前端开发规范文档_V2

1. 文档说明

该文档为前端开发规范文档，适用于公司所有前端开发工作，请开发人员严格遵守文档规范进行开发，使用文档过程中，有任何疑问可以随时提出，经过讨论后更新文档，请不要私自修改文档内容。

2. 项目规范

2.1 项目目录结构规范

2.1.1 常规前端目录结构

项目名

```
|--src
    |-- css
        |-- main.css // 项目主样式文件
    |-- js
        |-- main.js // 项目主js文件
    |-- php // 项目需要用到的php文件
|-- resource // 资源文件
        |--  images // 图片资源文件
        |-- font   // 如果不需要使用字体文件，可以不需要该文件夹;
        |-- icon   //如果不需要使用图标文件，可以不需要该文件夹;
|-- index.html/index.php // 项目入口文件
|-- style.css // 公共样式文件
```

2.1.2 小程序目录结构

```
|-- components // 组件目录
|-- images // 图片资源目录，由于小程序代码大小限制，除小型icon以外，其余图片都需上传到服务器后，通过链接引入，而不是放在项目目录中
|-- lib // 引入的第三方js库
|-- pages
    |-- index // 程序首页文件
|-- utils // 工具类文件
|-- 小程序自带框架全局文件
```

2.2 项目命名规范

- 命名需要明确体现出当前文件的功能（例如：article-detail, article-list），不要使用无意义的命名（~~asd~~）

2.2.1 常规项目命名

- 常规项目命名使用全小写英文命名，多个单词之间使用连字符"-"进行分割。
  例：my-first-project

2.2.2 移动端专题命名规范

- 移动端专题命名使用***拼音***命名(便于编辑同事操作)

2.3 目录和文件命名规范

2.3.1 常规前端目录/文件命名规范

常规目录/文件命名参考项目命名规则，采用全小写英文，多个单词使用连字符"-"分割;

例：my-page.html || my-func.js

2.3.2 小程序页面page命名规范

小程序页面命文件命名方式与常规前端一致，但是每个页面的文件，必须放在一个目录中，不允许单独暴露页面文件，目录下的文件名称与目录名称保持一致。
例:

```
|-- my-page
    |-- my-page.js
    |-- my-page.json
    |-- my-page.wxml
    |-- my-page.wxss
```

2.3.3 小程序自定义组件命名规范（同样适用于Vue项目和uniapp项目）

- 组件目录文件采用全小写英文方式命名，多个单词使用连字符"-"分割，目录下的文件命名为index;
  例:

```
|-- components
|-- get-avatar
    |-- index.js
    |-- index.json
    |-- index.wxml
    |-- index.wxss
```

  目录下有多个文件时:

```
|-- components
|-- get-avatar
    |-- my-avatar
        |-- index.js
        |-- index.json
        |-- index.wxml
        |-- index.wxss
    |-- your-avatar
        |-- index.js
        |-- index.json
        |-- index.wxml
        |-- index.wxss
```

3.JavaScript开发规范

JavaScript开发规范可以参考Airbnb JavaScript开发规范:
Airbnb JavaScript 代码规范

3.1 命名规范

3.1.1 变量命名规范

- 常量命名规范：字母全部大写，单词之间使用下划线"_"分割

```
const MY_NAME = "xxx";
const MY_AGE = "xxx";
```

- 变量命名规范：全英文小驼峰式命名，命名必须能够反应当前变量作用，不允许出现无意义命名，必要时添加注释进行解释;

```
let myName = "xx";
let myAge = "xx";
```

- 变量命名时，除常量外，统一使用let声明，并且要注意变量作用域，尽量避免局部变量与全部变量命名重复，注意变量的作用域链

3.1.2 函数命名规范

- 全英文小驼峰式命名，命名必须能够反应当前函数作用，不允许出现无意义命名，函数前必须添加注释;

```
/**
 * 获取用户姓名 -- 说明函数作用，尽量简短
 * @method getUserName -- 函数名称
 * @param {string} name 用户姓名 -- 说明函数参数，{参数类型，基本类型使用小写，引用类型首字母大写}
 * @param {string} [sex] 用户性别 -- 代表该参数为可选参数
 * @param {string} [age = 17] 用户年龄 -- 标明参数有默认值，并且默认值是17
 * @return {string} name 返回用户姓名 -- 说明函数返回值，如果没有，不写
 */
function getUserName(name, sex,age = 13) {
   return name;
}
```

- 函数命名，使用动词+名词形式进行命名，不允许使用其他格式;

```
/**
 * 获取用户姓名
 * @method getName
 * TODO -- 代表该功能等待开发
 */
function getName() {}

/**
 * 创建用户
 * @method createUser
 * TODO
 */
function createUser() {}

/**
 * 删除用户
 * @method deleteUser
 * TODO
```

```
 */
function deleteUser() {}
```

- 增、删、改、查、详情，一律使用`add / update / delete / detail / get`;

常用命名动词				
get 获取/set 设置	read 读取/write 写入	add 加入/append 添加	input 输入/output 输出	browse 浏览
add 增加/remove 删除	load 载入/save 保存	clean 清理/clear 清除	encode 编码/decode 解码	edit 编辑
create 创建/destory 移除	create 创建/destroy 销毁	sort 排序	encrypt 加密/decrypt 解密	select 选取/mark 标记
start 启动/stop 停止	begin 开始/end 结束	find 查找/search 搜索	emit 生成	copy 复制/paste 粘贴
open 打开/close 关闭	bind 绑定/separate 分离	build 构建	download 下载/upload 上传	insert 插入/delete 移除
refresh 刷新	update 更新/revert 复原	push 推/pull 拉	expand 展开/collapse 折叠	submit 提交/commit 交付
enter 进入/exit 退出				

3.1.3 类命名规范

- 采用全应为大驼峰式命名，命名必须能够体现当前类的作用，不允出现无意义命名，类前必须添加注释;

```
/**
 * 用户类 -- 说明类作用，尽量简短
 * @class Person -- 类名称
 * @author yangming -- 标明类作者
 */
class  User {
    /**
     * Person构造函数
     * @constructor -- 声明构造函数
     * @param {srting} name 用户姓名
     * @param {string} sex 用户性别
     */
    constructor(name, sex) {
        this.name = name;
        this.sex = sex;
    }

    /**
```

```
     * 获取用户姓名
     * @method getName
     * @return {string} this.name 返回用户姓名
     */
     getName() {
         return this.name;
     }
}

/**
 * 本地用户类
 * @class LocalUser
 * @exteds User -- 标明父类
 * @author yangming
 */
class LocalUser extends User {
    /**
     * LocalUser构造函数
     * @constructor
     * @param {string} name 用户姓名
     * @param {string} sex 用户性别
     * @param {string} address 用户地址
     */
    constructor(name, sex, address) {
        super(name, sex);
        this.address = address;
    }

    /**
     * 获取用户地址
     * @method getAddress
     * @return {string} this.address 返回用户地址
     */
    getAddress() {
        return this.address;
    }
}
```

3.1.4 对象属性命名规范

- 对象属性使用小驼峰式命名，属性名不允许使用双引号包含

```
let arr = {
    myAge: "13",
    name: "xx"
};

// 禁止以下写法
let arr = {
    "myAge": "13",
    "name": "xx"
}
```

3.2格式规范

3.2.1 缩进、空格、换行

3.2.1.1 缩进

- 使用Tab或空格缩进，无论使用哪一种，要保持整个项目缩进方式一致，为保证缩进效果，在vscode中，选择下方制表符长度(空格)，点击选择使用制表符输入，配置制表符长度为4。

3.2.1.2 空格、换行

- 在大括号前使用空格/换行

```
/**
 * 获取用户姓名
 * @method getUserName
 * @param {string} name 用户姓名
 */
function (name) {

}

function (name)
{

}

let flag = true;
if (flag) {

}

if (flag)
{

}

for (let i = 0; i <= 3; i++) {

}

for (let i = 0; i <= 3; i++)
{

}
```

- 在运算符前使用空格

```
// 注意，此处命名为了方便展示，正式项目中禁止使用无意义命名，详情见变量命名规范一节;
let a = 3;
let b = 4;
let c = a + b;
let d = 3 * 4;
```

- 在逗号后使用空格/换行

```
let name = getName("yangming", "26", "湖北省武汉市");

let user = {
    name: "xx",
    age: 13,
    address: "xx"
};
```

- 箭头函数中使用空格/换行

```
// 注意，此处命名为了方便展示，正式项目中函数前必须添加注释，详情见函数命名规范一节
let getName = () => {

}
```

- 注释符号后使用空格/换行

```
// 注释

/*
    多行注释
*/

/**
 * doc注释
 * doc注释
 */
```

- 不同代码块之间使用换行，只需要一个换行

```
// 正式项目必须添加注释
function getName() {

}

function getSex() {

}

for (let i = 0; i < 3; i ++) {

}
```

- for循环中使用空格

```
for (let i = 0; i < 3; i ++) {

}

for (let i = 0; i < 3; i++)
{

}
```

- 数组，对象中使用空格/换行

```
let arr = [1, 2, 3, 4];
let user = {
    name: "xx",
    sex: "xx",
    age: "xx"
};

let userList = [
    {
        id: 1,
        name: "xx",
        age: "xx"
    },
    {
        id: 2,
        name: "xx",
        age: "xx"
    },
    {
        id: 3,
        name: "xx",
        age: "xx"
    }
];
```

- 长字符串中间不使用换行

3.2.2 引号使用

- 所有引号一律使用双引号

3.3 语法规范

- 优先使用es6以上的语法和语法糖

3.3.1 函数/对象声明

- 声明数组和对象时，使用字面语法声明;

```
let user = {};
let age = [];
```

3.3.2 使用缩写语法

- 对象内部语法使用缩写语法，使用缩写语法时注意，缩写属性必须写在对象开头位置，不允许穿插

```
let name = "xx";
let age = "xx";

let user =  {
    name,
    age,
    sex: 'xx',
    getName() {

```

```
    },

    // 禁止以下写法
    sex: 'xx',
    name,
    job: 'xx',
    age: age, // 此处应使用缩写
    getName: function() {

    }
}
```

3.3.3 禁止省略分号

- 代码编写过程中，不允许省略分号

3.3.4 代码块标记

- 代码编写中， 代码块必须使用{}，不允许省略

```
// 禁止以下写法
if (a > b) alert(1);

if (a > b)
    alert(1);

for (let i = 0; i <= 3; i++) alert(1);
for (let i = 0; i <= 3; i++)
    alert(1);
// 正确写法
if (a > b) {
    alert(1);
}

for (let i = 0; i <= 3; i++) {
    alert(1);
}
```

3.3.5 字符串拼接

- 使用es6字符串模板代替"+"拼接字符串

```
let age = "xx";
let name = "xx";
let user = `name is ${name}, age is ${age}`;
```

3.3.6 箭头函数

- 代码书写过程中，匿名函数尽量使用箭头函数，使用时要注意this指向问题

```
let arr = [1, 2, 3];
arr.forEach(item => {

});

// 当箭头函数的函数体只有返回值时，采用简写形式

let arr = [1, 2, 3];
arr.map(item => item * 3);
```

3.3.7 三目运算

- 简单的判断语句可以使用三目运算，嵌套超过2层的三目运算，要使用if...else改写

4. HTML开发规范

4.1 HTML 、css、js分离

- html文件中，只包含html代码，样式、js全部分离为单独的文件，在html中引入使用。
- 引用js时注意顺序，引入的第三方库必须在head底部中引入，自己的js文件，必须在body底部引入，css文件一律在head底部引入，除必要情况，不要在内容板块中，引入js或css文件
- css文件和js文件引入时，css引入放在一起，js引入放在一起

```
<head>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="tools.css">
    <script src="/js/jquery-1.9.1.min.js"></script>
</head>

<body>
    <div>
        ...
    </div>

    <script src="js/main.js"></script>
</body>
```

4.2 内联样式

- html文件中，应尽量避免使用内联样式，样式文件全部写在单独的.css文件中，如果需要更改样式优先级，可以通过ID选择器或!important实现

4.3 书写规范

4.3.1 属性、属性值

- 所有属性必须使用英文小写，不允许使用大写，属性与属性值之间的等号前后不加空格

```
<div class="my-class"></div>

<!-- 禁止以下写法 -->
<DIV CLASS="xx"></DIV>

```

- id、class、name等属性值，全部使用纯小写英文命名，多个单词之间使用连字符"-"分割

```
<div class="my-class" id="my-id"></div>
```

4.3.2 命名规范

- class、id等命名时，要能够表达明确意义，不允许出现无意义命名，命名优先使用英文命名
- 不同模块，可以使用通用的class，但是要添加当前模块自己的class或者是id

```
<!-- 文章列表 -->
<div class="content-border articl-list">

</div>
<!-- /文章列表 -->

<!-- 文章内容 -->
<div class="content-border article-detail">

</div>
<!-- /文章内容 -->
```

4.3.3 标签闭合

- 自闭合标签(例如：img、input、br、hr等)必须使用"/"进行闭合
- 非自闭合标签必须关闭

```
<div>

</div>

<br />

<input />
```

4.3.4 换行、缩进

- html结构必须使用缩进来明确表示包含和层级关系，同一级元素必须对齐，子元素与父元素前使用一个Tab缩进（为保证缩进统一，在vscode中，点击右下方空格/制表符长度，选择使用tab缩进，并设置制表符长度为4，vscode中使用`alt`+`shift`+`f`可以自动格式化）
- 尽量避免多个元素出现在同一行
- 不同模块之间使用换行进行分割，并且使用注释来 标记模块开始和结束位置（此处注意灵活使用，大模块必须使用注释标记开始结束，小模块或容易理解的模块，可以不使用注释标记）
- 注释的内容两边要有空格

```
<!-- 文章列表 -->
<div>
    xxx
</div>
<!-- /文章列表模块 -->

<!-- 文章内容模块 -->
<div>

</div>
<!-- /文章内容模块 -->
```

4.3.5 img

- 除小型icon外，其余大图片，全部统一上传到服务器后，使用链接引入

4.3.6 引号使用

- 所有引号一律使用双引号，特殊情况下，可以单引号双引号交叉使用

5. Css开发规范

5.1 注释

- 不同模块的样式，应使用注释进行标明当前模块功能

5.2 书写规范

- 书写样式时，尽量避免直接书写某一个class或者id的样式，应该体现出当前样式所属的模块

```
/* 文章模块 */
.article .article-list .article-detail .title {
    ...
}
```

- 书写样式是，一行只能写一个样式，不能将多个样式写在同一行，样式结尾处必须有分号

```
.article {
    display: flex;
    margin: 0 23px;
    padding: 0 23px;
    border: 2px solid #ccc;
}
```

- 样式书写时，按照固定顺序书写：Position(top, right, bottom, left, z-index等) -> Box Style(display, width, height, margin, padding, border, overflow等) -> Font(font, color ,line-height, text-align等) -> 背景(background等) -> other(opacity, animation等)
- css中出现的所有顺序，全部按照上右下左的顺序书写

```
.div {
    position: absolute;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
    z-index: 1;
```

```
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    width: 100px;
    height: 100px;
    margin: 20px;
    padding: 20px;
    border: 2px solid #cccccc;
    border-radius: 10px;
    overflow: hidden;
    font-size: 32px;
    font-weight: bold;
    color: #ff6600;
    text-align: center;
    background-color: #000;
    opacity: .8;
}
```

- css选择器尽量使用一种选择器，避免多种选择器混合使用

*** 代码书写过程中，请严格遵守以上规范**