**ReactTS-Guidelines**

References:

* https://github.com/pillarstudio/standards/blob/master/reactjs-guidelines.md
* https://dev.to/deepeshk1204/best-practices-of-reactjs-with-typescript-24p4
* https://blog.csdn.net/2301\_79226238/article/details/133745757
* https://github.com/whatthehan/standard/blob/master/React%E7%BC%96%E7%A0%81%E8%A7%84%E8%8C%83.md
* https://jmfe.gitee.io/jm-fe/standard/cli/tslint.html

**基本规则**

* 每个文件只包含一个 React 组件。
* 避免使用any关键字，显式指定类型。

|  |
| --- |
| TypeScript const fetchData = (): Promise<User[]> => {  // Fetch user data from an API };  const handleData = (data: User[] | null) => {  // Handle data }; |

* 项目文件树按照以下方式命名：

|  |
| --- |
| Plain Text pages // 页面 services // 与服务端交互 layouts // 布局组件 components // 通用组件 utils // 公用方法 assets // 静态资源 如logo文件 config // 配置文件 |

* 字符串拼接使用 ES6 语法方式，避免直接相加。

const str = 早安，${name}，祝你开心每一天！;

* TSX 属性始终使用双引号（"），但其他 TS 始终使用单引号(')

|  |
| --- |
| TypeScript <Foo bar="bar" />  <Foo style={{ left: '20px' }} /> |

* 判断数值是否相等使用三等号 ===，判断为 null 或者 undifine 使用双等号 ==。
* 多个判断条件（三个以上）使用 switch case语句。
* 不要使用 Magic Number，使用enum给一个有意义的命名。

|  |
| --- |
| TypeScript enum ErrorCode {  Success = 200,  NotFound = 404, } // ... if (errCode === ErrorCode.Success) {  // do something } |

**命名**

* 文件名和组件名需要相同，都使用大驼峰PascalCase。

|  |
| --- |
| TypeScript // UserProfile.tsx const UserProfile: React.FC<UserProfileProps> = ({ user }) => {  // Component implementation }; |

* 类名和接口名都使用大驼峰PascalCase。

|  |
| --- |
| TypeScript interface UserProfileProps {  user: User; } class FooBarBaz {  //... } |

* 组件属性名、变量名、函数名使用小驼峰camelCase。

|  |
| --- |
| TypeScript // 属性名 <Foo  userName="hello"  phoneNumber={12345678} />  //变量名 fooBarBaz = 1;  //函数名 const getUserData = (): Promise<User> => {  // Fetch user data }; |

* 所有命名尽量避免使用缩写。

**标签**

* TSX标签按以下方式对齐。

|  |
| --- |
| TypeScript // 不好 <Foo superLongParam="bar"  anotherSuperLongParam="baz" />  // 好 <Foo  superLongParam="bar"  anotherSuperLongParam="baz" />  // 如果属性适合一行，则保持在同一行上 <Foo bar="bar" />  // 子项正常缩进 <Foo  superLongParam="bar"  anotherSuperLongParam="baz" >  <Spazz /> </Foo> |

* 对于没有子项的标签，始终自闭合，自闭合标签前留一个空格。

|  |
| --- |
| TypeScript // 不好 <Foo className="stuff"></Foo>  // 好 <Foo className="stuff" /> |

* 如果您的组件具有多行属性，另起一行关闭标签。

|  |
| --- |
| TypeScript // 不好 <Foo  bar="bar"  baz="baz" />  // 好 <Foo  bar="bar"  baz="baz" /> |

**组件**

* 没有内部状态需要维护的组件，都使用函数式组件。
* 使用interface或type定义props和states对象属性的类。

|  |
| --- |
| TypeScript interface MyComponentProps {  name: string;  age: number; }  interface MyComponentState {  isOpen: boolean; }  const MyComponent: React.FC<MyComponentProps> = ({ name, age }) => {  // Component implementation }; |

* 需要同时返回多个组件时，不要在外部增加其他无意义的标签，使用数组标识或空标签（React.Fragment）。

|  |
| --- |
| TypeScript // 空标签 return (  <>  <A />  <B />  <C />  </> );  // 数组标识 return [<A key="a" />, <B key="b" />, <C key="c" />]; |