**YP（云派）框架定位是一个业务框架，非底层的基础框架，核心目标是封装常规业务场景，并摆脱前端项目对特定外部基础库的依赖导致升级困难、语法不统一等诸多问题。**

**公共函数接口：**

框架的辅助函数采用类jQuery的封装，易移植并达到去jQuery化的目的。希望今后的项目升级，统一通过升级框架实现，不要在页面中留下难以升级或即将淘汰的方法函数。

**加载模块 使用yp.use**

yp.use([‘aaa’, ‘bbb’], function(yp) {

// 模块加载完执行的回调

});

**监听框架加载完成事件 使用 yp.ready**

yp.ready(function() {

// Dom和框架加载完成的回调

});

**yp.each代替 $.each**

yp.each([9,8,7], function(val, key) {

console.log(val, key);

});

yp.each在使用上的注意事项：基本语法与jq一致，但回调里的参数顺序与jq相反，这是按照w3c标准做的封装，同时解决foreach无法中断循环的局限。

**yp.format 代替 字符串拼接**

yp.format(‘<li name={name}>{val}</li>’, {name:’aaa’, val:’bbb’});

同时支持：yp.format(‘<li name={0}>{1}</li>’, ’aaa’,’bbb’);

**yp.mix 代替 $.extend**

yp.mix({}, aaa, {a:1, b:2});

**yp.ajax 代替 $.ajax**

yp.ajax(url, {type:post, data:{a:1}})

.done(function() {

//统一格式，一律采用异步接口定义回调

});

**自定义事件 使用yp.sub和yp.pub**

yp.sub(‘事件名’, function(e, data) {

// 定义要监听的自定义事件名和回调

// e是事件对象，data是参数对象

});

yp.pub(‘事件名’, {a:1, b:2});

sub代表监听事件，pub代表发布事件（也可以理解为触发事件），unsub代表取消监听

建议将表单提交或者按钮点击里大的处理逻辑代码拆分到几个小的自定义事件去。

**以上方法配合mvvm模板框架，几乎不需要操作dom结构和使用jq函数。**

**主要扩展模块介绍：**

**yp-loger：调试日志模块**

日志输出方法：yp.loger.log，通过mode参数可以控制输出位置

调用方式1（常规场景）：yp.log(123);

调用方式2（改变日志输出方式）：yp.loger.mode = ‘alert’;yp.log(123);

调用方式3（临时改变输出方式）：yp.log.bind({mode:’dom’}, 123)();

**yp-** **cache：全局缓存模块**

获取存储对象：cache = yp.localStorage;

添加或更新数据：cache.setItem('key', 123);

查询数据：cache.getItem('key'); // 返回123

删除数据：cache. removeItem (‘key’);

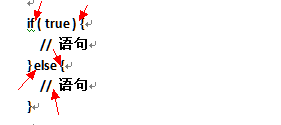
**yp-loader：数据加载模块**

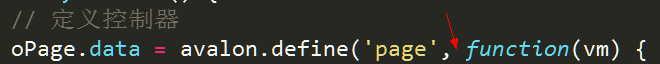
主要方法有yp.ajax、yp. ajaxSubmit

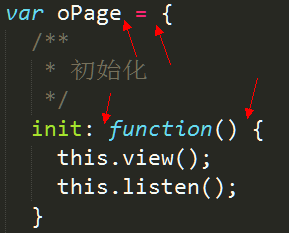
异步请求：yp. ajax(‘url’, {a:1});

提交表单：yp. ajaxSubmit(‘url’);

**项目编码建议风格：**

****

****

****

红色箭头处均要加空格，缩进也采用空格符，建议在编辑器中调整为两个字符间隔。

oPage代表页面对象，作为所有业务逻辑的统一入口，标配init、view、listen作为基础接口分别代表初始化，视图绘制、事件绑定。

具体细节参考附件中的例子

**其他参考：**

<https://github.com/rwaldron/idiomatic.js/tree/master/translations/zh_CN>

**css建议风格细则：**

1. 每个属性单独一行，避免多个属性同行
2. 每个属性结尾必须带分号作为结束
3. 属性顺序参考编码规范要求，属性命名尽量使用通用名称
4. 模块头部要写上注释说明，对复杂的样式或特效也尽量加上注释说明
5. 大的模块之间要加空行分隔
6. 尽量使用符合标准的样式，减少hack的使用