前端规范

以前大多是个人开发，自己随心所欲的开发，自己能看懂就行；但是进入公司，应该遵照公司项目组的规范进行开发，可以协同其他人一起开发，所以规范很重要，并且对代码习惯的养成也很重要。所以以下是一些平常自己没注意或者欠缺的规范

JS规范

1. 用let、const代替var(ES6)
   块级作用域、无变量提升、暂时性死区
2. 使用字面量创建对象、数组

```
// good
const item = {};
// best
const item = Object.create(null);
const items = [];
```

3. 对象方法、属性值简写

```
const atom = {
value: 1,
addValue(value) {
 return atom.value + value;
},
};
const obj = {
lukeSkywalker,
};
```

4. 使用解构存取和使用多属性对象、数组

```
// 函数参数解构赋值
function getFullName({ firstName, lastName }) {
return `${firstName} ${lastName}`;
}
const arr = [1, 2, 3, 4];
const [first, second] = arr;
```

5. 字符串使用单引号

```
const name = 'Capt. Janeway';
```

6. 使用模板字符串代替字符串连接

```
function sayHi(name) {
return `How are you, ${name}?`;
}
```

7. 函数声明代替函数表达式
   函数声明是可命名的，所以他们在调用栈中更容易被识别。此外，函数声明会把整个函数提升（hoisted），而函数表达式只会把函数的引用变量名提升。这条规则使得箭头函数可以取代函数表达式

```
function foo() {
}
```

8. 当你必须使用函数表达式（或传递一个匿名函数）时，使用箭头函数符号
   箭头函数创造了新的一个 this 执行环境（译注：参考 Arrow functions - JavaScript | MDN 和 ES6 arrow functions, syntax and lexical scoping），通常情况下都能满足你的需求，而且这样的写法更为简洁

```
[1, 2, 3].map((x) => {
return x * x;
});
```

9. 如果一个函数适合用一行写出并且只有一个参数

```
// good
[1, 2, 3].map(x => x * x);
// good
[1, 2, 3].reduce((total, n) => {
return total + n;
}, 0);
```

10. 使用模组 (import/export)

```
import { es6 } from './AirbnbStyleGuide';
export default es6;
```

11. 不要使用通配符 import
    这样能确保你只有一个默认 export

```
import AirbnbStyleGuide from './AirbnbStyleGuide';
```

12. 使用高阶函数例如 map() 和 reduce() 替代 for-of

```
let sum = 0;
numbers.forEach((num) => sum += num);
sum === 15;

// best (use the functional force)
const sum = numbers.reduce((total, num) => total + num, 0);
sum === 15;
```

13. 优先使用 === 和 !==
14. 条件表达式例如 if 语句通过抽象方法 ToBoolean 强制计算它们的表达式并且总是遵守下面的规则
    - 对象 被计算为 true
    - Undefined 被计算为 false
    - Null 被计算为 false
    - 布尔值 被计算为 布尔的值
    - 数字 如果是 +0、-0、或 NaN 被计算为 false, 否则为 true
    - 字符串 如果是空字符串 '' 被计算为 false，否则为 true
15. 使用 /** ... */ 作为多行注释。包含描述、指定所有参数和返回值的类型和值
16. 给注释增加 FIXME 或 TODO 的前缀可以帮助其他开发者快速了解这是一个需要复查的问题，或是给需要实现的功能提供一个解决方式。这将有别于常见的注释，因为它们是可操作的。使用 FIXME -- need to figure this out 或者 TODO -- need to implement
17. 使用 2 个空格作为缩进
18. 在花括号前放一个空格
19. 在使用长方法链时进行缩进。使用前面的点 . 强调这是方法调用而不是新语句
20. 如果你的文件只输出一个类，那你的文件名必须和类名完全保持一致

```
import CheckBox from './CheckBox';
```

项目规范

1. 代码中的任何地方不得出现拼音或拼音的缩写，包括URL、文件名等处中也不可出现
2. 不要随便加各种开发包，会导致项目越来越慢，不得不添加时需要征求项目owner意见获得许可
3. 新的模块按照既有方式和流程实现，包括文件目录结构、大小写规范，遵从既定项目规范、模块化、组件化开发原则，不可自己特立独行封装新的结构
4. 用到的基础库现在基本够用，lodash、moment.js、currency.js等，尽量使用这里的函数和方法，可以有效提高程序性能，能用ES6、lodash的地方不要自己实现相应功能
5. 供应链所有模块有关于金额的计算一律采用currency.js计算，且携带前导符（￥）
6. 请严格遵从口碑、蚂蚁金服的设计规范，antd的指引讲述的，颜色、组件使用方式等，https://ant.design/docs/spec/introduce-cn ，尤其是颜色引用，一定要在antd色板上取色使用
7. 原型中不确定或认为不好的设计请协调UED、产品同学意见，不要擅作主张
8. 通用的地方，需要独立为系统公用模块，举例：token限制、权限验证等；对于独立模块的复杂引用路径，请使用webpack的alias或resolve处理
9. 涉及到多出影响的模块，必须通知相应的负责人，请相关负责人进行验证其模块
10. 与服务端交互部分必须严格按照接口文档进行，接口负责人口头要求修改或实现的部分，督促其同步接口文档
11. 严格控制代码质量，代码必须遵从《辰森前端JavaScript编码规范参考》的编码规范，没有约束的不允许上传到GitLab，不得关闭Git push时的规范验证再上传
12. 项目原型图并不是UI设计图，按照原型图的设计实现模块功能，不用严格遵从其UI设计
13. 无用的文件不允许上传到gitlab，比如ide的管理文件等
14. UED的设计在页面实现时与口碑标准有冲突是一律遵从口碑标准，比如单数的字号（口碑规定，字号不可单数），带小数的单位等等
15. 关于物品的数量，数量、库存量、领料量、生产量等，小数位最多保留统一为2位，包括标准单位、辅助单位、生产规格单位