前端标准

一、项目目录结构规范

1.1 资源分类

资源分为两大类：
1、源代码资源：指开发者编写的源代码，包括 js、html、css、template 等。
2、内容资源：指希望做为内容提供给访问者的资源，包括图片、字体、flash、pdf 等。

1.2 目录命名原则

1、简洁。**有习惯性缩写的单词，必须采用容易理解的缩写**。
例如：

- src: 源文件目录。不允许使用 source 等。
- img: 图片。不允许使用 image、images、imgs 等。
- js: javascript 脚本。 不允许使用 script、scripts 等。
- css: 样式表。 不允许使用 style、styles 等。
- swf: flash。 不允许使用 flash 等。
- dep: 引入的第三方依赖包目录。 不允许使用 lib、library、dependency 等。

2、**不允许使用复数形式**。如：imgs、docs 是不被允许的。

1.3 目录划分

1.3.1 根目录结构划分

在根目录下，目录结构必须按照职能进行划分，不允许将资源类型或业务逻辑划分的目录直接置于根目录下。

1.3.2 业务项目目录结构划分

1、**根据业务逻辑划分 src 目录结构**

业务项目的 src 目录内，绝大多数情况应当根据业务逻辑划分目录结构。划

分出的子目录我们称为业务目录。

src 下必须只包含业务目录与 common 目录。业务公共资源必须命名为 common。common 目录做为业务公共资源的目录，也视如业务目录。

2、业务目录划分原则

（1）JS 资源不允许按资源类型划分目录，必须按业务逻辑划分目录。JS 资源应直接置于业务目录下。即：业务目录下不允许出现 js 目录。

（2）除 JS 资源外的源文件资源，当资源数量较多时，为方便管理，允许按资源类型划分目录。即：业务目录下允许出现 css、tpl 目录。

（3）内容资源允许按资源类型划分目录。即：业务目录下允许出现 img、swf、font 目录。

（4）业务目录中，如果文件太多不好管理，需要划分子目录时，也必须继续遵守根据业务逻辑划分的原则，划分子业务。

对于一个业务目录， 应将业务相关的源文件资源都直接置于业务目录下。

```
|—admin
  |—img
    |—add_button.png
  |—add.js
  |—add.tpl.html
  |—add.css
```

业务目录下源文件资源数量较多时，我们第一直觉应该是：是否业务划分不够细？是否应该划分子业务，建立子业务目录？

遇到确实是一个业务整体，无法划分子业务时，允许将非 JS 资源按资源类型划分目录进行管理。

```
|—admin
  |—css
    |—add.css
    |—edit.css
    |—remove.css
  |—img
    |—add_button.png
  |—tpl
    |—add.html
    |—edit.html
    |—remove.html
  |—add.js
  |—edit.js
  |—remove.js
```

3、业务项目目录划分示例

```
|—NHA
  |—src
    |—common
      |—img
        |—sprites.png
        |—logo.png
      |—conf.js
      |—layout.css
    |—zcyh
      |—img
        |—add_button.png
      |—add.js
      |—add.tpl.html
      |—add.less
    |—lwjc
      |—zbgl
        |—list.js
        |—list.tpl.html
        |—list.css
      |—tjfx
    |—dep
      |—er
          |—src
          |—test
      |—esui
          |—src
          |—test
  |—test
  |—doc
  |—index.html
  |—main.html
    ......
```

1.4 常用目录

src: src 目录用于存放开发时源文件。

dep: dep 目录用于存放项目引入依赖的第三方包。该目录下的内容通过平台工具管理，项目开发人员 不允许更改 dep 目录下第三方包的任何内容。

当项目需要修改引入的第三方代码时，第三方包应将源码直接置于/src 目录下。

tool：tool 目录用于存放开发时或构建阶段使用的工具。

test：test 目录用于存放测试用例以及开发阶段的模拟数据。

doc：doc 目录用于存放项目文档。项目文档可能是开发者维护的文档，也可能是通过工具生成的文档。

asset：asset 目录用于存放用于线上访问的静态资源。

二、JavaScript 编码规范

2.1 代码风格

2.1.1 文件

除需要预编译的文件外，全部使用.js 扩展名。

2.1.2 结构

1、缩进
两个空格为一个缩进，换行后添加一层缩进。
2、语句
解构多个变量时，如果超过行长度限制，每个解构的变量必须单独一行。

2.2 语言特性

2.2.1 变量

Es6 中，使用 let 和 const 定义变量，不使用 var。

2.2.2 函数

不要使用 arguments 对象，应使用 ...args 代替。

示例：

```
// good
function foo(...args) {
    console.log(args.join(''));
}

// bad
function foo() {
    console.log([].join.call(arguments));
}
```

2.2.3 对象

不建议使用 for..in 进行对象的遍历，以避免遗漏 hasOwnProperty 产生的错误。

2.2.4 集合

需要一个不可重复的集合时，应使用 Set。

2.2.5 异步

1、回调函数的嵌套不得超过 3 层。
2、建议使用 Promise 代替 callback。
3、不允许直接扩展 Promise 对象的 prototype。
4、建议使用 async/await 实现同步操作。

2.2.6 注释

1、文件注释
　　文件注释用于告诉不熟悉这段代码的读者这个文件中包含哪些东西。 应该提供文件的大体内容，它的作者，依赖关系和兼容性信息。

2、函数注释
（1）函数/方法注释必须包含函数说明，有参数和返回值时必须使用注释标识；

（2）参数和返回值注释必须包含类型信息和说明；
（3）当函数是内部函数，外部不可访问时，可以使用 @inner 标识；
示例：

```
/**
 * 函数描述
 *
 * @param {string} p1 参数 1 的说明
 * @param {string} p2 参数 2 的说明，比较长
 *       那就换行了.
 * @param {number=} p3 参数 3 的说明（可选）
 * @return {Object} 返回值描述
 */
function foo(p1, p2, p3) {
    var p3 = p3 || 10;
    return {
        p1: p1,
        p2: p2,
        p3: p3
    };
}
```