03 创建 React 项目

创建项目

Create React App ✕

```
npx create-react-app react-ts-demo --template typescript
```

> create-react-app 内部使用了 webpack 进行打包！
>
> ✔️ 这里本打算采用 create-react-app 创建 React 项目。不过注意，create-react-app 目前已**停止维护**，使用过程中遇到许多问题，故放弃，转而使用 Vite。
>
> ❌ npm start 时的报错解决 1
>
> - 错误信息
>
>   ```
>   (node:16536) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
>   (Use `node --trace-deprecation ...` to show where the warning was created)
>   (node:16536) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
>   ```
>
> - 解决方式：将 node_modules/react-scripts/config/webpackDevServer.config.js 中的
>
>   ```
>   onBeforeSetupMiddleware(devServer) {
>       // Keep `evalSourceMapMiddleware`// middlewares before `redirectServedPath` otherwise will not have any effect// This lets us fetch source contents from webpack for the error overlay
>       devServer.app.use(evalSourceMapMiddleware(devServer));
>
>       if (fs.existsSync(paths.proxySetup)) {
>       // This registers user provided middleware for proxy reasonsrequire(paths.proxySetup)(devServer.app);
>       }
>   },
>   ```

```
onAfterSetupMiddleware(devServer) {
    // Redirect to `PUBLIC_URL` or `homepage` from `package.json` if url not match
    devServer.app.use(redirectServedPath(paths.publicUrlOrPath));

    // This service worker file is effectively a 'no-op' that will reset any// previous service worker registered for the same host:port combination.// We do this in development to avoid hitting the production cache if// it used the same host and port.// https://github.com/facebook/create-react-app/issues/2272#issuecomment-302832432
    devServer.app.use(noopServiceWorkerMiddleware(paths.publicUrlOrPath));
},
```

替换为

```
setupMiddlewares: (middlewares, devServer) => {
    if (!devServer) {
        throw new Error('webpack-dev-server is not defined')
    }

    if (fs.existsSync(paths.proxySetup)) {
        require(paths.proxySetup)(devServer.app)
    }

    middlewares.push(
        evalSourceMapMiddleware(devServer),
        redirectServedPath(paths.publicUrlOrPath),
        noopServiceWorkerMiddleware(paths.publicUrlOrPath)
    )

    return middlewares;
},
```

❌ npm start 时的报错解决 2

- 错误信息

```
One of your dependencies, babel-preset-react-app, is importing the "@babel/plugin-proposal-private-property-in-object" package without declaring it in its dependencies. This is currently working because "@babel/plugin-proposal-private-property-in-object" is already in your node_modules folder for unrelated reasons, but it may break at any time.
```

```
6
7 babel-preset-react-app is part of the create-react-app project, which
8 is not maintianed anymore. It is thus unlikely that this bug will
9 ever be fixed. Add "@babel/plugin-proposal-private-property-in-object"
  to
10 your devDependencies to work around this error. This will make this
  message
11 go away.
```

- 解决方式：安装相关包

```
1 npm install --save-dev @babel/plugin-proposal-private-property-in-
  object
2 npm install --save-dev @babel/plugin-transform-private-property-in-
  object
```

Vite √

```
1 npm create vite@latest react-demo-vite -- --template react-ts
```

> Vite 使用 ES Module 语法，启动速度更快！

编码规范

- **eslint** 检查编码规范，如变量未定义、定义未使用。
- **prettier** 检查编码风格，如每一行末尾是否加分号。（eslint 也有编码风格的功能，两者可能会有冲突）

eslint

Step 1. 查看 package.json，**确保安装了以下开发依赖**，如果没有则需手动安装。使用上述 Vite 命令会自动安装以下依赖。

```
1 {
2   "devDependencies": {
3     "@typescript-eslint/eslint-plugin": "^7.15.0",
4     "@typescript-eslint/parser": "^7.15.0",
5     "eslint": "^8.57.0",
6   }
```

```
}
```

Step 2. 检查项目目录，**确保项目根文件夹下有 eslint 配置文件 .eslintrc.cjs**，如果没有则需要手动创建。使用上述 Vite 命令会自动创建该文件。

> `plugins` 与 `extends` 配置项的区别：`extends` 提供 eslint 现有规则的预设；`plugins` 提供除预设之外的自定义规则

Step 3. 安装 Vscode 插件 **ESLint**。

Step 4. 查看 package.json，**确保配置了 `lint` 命令**以检查编码规范，如果没有则需手动配置。通过 `npm run lint`，可以检查当前项目的编码规范。使用上述 Vite 命令会自动创建该配置。

```
{
  "scripts": {
    "lint": "eslint . --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
  },
}
```

prettier

Step 1. **安装以下开发依赖包**。

```
npm install prettier eslint-config-prettier eslint-plugin-prettier --save-dev
```

> eslint-config-prettier 禁用所有和 Prettier 产生冲突的规则
>
> eslint-plugin-prettier 将 Prettier 应用到 Eslint，实现 Eslint 提醒。

Step 2. 在 .eslintrc.cjs 文件中**追加以下配置**。

```
// .eslintrc.cjs
module.exports = {
    "extends": [
        'plugin:prettier/recommended',
    ]
}
```

Step 3. 安装 Vscode 插件 **Prettier - Code formatter**。

Step 4. 在 package.json 中**配置 `format` 命令**。通过 `npm run format` 可以格式化当前项目，使其符合编码风格，如单引号替换为双引号，每一行末尾添加分号等。

```
{
  "scripts": {
    "format": "prettier --write 'src/**/*.{js,ts,jsx,tsx}'",
  },
}
```

Step 5. 在项目根文件夹下**创建 .vscode/settings.json 文件，并添加如下配置**，实现保存文件时自动修复 Eslint 错误。

```
{
  "editor.codeActionsOnSave": {
      "source.fixAll.eslint": "explicit"
  }
}
```

Step 6. 在项目根文件夹下**创建 .prettierrc.cjs 配置文件，自定义编码风格**。此时使用 `npm run format` 和保存文件都会生效。

```
// .prettierc.cjs
module.exports = {
  arrowParens: 'avoid',
  bracketSpacing: true,
  endOfLine: 'crlf',
  printWidth: 100,
  proseWrap: 'preserve',
  semi: true,
  singleQuote: true,
  tabWidth: 2,
  useTabs: false,
  trailingComma: 'es5',
  parser: 'typescript',
};
```

提交规范

代码提交

这里将代码提交到 git 仓库 https://coding.net/

```
git remote add origin <仓库地址>
git push -u origin main
```

husky

husky 是一个 git hook 工具，用于**在 `git commit` 之前执行自定义的命令**，如进行代码风格检查等，从而避免提交不规范的代码。

Step 1. 安装。

```
npm install --save-dev husky
```

Step 2. 初始化：以下会在 .husky/ 中创建 pre-commit 脚本，并更新 package.json 中的 prepare 脚本。

```
npx husky init
```

Step 3. 自定义在 `git commit` 前要执行的命令。此时执行 `git commit` 命令前就会预先执行 .husky/pre-commit 中的自定义命令。

```
echo "npm run format" >> .husky/pre-commit
echo "git add ." >> .husky/pre-commit
echo "npm run lint" >> .husky/pre-commit
```

> ❌ 使用 `git commit` 时报错：cannot execute binary file，解决方式见 Husky 9 - cannot execute binary file · Issue #1426 · typicode/husky

commitlint

commitlint 是一个用于**检查 Git 提交消息是否遵循特定格式规范的工具**。一般来说，允许的提交格式为，

```
type(scope?): subject  #scope is optional; multiple scopes are supported
(current delimiter options: "/", "\" and ",")
```

Step 1. 安装。

```
npm install --save-dev @commitlint/{cli,config-conventional}
```

Step 2. 配置。

```
echo "export default { extends: ['@commitlint/config-conventional'] };" > commitlint.config.js
```

Step 3. 添加 husky 提供的 commit-msg 钩子，确保在每次提交时执行 commitlint。

```
echo "npx --no -- commitlint --edit \$1" >> .husky/commit-msg
```

> 此时尝试 `git commit -m "test"` 会失败，但尝试 `git commit -m "chore: commit lint"` 会成功