|  |
| --- |
| AvePoint |
| GA GUI编码规范 |
|  |
|  |
| **Jianxin.Li@avepoint.com** |
| **8/17/2015** |

|  |
| --- |
|  |

*内容目录*

Contents

[1 概览 1](#_Toc427148917)

[1.1 原则和主旨 1](#_Toc427148918)

[1.2 术语 2](#_Toc427148919)

[2 Web编码规范 3](#_Toc427148920)

[2.1 Model编码规范 3](#_Toc427148921)

[2.2 View编码规范 3](#_Toc427148922)

[2.3 Controller编码规范 4](#_Toc427148923)

[2.4 CSS编码规范 4](#_Toc427148924)

[2.5 JavaScript编码规范 4](#_Toc427148925)

[2.5.1 变量声明和初始化 4](#_Toc427148926)

[2.5.2 JavaScript中国际化词条的使用 5](#_Toc427148927)

[2.5.3 JavaScript命名空间 6](#_Toc427148928)

[2.5.4 页面错误提示 6](#_Toc427148929)

[2.5.5 HTTP Request 6](#_Toc427148930)

[2.5.6 事件处理 7](#_Toc427148931)

[2.5.7 DOM操作 8](#_Toc427148932)

[2.5.8 循环 8](#_Toc427148933)

[2.5.9 其他注意事项 8](#_Toc427148934)

# 概览

本文档初步设计为GA+项目组所使用的web端 编码规范 ，一般地也适用于其他的项目规范。该规范源自于产品开发过程中的经验，并在不断完善。如果您发现一些最佳实践或者话题并没有涵盖在本文档中，请联系[我](mailto:Jianxin.Li@avepoint.com)，以不断充实完善本文档。

一般说来，任何指导准则都可能会导致众口难调。本规范的目的在于帮助开发团队提高开发效率，减少代码中可能出现的bug，并增强代码的可维护性以及代码的性能。万事开头难，采纳一个不熟悉的规范可能在初期会有一些棘手和困扰，但是我们相信这些不适应很快便会消失，它所带来的好处和优势很快便会显现，特别是在当接手他人代码进行维护开发时。

## 原则和主旨

高质量web的代码往往具有如下特质:

1. **可维护性**– 代码必须易读且简单明确。它们必须能展示出重点所在。代码的相关部分应当易于重用。代码不可包含多余代码。他人维护时可以很快而看懂逻辑并且容易修改。
2. **高效的性能**– 代码不能存在明显的效率问题。在正常的case范围内，需要尽快的响应用户的操作
3. **良好的用户体验**–用户进行某些操作后需要给出合理的提示，不能让用户认为之前的操作没有生效。不能在进行某些操作后浏览器直接卡死。

合理使用编程实践，设计和语言特性决定了代码是否可以很好满足上述特性。本编程规范致力于帮助创建代码示例以使使用者能够作为最佳实践来效仿和学习。

## 术语

在整个文档中，会有一些对于标准和实践的推荐和建议。一些实践是非常重要的，必须严格执行，另一些指导准则并不一定处处适用，但是会在特定的场景下带来益处。为了清楚陈述规范和实践的意图，我们会使用如下术语。

|  |  |  |
| --- | --- | --- |
| 术语 | 意图 | 理由 |
| 🗹 **要...** | 该规范或实践在任何情况下都应该遵守。如果您认为您的应用是例外，则可能不适用。 | 该规范用于减少bug。 |
| 🗷 **不要...** | 不允许应用该规范或实践。 |
| 🗹 **考虑...** | 该规范和实践适用于大多数情况。 | 该规范用于统一编程风格，保持一致和清晰的风格。 |
| 🗷 **避免..** | 不应该应用该规范或实践，除非有合理的理由。 |
| 🗹 **可以…** | 该标准和规范可以按需应用。 | 该规范可用于编程风格，但不总是有益的。 |

# Web编码规范

通用编程规范对代码风格，格式和结构提供了全局通用的指导。

## Model编码规范

🗹 **要**确保命名简洁明了，能体现出命名规范。

🗹 **要**确保内容属性合理，保证各属性有效性，不冗余。

🗹 **要**确保内容覆盖对应View中需要的数据。

## View编码规范

🗹 **要**使用bundle来在View中引用js和css文件。

🗹 **要**将HTML和CSS进行隔离，**避免**使用inline style。

🗹 **要**使用Pascal命名方式来为元素设置id，该规范与Html扩展方法中的元素id一致。

🗹 **要**将HTML和JavaScript进行隔离，**避免**在HTML中嵌入script标签。

🗹 **考虑** 将View中逻辑操作分离到controller中去。

🗷**避免** 使用Html.EditorFor，改为使用Partial或扩展方法，这样能够规范化页面，并且有助于提高页面效率。

🗷 **避免** 使用ViewData，ViewBag,TempData，尽量使用Model来传递数据。

🗷 **不要** 在HTML元素上直接绑定事件，要在JavaScript中进行处理。

🗷 **不要** 为HTML元素添加重复ID，**考虑**页面重复引用情况下ID的使用。

## ­Controller编码规范

🗹 **考虑** 提高Controller中Action的重用性。

🗷 **不要** 在Controller中处理任何页面逻辑，Controller只负责处理页面Action。

🗷 **不要** 在Controller中new其他Controller的实例。

🗷 **不要** 在Controller中使用Partial Class，**考虑**使用继承。

## CSS编码规范

🗹 **要** 在style sheet中定义class，**不要**在View中定义class。

🗹 **要** 在定义css class时使用小写单词，单词间以‘-’分隔。

🗹 **要** 在新添加页面时使用已有样式，**避免**每个页面单独设置自己的样式。

🗹 **考虑** 使用伪类来控制一些特效，**避免**使用JavaScript事件来动态更改style控制。

## JavaScript编码规范

### 变量声明和初始化

🗹 **要**使用Camel命名规范来定义方法和属性以及私有变量，Pascal来定义构造函数。

// code sample:

function Person(name) {

var age,

setAge = function (newAge) {

age = newAge;

};

this.name = name;

this.sayName = function () {

alert(this.name);

}

}

🗹 **要** 使用一个var声明一个function内的所有变量。

🗹 **要** 使用字面量来定义Object或Array。

//code sample

var a = [1, 2, 3],

o = {

color: "red",

size: 15

};

Bad:

//code sample

var a = new Array(),

o = new Object();

a.push(1);

a.push(2);

a.push(3);

o.color = "red";

o.size = 15;

### JavaScript中国际化词条的使用

🗹 **要** 使用GAUtility中GetString方法来获取国际化词条，**不要**再使用hidden或直接在View页面获取。

//code sample:

//controller

this.ViewData["resources"] = new

{

key = GAResource.GetString("key")

};

//js

var i18nItem = gaUtility.GetString("key")

Bad:

//code sample:

//view

@Html.Hidden("key", GAResource.GetString("key\_Remote\_Farm"))

//js

var i18nItem = $("#key").val();

Bad:

//code sample

//view

<script>

var i18nItem = '@GAResource.GetString("key")';

</script>

### JavaScript命名空间

🗹 **要** 在JavaScript中使用命名空间来避免全局变量泛滥。

//code sample:

Namespace.Register("GA.Web.Scripts");

GA.Web.Scripts.ModelName = function () { };

### 页面错误提示

🗹 **要** 在使用统一的class来控制页面错误提示，**不要**通过设定inline style控制，这样可以方便的确认页面是否存在错误，而不用去挨个验证每个field。

//code sample:

//view

<div class="field-validation-valid" id="errorMessage"></div>

//js

if (valid) {

$("#errorMessage").prop("class", "field-validation-valid");

} else {

$("#errorMessage").prop("class", "field-validation-error");

}

Bad:

//code sample:

//view

<div style="color: red; display: none;" id="errorMessage"></div>

//js

if (valid) {

$("#errorMessage").hide();

} else {

$("#errorMessage").show();

}

### HTTP Request

🗹 **要** 使用异步ajax来加载数据。

🗹 **要**把获取数据和处理数据分离开来。

//code sample:

function getData(callback) {

$.ajax({

url: "url",

success: function (result) {

callback(result);

}

});

};

function dealData(data) { };

getData(dealData);

Bad:

//code sample

function getData() {

$.ajax({

url: "url",

success: function (result) {

//deal result

}

});

};

getData();

### 事件处理

🗹 **要**使用事件代理。

//code sample

$("#parent").bind("click", function (e) {

var elem = e.srcElement || e.target,

id = $(elem).prop("id");

switch (id) {

case "child1":

//xxx

break;

case "child2":

//xxx

break;

default:

break;

}

});

Bad:

//code sample

$("#child1").bind("click", function (e) {

//xxx

});

$("#child2").bind("click", function (e) {

//xxx

});

🗹 **要**解耦事件处理程序和应用逻辑

//code sample

$("#urlText").bind("blur", function (e) {

var url = $(this).val();

checkUrl(url);

});

function checkUrl(url) {

//check url

}

Bad:

//code sample

$("#urlText").bind("blur", function (e) {

var url = $(this).val();

//check url

});

### DOM操作

🗹 **要**缓存在一个function中使用多次的DOM元素。

🗷**避免**频繁更新DOM，在需要更改大量HTML节点时，一次性更新所有节点而不要每个节点挨个更新。

### 循环

🗹 **要**简化循环终止条件，如果是顺序无关的循环，使用减值迭代。

//code sample

var list = [1, 2, 3],

i;

for (i = list.length - 1; i >= 0; i--) {

//deal list[i]

}

🗹 **考虑**精简循环体，把不是必须在循环内执行的工作都放在循环体外。

### 其他注意事项

🗹 **要**用‘===’和‘!==’来判断相等和不等，**不要**用‘==’和‘!=’判断。

🗹 **考虑**使用switch代替if else在分支超过3个的时候，并且发生概率越高的写在越靠前的位置。

🗹 **考虑**使用‘+=’来拼接大量字符串而不是array.join。

🗷**避免**使用eval和Function。

🗷**避免**做重复工作。页面上有多个逻辑分支并且在当前页面只会执行同一条分支时，可以在第一次执行时把逻辑分支确定下来，避免之后执行再次判断。

//code sample

function A() {

if (type) {

A = typeA;

} else {

A = typeB;

}

}

🗷 **不要** 在setTimeout和setInterval中传递字符串参数。

🗷 **不要** 使用with语句。