开发规范

前端编码规范

- [TOC]

1 JavaScript 代码风格

1.1 命名

1. 声明变量必须加上关键字 var, 避免使用未声明的变量。
2. 变量名使用**驼峰命名**规则, 例如： gameScene、innerHeight。
3. 变量的命名, 不得使用 js 保留字。

> js保留字列表：abstract, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, var, void, while, with.

4. 自定义对象（类）的命名，每个单词首字母均需要大写，例如：Item。
5. 对象的方法采用驼峰命名方式，必须是动词或者动词短语，例如：obj.updateControl()。

1.2 注释

1. 每个类的定义都要附带一份注释, 描述类的功能和用法。
2. 对于一些特殊的属性，必要时进行注释。
3. 对于每个方法或者函数，必要时进行注释。
4. 除了类的定义、方法函数外，在重要的逻辑、或者较复杂的逻辑处, 增加必要的注释。

1.3 代码结构

1.3.1 缩进

1. 代码需要保持适当的缩进。
2. 缩进使用Tab缩进（1 tab = 4 spaces，可以在编辑器设置），注意不同操作平台的差异。
3. 复合语句(被包含在 “{}”(大括号)的语句序列, 包括函数定义)需要保持缩进。

1.3.2 空格

1. 各种运算符, 包括数值操作符、比较运算符、赋值运算符等前后，保留一个空格。
2. 在对象创建时，属性中的 : 之后必须有空格。

```
//good
var su = new SpriteUtilities(PIXI);
 setup = () => {
```

```
        window.innerHeight = 960;
        window.innerWidth = 1920;
        this.app = new Application({
            width: window.innerWidth,
            height: window.innerHeight,
        });
```

3. import/export 后面的花括号左右各保留一个空格。

4. () 和 [] 内紧贴括号部分不允许有空格。

```
//good
import { Item } from "./Item.js";
succeed = () => {
        this.app.ticker.stop();
        document.body.removeChild(this.app.view);
        location.replace('./index.html');
    };
```

1.3.3 换行

1. 左花括号 { 不要换行。

```
//good
if (e.key === "a") {
            this.sp.obj.playAnimation(this.sp.obj.states.walkLeft);
        }
```

2. 当一条语句一行写不下时, 折行。

3. 折行位置:在运算符号后面换行。在运算符后换行可以减少因为复制粘贴产生的错误被分号掩盖的几率。

1.3.4 语句行

1. 语句必须以分号作为结束符, 不能忽略分号。除了 for, function, if, switch, try, while 结束的花括号后无需分号。

2. 对于复合语句, if, for, while, do, switch, try ... catch 等代码体, 数定义的函数体, 对象的定义等 都需要放在花括号 {} 里面。

    - { 应在行末, 标志代码块的开始。
    - } 应在一行开头, 标志代码块的结束, 同时需和"{"所在行的开始对齐, 以表明是一个完整的复合语句。这 样可极大地提高代码的可阅读性, 控制逻辑能清晰地表现出来。
    - 被包含的代码段应该保持缩进。

- 即使被包含的代码段只有一句, 也应该用{}包含。尽管不用花括号代码也不会错, 但如若需要增加语句的 话, 则较容易因花括号遗漏而引起的编译错误或逻辑错误。

```
hitBorder = (sp1) => {
   let pos = undefined;
   if (sp1.x < 0) {
     pos = "left";
   } else if (sp1.x + sp1.width > window.innerWidth) {
     pos = "right";
   } else if (sp1.y < 0) {
     pos = "top";
   } else if (sp1.y + sp1.height > window.innerHeight) {
     pos = "buttom";
   }
   return pos;
 };
```

2 JavaScript 语言特性

2.1 声明

1. 使用const声明常量

   如果变量不需要重新分配值，全部用 const 声明

   > const 声明后，无法修改引用的值，可避免被重写
   >
   > const 也属于块级作用域
   >
   > const 不会带来变量提升（Hoisting）

2.2 条件语句

1. 使用 === 和 !== 而不是 == 和 !=
2. if 语句，即使只有单行，也要用{}括起来，例如：

```
if (w) {
     this.obj.vy = -this.obj.speedy;
   }
```

3. if/else/while/for 条件表达式必须有小括号, 且自占一行。
4. 尽量避免使用switch。

   > switch 的方式需要逐条 case 判断且匹配的 case，如果漏掉 break，会执行下一条 case （不论是否满足）或 default，直到遇到 break 为止。

2.3 循环

1. 尽量避免 for-in 循环, 只用于 object/hash 的遍历, 数组的遍历使用 for 循环。
2. for-in 循环体中必须用 hasOwnProperty 方法检查成员是否为自身成员, 避免来自原型链上的污染。
3. 避免在 if 和 while 语句的条件部分进行赋值。

3 css开发规范

3.1 命名

3.1.1 组成元素

1. 命名必须由单词、中划线或数字组成。

2. 不允许使用拼音，尤其是缩写的拼音、拼音与英文的混合。

3.1.2 词汇规范

1. 不依据表现形式来命名。
2. 可根据内容来命名。
3. 可根据功能来命名。

3.1.3 缩写规范

1. 保证缩写后还能较为清晰保持原单词所能表述的意思；
2. 使用业界熟知的或者约定俗成的。

3.1.4 命名-前缀规范

前缀	说明	示例
g-	全局通用样式命名，前缀g全称为global，一旦修改将影响全站样式	g-mod
m-	模块命名方式	m-detail
ui-	组件命名方式	ui-selector
js-	所有用于纯交互的命名，不涉及任何样式规则。JSer拥有全部定义权限	js-switch

3.2 id与class

1. 尽量使用class。

3.3 书写格式

1. 选择器与大括号之间保留一个空格。

2. 分号之后保留一个空格。

3. 逗号之后保留一个空格。

4. 所有规则需换行。

5. 多组选择器之间需换行。

```
//good
* {
    padding: 0;
    margin: 0;
}
```

3.4 规则与分号

每条规则结束后都必须加上分号。

3.5 属性书写顺序

1. 遵循先布局后内容的顺序。
2. 私有属性在前标准属性在后

```
.g-box {
      -webkit-box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
      -moz-box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
      -o-box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
      box-shadow: 1px 1px 5px rgba(0, 0, 0, .5);
}
```

3.6 注释规范

保持注释内容与星号之间有一个空格的距离。

普通注释（单行）

```
/* 普通注释 */
```

区块注释

```
/**
 * 模块: m-detail
 * 描述: 酒店详情模块
 * 应用: page detail, info and etc...etc
 */
```

> 有特殊作用的规则一定要有注释说明 应用了高级技巧的地方一定要注释说明

3.7 模块化

1. 每个模块必须是一个独立的样式文件，文件名与模块名一致。

2. 模块样式的选择器必须以模块名开头以作范围约定。

4 HTML开发规范

4.1 文档类型

1. 统一使用HTML5的标准文档类型：`<!DOCTYPE html>`。

> HTML5文档类型具备前后兼容的特质，并且易记易书写

2. 在文档doctype申明之前，不允许加上任何非空字符。

> 任何出现在doctype申明之前的字符都将使得HTML文档进入非标准模式

3. 不允许添加 `<meta>` 标签强制改变文档模式。

> 避免出现不可控的问题

4.2 属性

4.2.1 省略type属性

在调用CSS和JavaScript时，可以将type属性省略不写。

```
<link rel="stylesheet" href="base.css" />
<script src="base.js"></script>
```

4.2.2 省略属性值

非必须属性值可以省略。

```
<input type="text" readonly />
<input type="text" disabled />
```

4.2.3用双引号包裹属性值

所有的标签属性值必须要用双引号包裹，同时也不允许有的用双引号，有的用单引号的情况。

4.3 嵌套

所有元素必须正确嵌套

- 不允许交叉；
- 不允许非法的子元素嵌套。
- 不推荐inline元素包含block元素；

4.4 .标签闭合

所有标签必须闭合

```
<div>balabala...</div>
<link rel="stylesheet" href="*.css" />
```

4.5 按模块添加注释

在每个模块开始和结束的地方添加注释，注释内容左右两边保留和注释符号有1个空格位，在注释内容内不允许再出现中划线"-"，某些 浏览器会报错。

4.6 格式

1. 将每个块元素、列表元素或表格元素都放在新行。
2. inline元素视情况换行，以长度不超过编辑器一屏为宜。

4.7 语义化标签

1. 根据HTML元素的本身用途去使用它们。
2. 禁止使用被废弃的用于表现的标签，如 center, font 等。

   不允许：

   ```
   <p>标题</p>
   ```

   应该：

   ```
   <h1>标题</h1>
   ```

4.8 模块化

1. 每个模块必须有一个模块名。
2. 每个模块的基本组成部分应该一致。
3. 模块的子节点类名需带上模块名（防止模块间嵌套时产生不必要的覆盖）。