(function () {
                history.listen(({ pathname }) => {
                    document.title = pathname;
                });
            }, 1000);

        }
    }
});
app.model({
    namespace: 'counter2',
    state: { number: 0 },
    reducers: {//接收老状态，返回新状态
        add(state) { //dispatch({type:'add'});
            return { number: state.number + 1 };
        },
        minus(state) {//dispatch({type:'minus'})
            return { number: state.number - 1 };
        }
    }
});
type Counter1Props = Counter1State & { dispatch: Dispatch };
const Counter1 = (props: Counter1Props) => {
    return (

            {props.number}
             props.dispatch({ type: 'counter1/add' })}>add
             props.dispatch({ type: 'counter1/asyncAdd' })}>asyncAdd
             props.dispatch({ type: 'counter1/minus' })}>-

    )
}
type Counter2Props = Counter2State & { dispatch: Dispatch };
const Counter2 = (props: Counter2Props) => {
    return (

            {props.number}
             props.dispatch({ type: 'counter2/add' })}>+
             props.dispatch({ type: 'counter2/minus' })}>-

    )
}

const mapStateToProps1 = (state: CombinedState): Counter1State => state.counter1;
const ConnectedCounter = connect(
    mapStateToProps1
)(Counter1);
const mapStateToProps2 = (state: CombinedState): Counter2State => state.counter2;
const ConnectedCounter2 = connect(
    mapStateToProps2
)(Counter2);
app.router(
    (api?: RouterAPI) => {
        let { history } = api!;
        return (
            (

                    <>

                    </>

            )
        )
    }
);
app.start('#root');
```