说明项目中的配置项都是干什么的（为什么需要）？谁在用？怎么用？

.babelrc

干什么用的（为什么需要）

`.babelrc` 是一个配置文件，用于配置 Babel，一个 JavaScript 编译器，用于将 ES6+（ECMAScript 2015）代码转换为向后兼容的 JavaScript 代码，以便在当前和旧版浏览器上运行。`.babelrc` 配置文件可以指定哪些 Babel 插件、预设和其他选项应用于编译过程，从而定制编译的行为。

谁在用

craco.config.js

```
babel: {
    plugins: [
      [
        "import",
        {
          libraryName: "antd",
          // libraryDirectory: "es",
          style: true,
        },
      ],
    ],
  },
```

怎么用

当您在项目中创建 `.babelrc` 文件时，Babel 将在编译您的代码时查找该文件，并使用其中指定的配置。如果您使用的是 Babel 7 或更高版本，您可以在 `.babelrc` 文件中使用 JavaScript 或 JSON 格式。

Babel 会根据 `.babelrc` 文件中的配置来转换您的代码。配置中的选项可以包括预设（presets）和插件（plugins）。预设是一组 Babel 插件的集合，它们共同实现某个特定的转换。例如，`@babel/preset-env` 是一个预设，它包括了转换 ES6+ 代码所需的所有插件。插件则是单个 Babel 转换步骤。

在您的项目中使用 Babel 时，您需要在构建过程中将 Babel 的转换步骤添加到构建工具中。例如，在 Webpack 中，您需要使用 `babel-loader`。`babel-loader` 会自动查找 `.babelrc` 文件，并在构建过程中使用其中指定的配置来转换您的代码。

总之，当您在项目中创建 `.babelrc` 文件并使用 Babel 转换您的代码时，Babel 会使用该文件中指定的配置来进行转换。在某些情况下，您可能需要在构建工具中指定 `.babelrc` 文件的位置或手动指定 Babel 配置选项。

该文件可以在项目的根目录中创建，并使用 JSON 或 JavaScript 编写。下面是一个例子：

```
{
  "presets": [
    "@babel/preset-env"
  ],
  "plugins": [
    "@babel/plugin-proposal-class-properties"
  ]
}
```

在上面的示例中，我们使用了 `@babel/preset-env` 预设来指示 Babel 将 ES6+ 代码转换为向后兼容的代码，以便在当前和旧版浏览器上运行。我们还使用了 `@babel/plugin-proposal-class-properties` 插件来启用类属性提案的转换。您可以根据您的项目需要添加或删除其他插件或预设，以定制编译过程。

.env

.env.local

.eslintignore

.lintstagedrc

commitlint.config.js

craco.config.js

Dockerfile

jest.config.js

jest-env.js

package.json

path.json

干什么用的（为什么需要）

这个配置是针对 TypeScript 项目中的 `tsconfig.json` 文件的。它的作用是为 TypeScript 模块提供路径别名（path aliases），以便在导入模块时可以使用短路径而不是冗长的相对路径。

具体来说，这个配置中的 `"baseUrl": "src"` 指定了项目中源代码的基础路径。这意味着在定义模块路径别名时，可以省略 `"src/"` 前缀，从而使路径更短。例如，如果你的代码文件位于 `src/components/Button.tsx`，你可以使用 `@/components/Button` 而不是 `../../../components/Button` 来导入该模块。

而 `"paths": { "@/*": ["*"] }` 部分则指定了一个别名，其中 `@/*` 是该别名的名称，`*` 是匹配模式，表示任何模块名称都可以使用这个别名。这个别名的作用是将 `@/` 替换为项目的 `baseUrl`，从而允许使用短路径来导入模块。

因此，这个配置使得 TypeScript 项目中的路径更加易读和易于维护，特别是在有大量嵌套目录和相互依赖的模块时。

谁在用

tsconfig.json

怎么用

```
"extends": "./paths.json"
```

tips:这个配置与webpack中的alias的区别是什么

TypeScript 的路径别名配置是在编译时生效的，它告诉编译器在编译过程中如何解析导入语句中的路径别名。这个配置不需要任何其他工具支持，而是直接由 TypeScript 编译器解析，因此可以在任何支持 TypeScript 的环境中使用。

Webpack 的路径别名配置则是在打包时生效的，它告诉 Webpack 在打包过程中如何解析导入语句中的路径别名。这个配置需要在 Webpack 配置文件中进行设置，并且需要使用 Webpack 的别名功能来实现。Webpack 别名功能本质上是一个字符串替换机制，它将别名替换为实际的文件路径，然后在打包过程中使用这个文件路径。

因此，这两个配置的区别在于它们生效的时间和实现方式。TypeScript 的路径别名配置在编译时生效，不需要任何其他工具支持；而 Webpack 的路径别名配置需要在打包时生效，需要使用 Webpack 的别名功能来实现。

playwright.config.ts

tsconfig.json

干什么用的（为什么需要）

`tsconfig.json` 是 TypeScript 编译器的配置文件，用于指定 TypeScript 项目的编译选项和编译方式。当你在命令行中运行 TypeScript 编译器时，它会查找项目根目录下的 `tsconfig.json` 文件，并根据其中的配置进行编译。

谁在用

当 TypeScript 编译器启动时，它会加载 `tsconfig.json` 文件，并读取其中的配置项。这些配置项包括编译选项（如目标 ECMAScript 版本、模块系统、是否启用严格模式等）以及文件列表。根据文件列表，编译器会将每个 TypeScript 文件转换为 JavaScript 文件，并将它们合并为一个或多个输出文件。

在大多数情况下，你可以在项目根目录下创建一个名为 `tsconfig.json` 的文件，并根据自己的需要配置其中的选项。如果你使用的是一些开发工具，如 VS Code 或 Webpack，它们通常会自动检测和使用 `tsconfig.json` 文件，以便在开发过程中提供更好的开发体验和编译结果。

总的来说，`tsconfig.json` 文件是 TypeScript 项目的核心配置文件，它指导 TypeScript 编译器进行编译和类型检查，对于 TypeScript 项目的成功构建非常重要。

怎么用

```
{
  "compilerOptions": {
    "target": "es5",
    "lib": ["dom", "dom.iterable", "esnext"],
    "allowJs": true,
    "types": ["node", "@types/jest"],
    "typeRoots": ["./src/typings"],
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "noEmit": true,
    "jsx": "react-jsx",
    "noImplicitAny": true,
    "strictNullChecks": true,
    "sourceMap": true,
    "removeComments": true,
    "noImplicitReturns": true,
    "alwaysStrict": true,
    "baseUrl": "src",
    "isolatedModules": false,
    "downlevelIteration": true
  },
  "include": [
    "./src/**/*.ts",
    "./src/**/*.tsx",
    "./src/**/*.js",
    "./src/**/*.jsx",
    "./__test__/**/*"
  ],
  "exclude": ["node_modules", "build", "dist", "mock", "public"],
  "extends": "./paths.json"
}
```