# 大家网络前端开发规范

## 前言

为了在团队开发中，提升开发效率和协作能力，制定前端开发规范。该规范基于几个前提：

1. html和css兼容范围不仅限于W3C的html5规范标准（不包括移动端）；
2. js基于jquery-1.7.2库；

## HTML代码规范

在PC端，为了保证各浏览器之间兼容性，尽量以基础标记编写。标记ID和class名应当和标记的功能相对应。

1. 容器以div为主，尽量不要将元素标记（如：<a>）设置成容器；
2. 对于多个同类属性的以ul+li为主；
3. 标记ID以父元素ID为前缀，自身名追加在父元素名后面，并以”-”隔开；

例如：

<div id=”navigation”>

<div id=”navigation-header”></div>

</div>

1. 标记class以父元素class为前缀，自身类名追加在父元素类名后面，并以”\_”隔开；

例如：

<div class=”navigation”>

<div class=”navigation\_header”></div>

</div>

1. 图片统一使用background属性显示，避免使用<img>；

## CSS代码规范

#### 定位规范

1. 将界面按功能做一次基础布局，基础布局定位使用position:absolute;
2. 对于固定显示的基础容器按1定位，同时设置top、left、right、bottom，或者top、left、width、height，其他根据具体需求使用上述方式定位；
3. 对于多个同属性使用float排版；
4. 对于固定宽度、高度的，通过2定义高度或宽度，或者直接指定宽度和高度；

#### 类规范

1. 对于同一类属性的标记通过定义同一类名来配置显示数据；
2. 对于出现类似“选中”的特殊类提供唯一类名，并定义差异性属性，在js中通过addClass和removeClass来增加和删除；
3. 灵活使用“空类”：如临时增加一个类名，表明正在处理，类存在时不接收新处理，当正在处理完成后，清空类；

#### ID规范

1. ID的作用是用来表示唯一；
2. 对于插件的ID通过数据的主键或者时间戳来标注唯一性；
3. ID名使用英文，必须达意，禁止使用汉语拼音或者随意取不知所谓的名；

### JS代码规范

#### 基类

基类写法参照以下：

(function(global){

var classname = function(opts){

this.\_\_x1 = opts;

prepare.call(this);

};

function prepare(){

console.log(this.\_\_x1);

};

classname.prototype.init = function(contain){

this.\_\_y1 = contain;

};

var instance = this;

var proto = classname.prototype;

Global.classname = classname;

})(this);

基类变量使用this.xxx私有变量形式，禁止定义到类外部，如上述红字部分。如果变量值是唯一且不变的，可以放在类外部定义，如上述蓝字部分。

#### 子类

子类写法参照如下：

FW.Son = function(){

Var son = new classname();

Var x1,y1;

son.ask = function(para){

x1=para;

answer(‘hello world’);

};

function answer(para){

y1=para;

};

return son;

};

子类调用父类方法禁止使用如：son.init.apply(this)，将子类this替换父类this,如需传递子类this，使用参数传递。

#### 固定功能

如用于配置的js，使用对象式写法：

FW.Configure = {

Runtime:{

Webmode: ‘comfort’,

Cate: undefined

},

Enviroment:{

Html5: false,

getWebbrowser(){

return ‘ie’;

}

}

};

### JS开发规范

#### 命名规则

所有插件使用FW.UI.Components开头（摒弃以前YX开头）；

所有页面使用FW.Frame开头；

其中，FW为Framwork缩写。

#### 插件

所有插件基于Widget类，插件禁止重写Widget类中的方法。

具体写法如下：

FW.define(

undefined, //预留，做依赖js所在路径

['scripts/Components/memo/memo'], //数组：依赖js模块名

'scripts/Components/delayering/main', //string：本模块的模块名

function(a, b, c){ //object：本模块主体

//new基类，赋值本模块名

var myWidget = new Widget('scripts/Components/delayering/main');

//私有函数

function X(){

…

};

//公共方法

myWidget.Open = function(m,n){

…

};

return myWidget;

});

调用：

var delayering = FW.require('scripts/Components/delayering/main')(a,b,c);

#### 界面

所有页面基于Framework类，界面禁止重写Framework类的方法。

具体写法如下：

FW.define(

undefined, //预留，做依赖js所在路径

['scripts/views/navigation/navigation'], //数组：依赖js模块名

'scripts/frame', //string：本模块的模块名

function(a, b, c){ //object：本模块主体

//new基类，赋值本模块名

var myFrame = new Framework('scripts/frame');

//私有函数

function X(){

…

};

//公共方法

myFrame.Open = function(m,n){

…

};

return myFrame;

});

调用：

var frame = FW.require('scripts/frame')(a,b,c);

#### 其他

配置类js，按照“对象”写法；

功能类js，按照基类写法；

### 大家网络前端框架（测试版）原理

#### 模块请求

框架参照requirejs写法。用“界面类”作为例子：

FW.define(

undefined, //预留，做依赖js所在路径

['scripts/views/navigation/navigation'], //数组：依赖js模块名

'scripts/frame', //string：本模块的模块名

function(a, b, c){ //object：本模块主体

});

FW.define(…)作用是：

1. 保存依赖项路径（预留）；
2. 保存所有直接依赖项；
3. 保存本模块名；
4. 保存本模块主体；

上述4项被保存在缓存中，当需要请求某个模块时：

var frame = FW.require('scripts/frame')(a,b,c);

其中，FW.require返回的是模块主体的实例，也可写作

var frame = new FW.require('scripts/frame')(a,b,c);

两者区别在于对模块主体的加载原理（详情查看js对象加载原理）。

#### 模块加载

鉴于目前我们模块较小，采用ajax同域同步请求。目前测试没有对主线程产生阻塞影响。

所有被依赖模块使用ajax请求，请求方式为：

$.ajax({

type:'GET'

,url: url

,async: false

,dataType: 'script'

});

最终js脚本被缓存为script属性文本保存在浏览器中，其中加载时jquery自动给js脚本加上时间戳，所以被依赖模块发生更改时，不受浏览器缓存影响（如果以js文件形式存在，浏览器会优先加载本地已经缓存的js，导致发生更改的js没有被加载）。

注意：被ajax请求的js不会出现在调试器的source中，调试请在调试部分加入debugger。

#### 模块间通信

框架基类Framework和Widget，提供模块间“通信”，分为1、请求监听其他模块的特定消息，2、分发本模块的特定消息。具体方式如下：

FW.define(

undefined, //预留，做依赖js所在路径

['scripts/views/navigation/navigation'], //数组：依赖js模块名

'scripts/frame', //string：本模块的模块名

function(a, b, c){ //object：本模块主体

var frame = new Framework(‘scripts/frame’);

function init(){

frame.listen('scripts/views/navigation/navigation', callback, ‘user\_message’);

};

function callback(target, from, events, paras){

};

});

其中，第一个参数是监听的模块名，第二个是消息回调函数，第三个是监听的自定义消息。那么模块分发为：

FW.define(

undefined, //预留，做依赖js所在路径

undefined, //数组：依赖js模块名

'scripts/views/navigation/navigation', //string：本模块的模块名

function(a, b, c){ //object：本模块主体

var navigation = new Framework('scripts/views/navigation/navigation');

$(‘#...’).bind(‘click’, function(ev){

navigation.dispatch(this, ‘user\_message’, paras);

}）;

});

其中，第一个参数始终是this，第二个参数是自定义消息，第三个是传递的参数。

注意：不管是监听还是分发，与对象模块的存在与否、加载与否没有关联。