1、前端命名规范

(1)、文件夹统一采用中划线命名，如果是index类型的文件就是小写index.js,如果是组件类型的驼峰大写。

(2)、webRoot下componets文件夹中定义的公共组件，可以根据需要用文件夹包起来，在文件夹中定义index.js和style.less

(3)、pages文件夹下面页面组件的compents文件夹定义的组件也可根据需要用文件夹包起来

(4)、js文件中的常量命名采用大写+_下划线的形式

2、server端模块划分

(1)、对于页面特有的接口，以页面为维度划分

(2)、对于公共的模块，比如车辆、报警类型以实体为维度来划分

3、node端命名规范

(1)、server端的api和service的js文件统一采用驼峰命名

(2)、相同页面或者相同模块的api与service命名尽可能保持一致，方便定位

(3)、定义node端接口的path，以'/api/对应的service文件名/....'命名，方便前端调用接口明确知道调用的哪个service或者api文件的接口

4、复杂函数注释规范

```
    /**
     * Get the x value.
     * @return {number} The x value.
     */
    getX() {
        // ...
    }

    /**
     * Get the y value.
     * @return {number} The y value.
     */
    getY() {
        // ...
    }
```

常见的注释关键字：

@param：描述参数信息 例如：@param name {String} 传入名称

@return： 描述返回信息 例如：@return {Boolean} true:可执行;false:不可执行

@author : 描述此函数的作者的信息 例如：@author 张三 2015/07/21

@version : 描述此函数的版本号 例如：@version 1.0.3

@example : 演示函数的使用 例如： @example setTitle('测试')

更多更详细可参考jsdoc官方文档：https://jsdoc.app/

5、js文件import 引入模块的书写顺序

(1)、首先引入核心模块（React，nodejs相关模块）

(2)、其次引入npm安装的模块

(3)、其次引入自定义模块（绝对路径在前，相对路径在后）

(4)、引入less样式文件

(5)、书写const变量

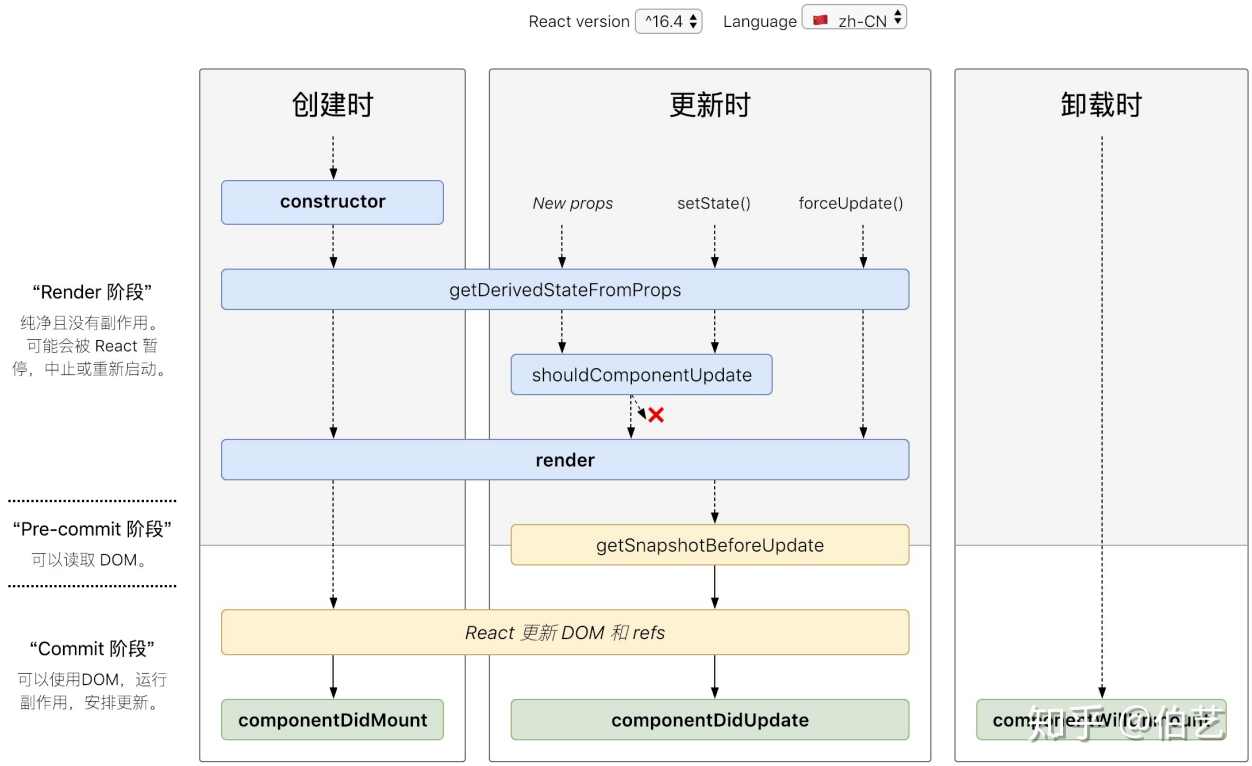
.....

6、代码规范

http://192.168.150.51:3118/article/5d820d17fbbf3b1645b2cdc0

主要来自Airbnb

7、react新生命周期



```
  static getDerivedStateFromProps(nextProps, prevState) {
    4. Updating state based on props
    7. Fetching external data when props change // Clear out previously-loaded
data so we dont render stale stuff
  }
  constructor() {
  1. Initializing state
  }
  componentWillMount() {
    // 1. Initializing state
    // 2. Fetching external data
    // 3. Adding event listeners (or subscriptions)
  }
  componentDidMount() {
  2. Fetching external data
  3. Adding event listeners (or subscriptions)
  }
  componentWillReceiveProps() {
    // 4. Updating state based on props
    // 6. Side effects on props change
    // 7. Fetching external data when props change
  }
  shouldComponentUpdate() {
  }
  componentWillUpdate(nextProps, nextState) {
    // 5. Invoking external callbacks
    // 8. Reading DOM properties before an update

  }
  render() {
  }
  getSnapshotBeforeUpdate(prevProps, prevState) {
  8. Reading DOM properties before an update
  }
  componentDidUpdate(prevProps, prevState, snapshot) {
  5. Invoking external callbacks
  6. Side effects on props change
  }

  componentWillUnmount() {
  }
```

8、异常处理

(1)、对于node端请求的接口数据，做默认值处理，比如

```
let data = await sword.service('businessEvaluation.queryRank')({ ...query });
data = data || { dataList: [], dataCount: 0 };
this.success(data);
```

避免当没有数据后端返回null，但是前端期望是数组导致的语法错误

(2)、与请求相关的（除了model里面的）需要加try catch

使用try catch需要注意，try catch捕获不到语法和异步错误

```
try {
    const name = "张三; // 不能捕获
} catch (error) {
    console.log('异常捕获');
}

try {
    axios.get().then(res => {
        // 里面异常捕获不到
    });
} catch (error) {
    console.log('异常捕获');
}
```

9、小的点

(1)、class文件组件内部函数尽可能定义成箭头函数避免this绑定的问题

(2)、对于不影响页面渲染的数据不要放在state中，避免因为数据改变造成页面不必要的渲染

10、git commit提交规范

每个commit message 包括header,body和footer，各占一行，每行不超过100字符。其中header由type、scope和subject组成。**header必须要写**，header的scope是可选的。

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

Header部分只有一行，包括三个字段：`type`（必需）、`scope`（可选）和`subject`（必需）。

type

用于说明 commit 的类别，只允许使用下面7个标识。

- feat：新功能（feature）

- fix：修补bug
- docs：文档（documentation）
- style： 格式（不影响代码运行的变动）
- refactor：重构（即不是新增功能，也不是修改bug的代码变动）
- test：增加测试
- chore：构建过程或辅助工具的变动

scope（非必填）

scope用于说明 commit 影响的范围，比如数据层、控制层、视图层等等，视项目不同而不同。

subject（非必填）

`subject`是 commit 目的的简短描述