**前端编程规范**

**（使用React框架）**

1. 基本规则

1.1 一个文件声明一个组件，组件名称和定义该组件的文件名称建议要保持一致，如下所示：

import Footer from './Footer'

1.2 使用 JSX 语法（不要使用 React.createElement 的写法）。

1.3 函数组件和 class 类组件的使用场景： 如果定义的组件不需要 props 和 state ，建议将组件定义成函数组件，否则定义成 class 类组件。

2. 命名

2.1 组件名称：使用大驼峰命名法，如MyComponent

2.2 属性名称：使用小驼峰命名法，且不要使用与HTML属性相同的命名，如onClick

2.3 className的值使用小驼峰命名法，如className="myClass"

2.4 Style样式属性：使用小驼峰命名法，如backgroundColor

2.5 监听方法应使用handle开头，采用小驼峰命名法，如handleClick

2.6 负责绘制界面的函数使用render开头，采用小驼峰命名法，如renderCanvas

2.7 变量名均使用小驼峰命名法

2.8 表示判断的布尔值使用is/has开头，采用小驼峰命名法，如isEmpty

示例：

3. JSX 写法

3.1 标签

3.1.1 当标签没有子元素的时候，始终使用自闭合的标签，且关闭标签前要留一个空格。如下所示：

<Component />

3.1.2 如果标签有多行属性，关闭标签要另起一行。如下所示：

<Component

bar="bar"

baz="baz"

/>

3.2 对齐

3.2.1 如果只有一个属性，则放在同一行：

<Foo bar="bar" />

3.2.2 如果有两个及以上的属性，则所有属性应各自占一行，并缩进：

<Foo

superLongParam="bar"

anotherSuperLongParam="baz"

/>

3.3 引号

JSX的属性使用双引号，其它使用单引号：

<Foo bar="bar" />

<Foo style={{ left: '20px' }} />

4. 样式

4.1 通过使用className属性来使用外部的样式表定义样式。

4.2 使用style行内样式

5. 组件的代码顺序

5.1 static开头的静态属性，如defaultProps

5.2 构造函数constructor

5.3 组件生命周期的钩子函数：

5.3.1 componentWillMount

5.3.2 componentWillReceiveProps

5.3.3 shouldComponentUpdate

5.3.4 componentDidMount

5.3.5 componentDidUpdate

5.3.6 componentWillUnmount

5.4 普通函数

5.5 事件监听方法handle\*

5.6 绘制函数render\*

5.7 render()方法

示例：

class Example extends Component {

// 静态属性

static defaultProps = {}

// 构造函数

constructor(props) {

super(props);

this.state={}

}

// 生命周期钩子函数

componentWillMount() { ... }

componentWillReceiveProps() { ... }

shouldComponentUpdate() { ... }

componentDidMount() { ... }

componentDidUpdate() { ... }

componentWillUnmount() { ... }

//普通函数

calculate = (param1,param2) => { ... }

// 事件函数

handleClick = (e) => { ... }

//绘制函数

renderCanvas= () => { ... }

//render 方法

render() { ... }

}

6. 注释规则

6.1 尽可能减少代码中的注释。可以通过让变量名更语义化，只注释复杂、潜在逻辑，从而减少注释量。同时也提高了可维护性，减少代码和注释间的同步。

6.2 注释使用/\* ... \*/,便于多行注释。

**后端编程规范**

**（使用Golang的Iris框架）**

1. 基本规则
   1. 采用Middleware-Controller-Service-Dao-Repository的分层架构。
   2. 跨域控制需在Middleware层中完成，Controller定义提供给前端的接口，Service层负责处理业务逻辑，Dao层负责提供对数据库的高层抽象与包装对象，Repository层负责底层的数据库操作。各层功能解耦，各司其职，不得混淆。
   3. 采用ORM映射的方法定义数据库对象。
   4. 使用Go Modules进行包管理
2. 命名
   1. 需要export的对象：采用大驼峰命名法。  
      如：NewProject()，const IsTest
   2. 作用域仅限当前文件的对象：采用小驼峰命名法。  
      如：ownedPids
   3. Controller接口命名以New、Get、Modify和Delete开头，分别代表新建、获取、修改与删除对象的操作。  
      如：NewFile，GetUser，ModifyFile，DeleteProject，
   4. 接口接受的前端参数以“接口名+Params ”命名的结构体定义。  
      如：NewFileParams
   5. Response返回的枚举信息命名以“操作对象（User/Project/File/Room）+操作（New/Get/Modify/Delete）+结果（success etc.） ”命名  
      如：FileModifySuccess
3. 接口规范
   1. Response的基本格式为

{

success: true/false, // 简单的成功与否

msg: int, // 通过枚举类型详细定义的操作结果

data : json // 前端请求的数据（如果有）

}