**HTML规范**

**1.基础规范**

1.1 为每个HTML 页面的第一行添加标准模式（standard mode）的声明，这样能够确保在每个浏览器中拥有一致的展现。(Line1)

1.2 html根元素指定lang属性，从而为文档设置正确的语言。(Line2)

1.3 声明文档的编码charset，且与文件本身编码保持一致，推荐使用UTF-8编码。根据页面内容和需求填写适当的keywords和description。(Line4)

1.4 每个页面必须有且仅有一个title元素。(Line5)

1.5 在引入CSS和JavaScript文件时一般不需要指定type属性，因为text/css和text/javascript分别是它们的默认值。(Line6/7)

1.6 当link元素用于引用CSS文档时, 默认media是screen, 如为特殊终端提供样式, 请指定media属性, 如media=“print”。

1.7 css 引用置于头部<head>标签内。

1.8 js 引用置于底部</body>标签前。

**Tip:**

IE 支持通过特定的<meta>标签来确定绘制当前页面所应该采用的IE版本。除非有强烈的特殊需求，否则最好是设置为edge mode，从而通知 IE 采用其所支持的最新的模式。

<meta http-equiv="X-UA-Compatible" content="IE=Edge">

**2.属性顺序**

class （class是为高可复用组件设计的，所以应处在第一位）

id、name （id更加具体且应该尽量少使用，所以将它放在第二位）

data-\*

src、for、type、href、value

placeholder、title、alt

aria-\*、role

required、readonly、 disabled

**3. id/class命名规则**

3.1 **遵循“内容优先，表现为辅”的基本原则**

首先根据内容命名，如header、footer。若根据内容无法找到合适的命名，再结合表现进行辅助，如col-main、blue-box。

3.2 **一律小写，多个单词以“-”连接**

不能使用下划线和驼峰命名法，如main-nav。可基于最近的父元素名称作为前缀。

3.3 **在不影响语义的情况下，可适当使用缩写**

缩写只用来表示结构，如col、nav、btn等，不可自造缩写。

3.4 **避免广告拦截词汇**

ad、ads、adv、banner、sponsor、gg、guangg、guanggao等，页面中尽量避免采用以上词汇来命名

**4. 标签使用**

4.1 <base>标签必须放在<head>内。

4.2 <strong>标签用于强调重要性, <em>标签用于表示内容的着重点。[参考](https://link.jianshu.com?t=http%3A%2F%2Fwww.css88.com%2Farchives%2F644)

4.3 当link元素用于引用CSS文档时, 默认media是screen, 如为特殊终端提供样式, 请指定media属性, 如media=“print”;

4.4 img标签必须加alt，尤其是logo、商品图片等关键图片信息，对SEO友好。

4.5 减少标签的数量

编写 HTML 代码时，尽量避免多余的父元素。很多时候，这需要迭代和重构来实现。

4.6 所有html标签必须有结束符，<img />, <col />, <base />, <link />, <meta />, <input /> 除外。

**5. 注释规范**

页面中使用注释划分结构块，注意与css中的注释达成统一格式。

**CSS规范**

**1. 属性顺序**

1.1 位置属性(position、top、right、z-index、display、float等)；

1.2 大小(width, height, padding, margin等)；

1.3 文字系列(font、line-height、letter-spacing、color、text-align等)；

1.4 背景(background、border等)；

1.5 其他(animation、transition等)。

.declaration-order {
/\* Positioning \*/
position: absolute;
top: 0;
right: 0;
bottom: 0;
left: 0;
z-index: 100;
/\* Box-model \*/
display: block;
float: right;
width: 100px;
height: 100px;
/\* Typography \*/
font: normal 13px "Helvetica Neue", sans-serif;
line-height: 1.5;
color: #333;
text-align: center;
/\* Visual \*/
background-color: #f5f5f5;
border: 1px solid #e5e5e5;
border-radius: 3px;
/\* Misc \*/
opacity: 1;
}

**2. 选择器**

如无必要，不得为 id、class 选择器添加类型选择器进行限定，在性能和维护性上，都有一定的影响。

/\* 推荐\*/
#error,
.danger-message {
font-color: #c00;
}
/\* 不推荐\*/
dialog#error,
p.danger-message {
font-color: #c00;
}

**3. 属性使用缩写**

在需要显示地设置所有值的情况下，应当尽量限制使用简写形式的属性声明。常见的滥用简写属性声明的情况如下：

padding
margin
font
background
border
border-radius

另外，对于#aabbcc形式的颜色值也可简化为#abc，这样精简代码同时又能提高用户的阅读体验。

**4. a标签伪类书写顺序**

<a>标签的这四个伪类的特殊性是一样的，所以当某个链接处于的状态同时激活多个伪类时，那么伪类的书写顺序就起到了关键作用，从而影响最终的显示效果。

link->visited->hover->active

**5. 媒体查询（Media query）的位置**

媒体查询放在相关规则后面或者统一放在文档底部，若统一放在文档底部的时候，顺序应按照正常样式进行排布且注释统一。

**JS规范**

**1. 文件命名**

文件夹和文件名的命名应该能代表代码功能，与后端一致为佳。

**2. 语言规范**

**2.1 变量**

声明变量必须加上 var 关键字.

当你没有写 var, 变量就会暴露在全局上下文中, 这样很可能会和现有变量冲突. 另外, 如果没有加上, 很难明确该变量的作用域是什么, 变量也很可能像在局部作用域中, 很轻易地泄漏到 Document 或者 Window 中, 所以务必用 var 去声明变量.

**2.2 分号**

总是使用分号

如果仅依靠语句间的隐式分隔, 有时会很麻烦. 你自己更能清楚哪里是语句的起止,而且有些情况下，漏掉分号会很危险。

**2.3 块内函数声明**

不要在块内声明一个函数，不推荐一下写法：

if (x) {
function foo() {}
}

虽然很多 JS 引擎都支持块内声明函数, 但它不属于 ECMAScript 规范 (见 [ECMA-262](https://link.jianshu.com?t=http%3A%2F%2Fwww.ecma-international.org%2Fpublications%2Fstandards%2FEcma-262.htm), 第13和14条). 各个浏览器糟糕的实现相互不兼容, 有些也与未来 ECMAScript 草案相违背. ECMAScript 只允许在脚本的根语句或函数中声明函数. 如果确实需要在块中定义函数, 建议使用函数表达式来初始化变量:

if (x) {
var foo = function() {}
}

**2.4 闭包**

比较好的介绍闭包原理的[文档](https://link.jianshu.com?t=http%3A%2F%2Fjibbering.com%2Ffaq%2Ffaq_notes%2Fclosures.html).

有一点需要牢记, 闭包保留了一个指向它封闭作用域的指针, 所以, 在给 DOM 元素附加闭包时, 很可能会产生循环引用, 进一步导致内存泄漏. 比如下面的代码:

function foo(element, a, b) {
element.onclick = function() { /\* uses a and b \*/ };
}

这里, 即使没有使用 element, 闭包也保留了 element, a 和 b 的引用, . 由于 element 也保留了对闭包的引用, 这就产生了循环引用, 这就不能被 GC 回收.

这种情况下, 可将代码重构为:

function foo(element, a, b) {
element.onclick = bar(a, b);
}
function bar(a, b) {
return function() { /\* uses a and b \*/ }
}

**2.5 this**

仅在对象构造器, 方法, 闭包中使用.

this 的语义很特别. 有时它引用一个全局对象(大多数情况下), 调用者的作用域(使用eval时), DOM 树中的节点(添加事件处理函数时), 新创建的对象(使用一个构造器), 或者其他对象(如果函数被call() 或 apply())。

**2.6 for-in 循环**

最好只用于 object/map/hash 的遍历

对 Array 用 for-in 循环有时会出错. 因为它并不是从 0 到 length - 1 进行遍历, 而是所有出现在对象及其原型链的键值.

例如：给原型添加属性之后，默认情况下枚举，最后输出1234513

function getNewArrayTwo(){
var array=[1,2,3,4,5 ];
Array.prototype.age=13;
var result=[];
for(var i in array){
result.push(array[i]);
}
alert(result.join(''));
}

所以建议不要对数组执行for in循环，事实上，在高性能javascript这本书中,也强调了for in循环的不好,因为它总是会访问该对象的原型,看下原型上是否有属性,这在无意中就给遍历增加了额外的压力。

**3. 编码风格**

**3.1 明确作用域**

任何时候都要明确作用域 – 提高可移植性和清晰度. 例如, 不要依赖于作用域链中的 window 对象.

可能在其他应用中, 你函数中的 window 不是指之前的那个窗口对象。

**3.2 代码格式化**

数组和对象的初始化,如果初始值不是很长, 就保持写在单行上:

var arr = [1, 2, 3]; // No space after [ or before ].
var obj = {a: 1, b: 2, c: 3}; // No space after { or before }.

初始值占用多行时, 缩进2个空格.

// Object initializer.
var inset = {
top: 10,
right: 20,
bottom: 15,
left: 12
};
// Array initializer.
this.rows\_ = [
'"Slartibartfast" <fjordmaster@magrathea.com>',
'"Zaphod Beeblebrox" <theprez@universe.gov>',
'"Ford Prefect" <ford@theguide.com>',
'"Arthur Dent" <has.no.tea@gmail.com>',
'"Marvin the Paranoid Android" <marv@googlemail.com>',
'the.mice@magrathea.com'
];

**3.3 引号的使用**

单引号 (‘) 优于双引号 (“).

当你创建一个包含 HTML 代码的字符串时就知道它的好处了。

**3.5 过长的单行予以换行**

换行应选择在操作符和标点符号之后。

if (oUser.nAge < 30
&& oUser.bIsChecked === true
|| oUser.sName === 'admin') {
// code
}

**3.6 用数组和对象字面量来代替数组和对象构造器**

数组构造器很容易让人在它的参数上犯错，如new Array(1, 2, 3)会生成数组[1, 2, 3]，但

new Array(3)只会生成长度为3的空数组。

对象构造器不会有类似的问题，但是为了可读性和统一性，我们应该使用对象字面量。

// 推荐
var a1 = [x1, x2, x3];
var a2 = [x1];
var o = {
a: 0,
'strange key': 3
};
// 不推荐
var a1 = new Array(x1, x2, x3);
var a2 = new Array(x1);
var o = new Object();
o.a = 0;
o['strange key'] = 3;

3.6 循环的使用

在循环中，尽量使用变量先获取到相关数值，在放入循环中进行判断，否则非常影响性能。

// 推荐
for (var i = 0, max = $('#id').children().length; i < max; i++) {
// code
}
// 不推荐
for (var i = 0; i < $('#id').children().length; i++) {
// code
}

**3.7 注释**

3.71 函数注释

/\*\*
\* 简述
\*
\* 功能详细描述
\*
\* @param <String> arg1 参数1
\* @param <Number> arg2 参数2，默认为0
\* @return <Boolean> 判断xxx是否成功
\*/
function fooFunction (arg1, arg2) {
// code
}

3.71 语句注释

单行注释：

单独一行：//(双斜线)与注释文字之间保留一个空格；

在代码后面添加注释：//(双斜线)与代码之间保留一个空格，并且//(双斜线)与注释文字之间保留一个空格；

//(双斜线)与代码之间保留一个空格。

// 调用了一个函数；1)单独在一行
setTitle();
var maxCount = 10; // 设置最大量；2)在代码后面注释
// setName(); // 3)注释代码

**Vue规范**

**1. Vue属性书写顺序**

export default {
mixins,
data,
props,
store，
computed，
route,
created，
ready， // => 生命周期顺序不赘述
event,
watch,
components,
methods
}

**2. 组件**

2.1 **组件以驼峰命名**

2.2 **Vue组件的书写顺序**

2.3 **组件引用**

import myComponentsA from './myComponentsA.vue'
import myComponentsB from './myComponentsB.vue'
import myComponentsC from './myComponentsC.vue'
import myComponentsD from './myComponentsD.vue'
export default {
components: {
myComponentsA,
myComponentsB,
myComponentsC,
myComponentsD,
}
}

**3. 事件**