link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=351 sentences=589, words=4854

1. TypeScript工程化开发

- 前端工程化就是通过流程规范化、标准化提升团队协作效率
- 通过组件化、模块化提升代码质量
- 使用构建工具、自动化工具提升开发效率
- 编译 => 打包(合并) => 压缩 => 代码检查 => 测试 => 持续集成

2.初始化项目

```
mkdir zhufeng_typescript_development
cd zhufeng_typescript_development
npm init
package name: (zhufeng_typescript_development)
version: (1.0.0)
description: TypeScript工程化开发
entry point: (index.js)
test command:
git repository: https:
keywords: typescript,react
author: zhangrenyang
license: (ISC) MIT
```

3. git规范和changelog

3.1 良好的git commit好处

- 可以加快code review 的流程
- 可以根据git commit 的元数据生成changelog
- 可以让其它开发者知道修改的原因

3.2 良好的commit

- commitizen (https://www.npmjs.com/package/commitizen)是一个格式化commit message的工具
- validate-commit-msg (https://www.npmjs.com/package/validate-commit-msg) 用于检查项目的 `Commit message` 是否符合格式
- conventional-changelog-cli (https://www.npmjs.com/package/conventional-changelog-cli)可以从 `git metadata`生成变更日志
- 统一团队的git commit 标准
- 可以使用 `angular`的 `git commit`日志作为基本规范
    - 提交的类型限制为 feat、fix、docs、style、refactor、perf、test、chore、revert等
    - 提交信息分为两部分，标题(首字母不大写，末尾不要加标点)、主体内容(描述修改内容)
- 日志提交友好的类型选择提示 使用commitize工具
- 不符合要求格式的日志拒绝提交 的保障机制
    - 需要使用 `validate-commit-msg`工具
- 统一changelog文档信息生成
    - 使用 `conventional-changelog-cli`工具

```
cnpm i commitizen  validate-commit-msg conventional-changelog-cli -D
commitizen init cz-conventional-changelog --save --save-exact
git cz
```

3.3 .gitignore

.gitignore

```
node_modules
.vscode
dist
```

3.4 提交的格式

```
():
```

- 代表某次提交的类型，比如是修复bug还是增加feature
- 表示作用域，比如一个页面或一个组件
- 主题 ，概述本次提交的内容
- 详细的影响内容
- 修复的bug和issue链接

类型 含义 feat 新增feature fix 修复bug docs 仅仅修改了文档，比如README、CHANGELOG、CONTRIBUTE等 style 仅仅修改了空格、格式缩进、偏好等信息，不改变代码逻辑 refactor 代码重构，没有新增功能或修复bug perf 优化相关，提升了性能和体验 test 测试用例，包括单元测试和集成测试 chore 改变构建流程，或者添加了依赖库和工具 revert 回滚到上一个版本 ci CI 配置，脚本文件等更新

3.4 husky

- `validate-commit-msg`可以来检查我们的commit规范
- husky可以把 `validate-commit-msg`作为一个 `githook`来验证提交消息

```
cnpm i husky  validate-commit-msg --save-dev
```

```
"husky": {
  "hooks": {
    "commit-msg": "validate-commit-msg"
  }
}
```

3.5 生成CHANGELOG.md

- `conventional-changelog-cli` 默认推荐的 commit 标准是来自angular项目
- 参数 `-i CHANGELOG.md`表示从 `CHANGELOG.md` 读取 `changelog`

- 参数 -s 表示读写 `CHANGELOG.md` 为同一文件
- 参数 -r 表示生成 changelog 所需要使用的 release 版本数量，默认为1，全部则是0

```
cnpm i conventional-changelog-cli -D
```

```
"scripts": {
    "changelogs": "conventional-changelog -p angular -i CHANGELOG.md -s -r 0"
}
```

4. 支持Typescript

- 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

5. 支持React

5.1 安装

```
cnpm i react react-dom @types/react @types/react-dom react-router-dom @types/react-router-dom   -S
cnpm i webpack webpack-cli webpack-dev-server html-webpack-plugin hoist-non-react-statics -D
cnpm i typescript ts-loader source-map-loader -D
cnpm i redux react-redux @types/react-redux redux-thunk  redux-logger @types/redux-logger -S
cnpm i connected-react-router -S
```

包名 作用 react @types/react react核心库 react-dom @types/react-dom react操作DOM库 react-router-dom @types/react-router-dom react路由库 webpack webpack核心库 webpack-cli webpack命令行文件 webpack-dev-server webpack开发服务器 html-webpack-plugin webpack用于生成html的插件 redux 全局状态管理库 react-redux @types/react-redux 连接react和redux的库 redux-thunk 可以让store派发一个函数的中间件 redux-logger @types/redux-logger 可以在状态改变前后打印状态的中间件 typescript JavaScript语言扩展 ts-loader 可以让Webpack使用TypeScript的标准配置文件 `tsconfig.json`

编译TypeScript代码 source-map-loader 使用任意来自Typescript的sourcemap输出，以此通知webpack何时生成自己的sourcemaps,这让你在调试最终生成的文件时就好像在调试TypeScript源码一样

- ts-loader可以让Webpack使用TypeScript的标准配置文件tsconfig.json编译TypeScript代码。
- source-map-loader使用任意来自Typescript的sourcemap输出，以此通知webpack何时生成自己的sourcemaps,这让你在调试最终生成的文件时就好像在调试TypeScript源码一样。

5.2 tsconfig.json

```
{
  "compilerOptions": {
    "outDir": "./dist",
    "sourceMap": true,
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "module": "commonjs",
    "target": "es5",
    "jsx": "react",
    "baseUrl": ".",
    "paths": {
      "@/*": [
        "src/*"
      ]
    }
  },
  "include": [
    "./src/**/*"
  ]
}
```

包名 作用 outDir 指定输出目录 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 JavaScript 文件，但是它也会报告一个错误 module 代码规范 target 转换成es5 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 include 需要编译的目录

5.3 webpack.config.js

webpack.config.js

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

  module: {
    rules: [
      {
        test: /\.tsx?$/,
        loader: "ts-loader"
      }
    ],
  },

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

5.3 src\index.tsx

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
let root = document.getElementById('root');

let props = { className: 'title' };
let element= React.createElement('div', props, 'hello');
ReactDOM.render(element, root);
```

5.4 src\index.html

src\index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>typescripttitle>
head>
<body>
    <div id="root">div>
body>
html>
```

5.5 package.json

```
{
  "scripts": {
+    "start": "webpack serve --config webpack.config.js",
+    "build": "webpack --config webpack.config.js"
    "eslint": "eslint src --ext .ts",
    "eslint:fix": "eslint src --ext .ts --fix",
    "changelogs": "conventional-changelog -p angular -i CHANGELOG.md -s -r 0",
    "test": "mocha --require ts-node/register test/**/*"
  }
}
```

6. 代码规范

- 规范的代码可以促进团队合作
- 规范的代码可以降低维护成本
- 规范的代码有助于 code review(代码审查)

6.1 常见的代码规范文档

- airbnb中文版 (https://github.com/lin-123/javascript)
- standard中文版 (https://github.com/standard/standard/blob/master/docs/README-zhcn.md)
- 百度前端编码规范 (https://github.com/ecomfe/spec)
- styleguide (https://github.com/fex-team/styleguide/blob/master/css.md)
- CSS编码规范 (https://github.com/ecomfe/spec/blob/master/css-style-guide.md)

6.2 代码检查

- Eslint (https://eslint.org) 是一款插件化的 JavaScript 静态代码检查工具，ESLint 通过规则来描述具体的检查行为

6.2.1 模块安装

```
cnpm i eslint typescript @typescript-eslint/parser @typescript-eslint/eslint-plugin --save-dev
```

6.2.2 eslintrc配置文件

- 英文rules (https://eslint.org/docs/rules/)
- 中文rules (https://cn.eslint.org/docs/rules/)
- 需要添加 parserOptions以支持模块化的写法

.eslintrc.js

```
module.exports = {
    "parser":"@typescript-eslint/parser",
    "plugins":["@typescript-eslint"],
    "rules":{
        "no-var":"error",
        "no-extra-semi":"error",
        "@typescript-eslint/indent":["error",2]
    },
    "parserOptions": {
        "ecmaVersion": 6,
        "sourceType": "module",
        "ecmaFeatures": {
          "modules": true
        }
    }
}
```

6.2.3 代码检查

package.json

```
"scripts": {
    "start": "webpack",
    "build": "tsc",
+    "lint": "eslint src --ext .tsx",
+    "lint:fix": "eslint src --ext .tsx --fix"
  }
```

src/1.ts

```
var name2 = 'zhufeng';;;
if(true){
    let a = 10;
}
```

执行命令

```
npm run eslint
1:1   error  Unexpected var, use let or const instead      no-var
1:23  error  Unnecessary semicolon                         no-extra-semi
1:24  error  Unnecessary semicolon                         no-extra-semi
3:1   error  Expected indentation of 2 spaces but found 4  @typescript-eslint/indent
```

6.2.4 配置自动修复

- 安装vscode的eslint (https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)插件
- 配置自动修复参数

.vscode\settings.json

```
{
    "eslint.validate": [
        "javascript",
        "javascriptreact",
        "typescript",
        "typescriptreact"
    ],
    "editor.codeActionsOnSave": {
        "source.fixAll.eslint": true
    }
  }
```

7.单元测试

7.1 安装配置

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

7.2 src\sum.tsx

src\sum.tsx

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

7.3 tests\sum.spec.tsx

tests\sum.spec.tsx

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

7.4 package.json

package.json

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

8. 持续集成

- Travis CI (https://travis-ci.com) 提供的是持续集成服务（Continuous Integration，简称 CI）。它绑定 Github 上面的项目，只要有新的代码，就会自动抓取。然后，提供一个运行环境，执行测试，完成构建，还能部署到服务器
- 持续集成指的是只要代码有变更，就自动运行构建和测试，反馈运行结果。确保符合预期以后，再将新代码集成到主干
- 持续集成的好处在于，每次代码的小幅变更，就能看到运行结果，从而不断累积小的变更，而不是在开发周期结束时，一下子合并一大块代码

8.1 登录并创建项目

- Travis CI (https://travis-ci.com) 只支持 Github,所以你要拥有 GitHub帐号
- 该帐号下面有一个项目,面有可运行的代码,还包含构建或测试脚本
- 你需要激活了一个仓库， Travis 会监听这个仓库的所有变化

8.2 .travis.yml

- Travis 要求项目的根目录下面，必须有一个 .travis.yml文件。这是配置文件，指定了 Travis 的行为
- 该文件必须保存在 Github 仓库里面，一旦代码仓库有新的 Commit，Travis 就会去找这个文件，执行里面的命令
- 这个文件采用 YAML 格式。下面是一个最简单的 Node 项目的 .travis.yml文件
    - language 字段指定了默认运行环境,所有的语言在此 (https://docs.travis-ci.com/user/languages)

```
language: node_js
node_js:
  - "11"
install: npm install
script:  npm test
```

8.3 实战

8.3.1 生成项目并上传github

```
npx create-react-app zhufeng-typescript-development
```

8.3.2 同步仓库

- 登录travis-ci.com (https://travis-ci.com/)选择同步仓库

8.3.3 设置仓库环境变量

变量名 含义 GH_TOKEN 用户生成的令牌 GH_REF 仓库地址 github.com/zhufengnodejs/zhufeng_typescript_development.git

8.3.4 Github生成访问令牌 (即添加授权)

- 访问令牌的作用就是授权仓库操作权限
- https://github.com/settings/tokens (https://github.com/settings/tokens)
- Github>Developer settings>Personal access tokens> Generate new token > Generate token> Copy Token

8.3.5 .travis.yml

```
language: node_js
node_js:
    - '11'
install:
  - npm install
script:
  - hexo g
after_script:
  - cd ./public
  - git init
  - git config user.name "${USERNAME}"
  - git config user.email "${UESREMAIL}"
  - git add -A
  - git commit -m "Update documents"
  - git push --force  "https://${GH_TOKEN}@${GH_REF}" "master:${GH_BRANCH}"
branches:
  only:
    - master
```

9.React元素

9.1 原生组件

src\index.tsx

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

src\typings.tsx

```
export interface DOMAttributes {
  children?: ReactNode;
}
export interface HTMLAttributes extends DOMAttributes {
  className?: string;
}

export interface ReactElement {
  type: T;
  props: P;
}
export interface DOMElement extends ReactElement{}
export interface ReactHTML { div:  HTMLDivElement }
export interface DetailedReactHTMLElement extends DOMElement{
  type: keyof ReactHTML;
}

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

export declare function createElement<P extends {}>(
  type: string,
  props?: P,
  ...children: ReactNode[]): ReactElement;
```

9.2 函数组件

src\index.tsx

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

src\typings.tsx

```
export interface DOMAttributes {
  children?: ReactNode;
}
export interface HTMLAttributes extends DOMAttributes {
  className?: string;
}
+export type JSXElementConstructor = ((props: P) => ReactElement | null)
+export interface ReactElement = string> {
  type: T;
  props: P;
}
export interface DOMElement extends ReactElement{}
export interface ReactHTML { div:  HTMLDivElement }
export interface DetailedReactHTMLElement extends DOMElement{
  type: keyof ReactHTML;
}

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

+type PropsWithChildren = P & { children?: ReactNode };
+interface FunctionComponent {
+  (props: PropsWithChildren): ReactElement | null;
+}
+interface FunctionComponentElement extends ReactElement> {}
+export declare function createElement(
+  type: FunctionComponent,
+  props?: P,
+  ...children: ReactNode[]): FunctionComponentElement;
export declare function createElement(
  type: string,
  props?: P,
  ...children: ReactNode[]): ReactElement;
```

9.3 类组件

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
let root: HTMLElement | null = document.getElementById('root');
interface Props {
  className: string
}
interface State {
  count:number
}
class Welcome extends React.Component<Props, State> {
  state = { count: 0 }
  render():React.DetailedReactHTMLElement {
    return React.createElement('div', this.props, this.state.count);
  }
}
let props: Props = { className: 'title' };
let element = (
  React.createElement(Welcome, props)
)
ReactDOM.render(element, root);
```

src\typings.tsx

```
export interface DOMAttributes {
  children?: ReactNode;
}
export interface HTMLAttributes extends DOMAttributes {
  className?: string;
}
export type JSXElementConstructor =
| ((props: P) => ReactElement | null)
+| (new (props: P) => Component);

export interface ReactElement = string> {
  type: T;
  props: P;
}
export interface DOMElement extends ReactElement{}
export interface ReactHTML { div:  HTMLDivElement }
export interface DetailedReactHTMLElement extends DOMElement{
  type: keyof ReactHTML;
}

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

type PropsWithChildren = P & { children?: ReactNode };
interface FunctionComponent {
  (props: PropsWithChildren): ReactElement | null;
}
interface FunctionComponentElement extends ReactElement> {}

+type ComponentState = any;
+declare class Component {
+  setState(state: any): void;
+  render(): ReactNode;
+}
+interface ComponentClass {
+  new(props: P): Component;
+}
+interface ComponentElement extends ReactElement> {}
+export declare function createElement(
+  type:  ComponentClass,
+  props?: P,
+  ...children: ReactNode[]): ComponentElement;
export declare function createElement(
  type: FunctionComponent,
  props?: P,
  ...children: ReactNode[]): FunctionComponentElement;
export declare function createElement(
  type: string,
  props?: P,
  ...children: ReactNode[]): ReactElement;
```

10. 创建组件

10.1 Counter.tsx

src\components\Counter.tsx

```
import * as React from 'react';
export interface Props {
    number: number
}
export default class Counter extends React.Component<Props>{
    render() {
        const { number } = this.props;
        return (
            <div>
                <p>{number}p>
            div>
        )
    }
}
```

10.2 types.tsx

src\components\Todos\types.tsx

```
export type Todo = {
    id:number;
    text:string
}
```

10.3 TodoItem.tsx

src\components\Todos\TodoItem.tsx

```
import * as React from "react";
import { Todo } from './types';
const todoItemStyle: React.CSSProperties = {
  color: "red",
  backgroundColor: "green",
};
interface Props {
  todo: Todo;
}
const TodoItem: React.FC = (props: Props) => (
  <li style={todoItemStyle}>{props.todo.text}li>
);
TodoItem.defaultProps;

export default TodoItem;
```

10.4 TodoInput.tsx

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from './types';
interface Props {
  addTodo:(todo:Todo)=>void
}
interface State {
  text:string
}
let id=0;
export default class TodoInput extends React.Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.state = {
      text: "",
    };
  }
  public render() {
    const { text } = this.state;
    const { handleChange, handleSubmit } = this;

    return (
      <form onSubmit={handleSubmit}>
        <input type="text" value={text} onChange={this.handleChange} />
        <button type="submit">添加button>
      form>
    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
```

10.5 Todos\index.tsx

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from './types';
const ulStyle: React.CSSProperties = {
  width: "100px"
};
export interface Props {
  title:string
}
export interface State {
  todos: Todo[]
}
export default class Todos extends React.Component<Props,State>{
  state = {todos:new Array()};
  addTodo = (todo:Todo) =>{
    this.setState({todos:[...this.state.todos,todo]});
  }
  render() {
    return (
      <div>
        <h1>{this.props.title}h1>
        <TodoInput addTodo={this.addTodo}/>
        <ul style={ulStyle}>
          {
            this.state.todos.map(todo=><TodoItem key={todo.id} todo={todo}/>)
          }
        ul>
      div>
    )
  }
}
```

10.6 src\index.tsx

src\index.tsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import Counter from "./components/Counter";
import Todos from "./components/Todos";
ReactDOM.render(<><Counter number={100} /><Todos title="待办事项"/></>, document.getElementById("root"));
```

11. 默认属性

11.1 TodoInput.tsx

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from './types';

+let defaultProps = {
+  setting:{
+    maxLength: 6,
+    placeholder: '请输入待办事项'
+  }
+}
+export type DefaultProps = Partial;
+interface OwnProps {
+  addTodo:(todo:Todo)=>void
+}
+type Props = OwnProps & DefaultProps;
interface State {
  text:string
}
let id=0;
export default class TodoInput extends React.Component {
+  static defaultProps: Required = defaultProps;
  constructor(props: Props) {
    super(props);
    this.state = {
      text: ""
    };
  }
  public render() {
    const { text } = this.state;
+   const { setting } = this.props as (Props & Required);
    const { handleChange, handleSubmit } = this;
    return (

+
        value={text} onChange={handleChange} />
        添加

    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
```

12. 高阶组件

12.1 安装

```
cnpm i hoist-non-react-statics -S
```

12.2 src\utils.tsx

src\utils.tsx

```
import * as React from 'react';
import  hoistNonReactStatics from 'hoist-non-react-statics';
export const defaultProps = {
  setting: {
    maxLength: 6,
    placeholder: '请输入待办事项'
  }
}

export type DefaultProps = Partial<typeof defaultProps>;
export const withDefaultInputProps =  (OldComponent: React.ComponentType) => {
  type OwnProps = Omit;
  class NewComponent extends React.Component {
     public render() {
       let props = { ...defaultProps, ...this.props} as Props;
       return (

       );
     }
  }
  return hoistNonReactStatics(NewComponent, OldComponent);
};
```

12.3 TodoInput.tsx

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
import { Todo } from './types';
+import {withDefaultInputProps,DefaultProps } from '@/utils';
interface OwnProps {
  addTodo:(todo:Todo)=>void
}
type Props = OwnProps & DefaultProps;
interface State {
  text:string
}
let id=0;
class TodoInput extends React.Component {
  constructor(props: Props) {
    super(props);
    this.state = {
      text: ""
    };
  }
  public render() {
    const { text } = this.state;
    const { setting } = this.props as (Props & Required);
    const { handleChange, handleSubmit } = this;
    return (

        添加

    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
+ export default withDefaultInputProps(TodoInput);
```

12.4 src\hoistNonReactStatics.tsx

src\hoistNonReactStatics.tsx

```
import * as React from "react";

function hoistNonReactStatics<T extends React.ComponentType<any>,S extends React.ComponentType<any>
>(
  TargetComponent: T,
  SourceComponent: S
): T & S;
function hoistNonReactStatics<T extends React.ComponentType<any>, S extends React.ComponentType<any>>
(targetComponent:T , sourceComponent: S):T&S {
  let keys = Object.getOwnPropertyNames(sourceComponent);
  for (let i = 0; i < keys.length; ++i) {
    const key = keys[i];
    const descriptor = Object.getOwnPropertyDescriptor(sourceComponent, key);
    Object.defineProperty(targetComponent, key, descriptor!);
  }
  return targetComponent as T&S;
}
interface Props{}
interface State{}
class Old extends React.Component<Props, State> {
  static age1: number = 10;
}

class New extends React.Component<Props, State> {
  static age2: number = 10;
}
let c = hoistNonReactStatics<typeof New, typeof  Old>(New, Old);
c.age1;
c.age2;
```

13. 集成redux

13.1 models\todos.tsx

src\models\todos.tsx

```
export type Todo = {
    id: number;
    text: string
}
```

13.2 models\index.tsx

src\models\index.tsx

```
import { Todo} from './todos';
export {
    Todo
}
```

13.3 action-types.tsx

src\store\action-types.tsx

```
export const ADD = 'ADD';
export const MINUS = 'MINUS';

export const ADD_TODO = 'ADD_TODO';
```

13.4 reducers\counter.tsx

src\store\reducers\counter.tsx

```
import * as types from '../action-types';
import { Action } from '../actions';
export interface CounterState{
    number:number;
}
let initialState: CounterState = { number: 0 };
export default function (state: CounterState = initialState, action: Action): CounterState {
    switch (action.type) {
        case types.ADD:
            return { ...state, number: state.number + 1 };
        case types.MINUS:
            return { ...state, number: state.number - 1 };
        default:
            return state;
    }
}
```

13.5 reducers\todos.tsx

src\store\reducers\todos.tsx

```
import { ADD_TODO} from '../action-types';
import { Action } from '../actions';
import { Todo} from '../../models';
export interface TodosState {
    list: Array;
}
let initialState: TodosState = { list: new Array()};
export default function (state: TodosState = initialState, action: Action): TodosState {
    switch (action.type) {
        case ADD_TODO:
            return { list: [...state.list,action.payload]};
        default:
            return state;
    }
}
```

13.6 reducers\index.tsx

src\store\reducers\index.tsx

```
import counter, { CounterState} from './counter';
import todos,{TodosState} from './todos';
import { combineReducers } from 'redux';
let reducers = {
    counter,
    todos
};
type ReducersType = typeof reducers;
type CombinedState = {
    [key in keyof ReducersType]: ReturnType
}
export { CombinedState, CounterState, TodosState}

let combinedReducer = combineReducers(reducers);
export default combinedReducer;
```

13.7 actions\counter.tsx

src\store\actions\counter.tsx

```
import { ADD, MINUS } from '../action-types';
export interface AddAction {
    type: typeof ADD
}
export interface MinusAction {
    type: typeof MINUS
}

export function add(): AddAction {
    return { type: ADD };
}
export function minus(): MinusAction {
    return { type: MINUS };
}
```

13.8 actions\todos.tsx

src\store\actions\todos.tsx

```
import { ADD_TODO } from '../action-types';
import { Todo} from '@/models';
export interface AddTodoAction {
    type: typeof ADD_TODO,
    payload:Todo
}

export function addTodo(todo:Todo): AddTodoAction {
    return { type: ADD_TODO,payload:todo };
}
```

13.9 actions\index.tsx

src\store\actions\index.tsx

```
import { AddAction,MinusAction} from './counter';
import { AddTodoAction } from './todos';
export type Action = AddAction | MinusAction | AddTodoAction;
```

13.10 components\Counter.tsx

src\components\Counter.tsx

```
import * as React from 'react';
+import { connect } from 'react-redux';
+import { CombinedState, CounterState } from '../store/reducers';
+import * as actions from '@/store/actions/counter';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+type Props = StateProps & DispatchProps;

class Counter extends React.Component{
    render() {
        const { number } = this.props;
        return (

                {number}
                this.props.add()}>+

        )
    }
}

+let mapStateToProps = function (state: CombinedState): CounterState {
+    return state.counter;
+}

+export default connect(mapStateToProps, actions)(Counter);
```

13.11 TodoItem.tsx

src\components\Todos\TodoItem.tsx

```
import * as React from "react";
+import { Todo } from '@/models';
const todoItemStyle: React.CSSProperties = {
  color: "red",
  backgroundColor: "green",
};
interface Props {
  todo: Todo;
}
const TodoItem: React.FC = (props: Props) => (
  {props.todo.text}
);
TodoItem.defaultProps;

export default TodoItem;
```

13.12 TodoInput.tsx

src\components\Todos\TodoInput.tsx

```
import * as React from "react";
+import { Todo } from '@/models';
import { withDefaultInputProps,DefaultProps } from '@/utils';
interface OwnProps {
  addTodo:(todo:Todo)=>void
}
type Props = OwnProps & DefaultProps;
interface State {
  text:string
}
let id=0;

class TodoInput extends React.Component {
  static age:number = 10;
  constructor(props: Props) {
    super(props);
    this.state = {
      text: ""
    };
  }
  public render() {
    const { text } = this.state;
    const { setting } = this.props as Props & Required;
    const { handleChange, handleSubmit } = this;
    return (

        添加

    );
  }

  handleChange = (e: React.ChangeEvent) => {
    this.setState({ text: e.target.value });
  }

  handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    let text = this.state.text.trim();
    if (!text) {
      return;
    }
    this.props.addTodo({id:id++,text});
    this.setState({ text: "" });
  }
}
export default withDefaultInputProps(TodoInput);
```

13.13 Todos\index.tsx

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
+import { Todo} from '@/models';
+import { CombinedState,TodosState  } from '@/store/reducers';
+import * as actions from '@/store/actions/todos';
+import {connect} from 'react-redux';
+const ulStyle: React.CSSProperties = {
+  width: "100px"
+};
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+type Props = StateProps & DispatchProps;
export interface State {
  todos: Todo[]
}
class Todos extends React.Component{
  addTodo = (todo:Todo) =>{
    this.props.addTodo(todo);
  }
  render() {
    return (

          {
            this.props.list.map(todo=>)
          }

    )
  }
}
+let mapStateToProps = function (state: CombinedState): TodosState {
+  return state.todos;
+}
+export default connect(mapStateToProps, actions)(Todos);
```

13.14 store\index.tsx

src\store\index.tsx

```
import { createStore } from 'redux'
import combinedReducer from './reducers';
let store = createStore(combinedReducer);
export default store;
```

13.15 src\index.tsx

src\index.tsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import Counter from "./components/Counter";
import Todos from "./components/Todos";
+import { Provider } from 'react-redux';
+import store from './store';
+ReactDOM.render(
+
+
+
+
+
+
+, document.getElementById("root"));
```

14. 使用路由

14.1 src\index.tsx

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import Counter from './components/Counter';
import Todos from './components/Todos';
import { Provider } from 'react-redux';
import store from './store';
+import { BrowserRouter as Router, Route, Link } from 'react-router-dom';
ReactDOM.render((

+
+                    counter
+                    todos
+
+
+

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

14.2 Counter.tsx

src\components\Counter.tsx

```
import * as React from 'react';
import { connect } from 'react-redux';
import { CombinedState, CounterState } from '../store/reducers';
+import * as actions from '@/store/actions/counter';
+import { RouteComponentProps } from 'react-router-dom';
+import { StaticContext} from 'react-router';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+interface Params { name:string}
+interface LocationState { }
+type Props = StateProps & DispatchProps & RouteComponentProps;

class Counter extends React.Component{
    render() {
+        const { name} = this.props.match.params;
        const { number } = this.props;
        return (

+                名称:{name}
                {number}
                this.props.add()}>+
                 this.props.history.push({ pathname: "/todos", state: { todoName: 'todoName' } })}>/todos

        )
    }
}

let mapStateToProps = function (state: CombinedState): CounterState {
    return state.counter;
}

export default connect(mapStateToProps, actions)(Counter);
```

14.3 Todos\index.tsx

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from '@/models';
+import { CombinedState, CounterState, TodosState } from '@/store/reducers';
+import { RouteComponentProps } from 'react-router-dom';
+import { StaticContext } from 'react-router';
+import * as actions from '@/store/actions/todos';
+import { connect } from 'react-redux';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+const ulStyle: React.CSSProperties = {
+    width: "100px"
+};
+export interface State {
+    todos: Todo[]
+}
+interface Params {}
+interface LocationState { name:string }
+type Props = StateProps & DispatchProps & RouteComponentProps;
class Todos extends React.Component{
    addTodo = (todo:Todo) =>{
       this.props.addTodo(todo);
    }
    render() {
        return (

+               名称:{this.props.location.state.name}

                    {
                        this.props.list.map(todo=>)
                    }

        )
    }
}
let mapStateToProps = function (state: CombinedState): TodosState {
    return state.todos;
}
export default connect(mapStateToProps, actions)(Todos);
```

15. connected-react-router

15.1 src\history.tsx

src\history.tsx

```
import { createBrowserHistory } from 'history'
const history = createBrowserHistory()
export default history;
```

15.2 src\index.tsx

src\index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import Counter from './components/Counter';
import Todos from './components/Todos';
import { Provider } from 'react-redux';
import store from './store';
import {Route, Link } from 'react-router-dom';
+import history from '@/history';
+import {ConnectedRouter} from 'connected-react-router';
ReactDOM.render((

+

                    counter
                    todos

+

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

15.3 store\index.tsx

src\store\index.tsx

```
import { createStore, applyMiddleware } from 'redux'
+import combinedReducer from './reducers';
+import { routerMiddleware } from 'connected-react-router';
+import history from '@/history';

+let store =applyMiddleware(routerMiddleware(history))(createStore)(combinedReducer);
export default store;
```

15.4 reducers\index.tsx

src\store\reducers\index.tsx

```
import counter, { CounterState} from './counter';
import todos,{TodosState} from './todos';
import { combineReducers } from 'redux';
+import history from '@/history';
+import { connectRouter } from 'connected-react-router'
let reducers = {
    counter,
    todos,
+    router: connectRouter(history)
};
type ReducersType = typeof reducers;
type CombinedState = {
    [key in keyof ReducersType]: ReturnType
}
export { CombinedState, CounterState, TodosState}

let combinedReducer = combineReducers(reducers);
export default combinedReducer;
```

15.5 actions\counter.tsx

src\store\actions\counter.tsx

```
import { ADD, MINUS } from '../action-types';
+import { push, CallHistoryMethodAction } from 'connected-react-router';
+import { LocationDescriptorObject,Path} from 'history';
+import { TodosLocationState} from '@/components/Todos';
export interface AddAction {
    type: typeof ADD
}
export interface MinusAction {
    type: typeof MINUS
}
export function add(): AddAction {
    return { type: ADD };
}
export function minus(): MinusAction {
    return { type: MINUS };
}
+export function go(path: LocationDescriptorObject): CallHistoryMethodAction {
+    return push(path);
+}
```

15.6 actions\index.tsx

src\store\actions\index.tsx

```
import { AddAction,MinusAction} from './counter';
import { AddTodoAction } from './todos';
+import { CallHistoryMethodAction } from 'connected-react-router';
+export type Action = AddAction | MinusAction | AddTodoAction | CallHistoryMethodAction;
```

15.7 Counter.tsx

src\components\Counter.tsx

```
import * as React from 'react';
import { connect } from 'react-redux';
import { CombinedState, CounterState } from '../store/reducers';
import * as actions from '@/store/actions/counter';
import { RouteComponentProps } from 'react-router-dom';
+import { StaticContext} from 'react-router';
+import { LocationDescriptorObject} from 'history';
+import { TodosLocationState } from '@/components/Todos';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+interface Params { name:string}
+interface CounterLocationState { }
+type Props = StateProps & DispatchProps & RouteComponentProps;
class Counter extends React.Component{
    render() {
+       const { name} = this.props.match.params;
        const { number } = this.props;
+        let path: LocationDescriptorObject = { pathname: "/todos", state: { name: 'todoName' } };
        return (

+                名称:{name}
                {number}
                this.props.add()}>+
                 this.props.go(path)}>/todos

        )
    }
}

let mapStateToProps = function (state: CombinedState): CounterState {
    return state.counter;
}

export default connect(mapStateToProps, actions)(Counter);
```

15.8 Todos\index.tsx

src\components\Todos\index.tsx

```
import * as React from 'react';
import TodoInput from './TodoInput';
import TodoItem from './TodoItem';
import { Todo} from '@/models';
import { CombinedState, CounterState, TodosState } from '@/store/reducers';
import { RouteComponentProps } from 'react-router-dom';
import { StaticContext } from 'react-router';
import * as actions from '@/store/actions/todos';
import { connect } from 'react-redux';
type StateProps = ReturnType;
type DispatchProps = typeof actions;
const ulStyle: React.CSSProperties = {
    width: "100px"
};
export interface State {
    todos: Todo[]
}
+interface Params {}
+export interface TodosLocationState { name: string }
+type Props = StateProps & DispatchProps & RouteComponentProps;
class Todos extends React.Component{
    addTodo = (todo:Todo) =>{
       this.props.addTodo(todo);
    }
    render() {
        return (

+                名称:{this.props.location.state.name}

                    {
                        this.props.list.map(todo=>)
                    }

        )
    }
}
let mapStateToProps = function (state: CombinedState): TodosState {
    return state.todos;
}
export default connect(mapStateToProps, actions)(Todos);
```

16. redux-thunk

16.1 redux-thunk\index.tsx

src\redux-thunk\index.tsx

```
import {
    Middleware,
    Dispatch,
    MiddlewareAPI,
    Action
} from "redux";

export type ThunkAction = (
    dispatch: ThunkDispatch,
    getState: () => S
) => void;
export interface ThunkDispatch {
    (action: T): T;
    (asyncAction: ThunkAction): R;
}
type ThunkDispatched = ThunkDispatch;
const thunk: Middleware<
    ThunkDispatched,
    {},
    ThunkDispatched
> = (api: MiddlewareAPI) => (
    next: Dispatch
) => (action: Function | Action): any => {
    if (typeof action === "function") {
        return action(api.dispatch, api.getState);
    }
    return next(action);
};
export default thunk;
```

16.2 store\index.tsx

src\store\index.tsx

```
import {
    createStore,
    applyMiddleware,
    Action,
    Store,
    AnyAction,
    StoreEnhancer,
    StoreEnhancerStoreCreator
} from "redux";
import combinedReducer,{CombinedState} from './reducers';
import { routerMiddleware } from 'connected-react-router';
import history from '@/history';

import thunk, { ThunkAction, ThunkDispatch } from '@/redux-thunk';
interface Ext {
    dispatch: ThunkDispatchundefined, AnyAction>
}
interface StateExt{}
let storeEnhancer: StoreEnhancer = applyMiddleware(routerMiddleware(history),thunk);

let storeEnhancerStoreCreator: StoreEnhancerStoreCreator = storeEnhancer(createStore);
let store: Store & Ext = storeEnhancerStoreCreator(combinedReducer);
let thunkAction: ThunkAction<void,CombinedState, undefined, AnyAction> = (
    dispatch: ThunkDispatchundefined, AnyAction>,
    getState: () => CombinedState
): void => { }
store.dispatch(thunkAction);
export default store;
```

参考

- commit_message_change_log (http://www.ruanyifeng.com/blog/2016/01/commit_message_change_log.html)