TES 教育
前端代码规范

版本号	撰稿	审定	更新日期	备注
0.0.1	胡亮		2017.10.26	草案

一、通用

1. 【强制】缩进
   用两个空格来代替制表符（tab）作为缩进。

2. 【强制】大小写
   以下都应该用小写：
   HTML 元素名称，属性，属性值（除非 text/CDATA）；
   CSS 选择器，属性，属性值。

一、HTML

1. 【强制】声明文档模式为 HTML5 doctype
   在每个 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。
   正例：

```
<!DOCTYPE HTML>
```

2. 【强制】指定文档的语言属性
   根据 HTML5 规范，强烈建议为 html 根元素指定 lang 属性，从而为文档设置正确的语言。
   正例：

```
<html lang="zh-CN"></html>
```

3. 【强制】字符编码
   通过明确声明字符编码，能够确保浏览器快速并容易的判断页面内容的渲染方式。这样做的好处是，可以避免在 HTML 中使用字符实体标记（character entity），从而全部与文档编码一致（一般采用 UTF-8 编码）。
   正例：

```
<meta charset="UTF-8">
```

4. 【强制】IE 兼容模式
   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。
正例：

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

5. 【强制】引入 CSS 和 JavaScript 文件
根据 HTML5 规范，在引入 CSS 和 JavaScript 文件时一般不需要指定 type 属性，除非不是 CSS 或 JavaScript，因为 text/css 和 text/javascript 分别是它们的默认值。
正例：

```
<link rel="stylesheet" href="code-guide.css">
<script src="code-guide.js"></script>
```

6. 【推荐】使用语义化标签
尽量遵循 HTML 标准和语义，但是不要以牺牲实用性为代价。根据元素被创造出来时的初始意义来使用它。例如，使用 h1 元素创建标题，p 元素创建段落，a 元素创建链接等等。正确的使用 HTML 元素对于可访问性、可重用性以及编码效率都很重要。

7. 【推荐】减少标签数量
任何时候都要尽量使用最少的标签，避免多余的父元素，保持最小的复杂度。
反例：

```
<span class="avatar">
   <img src="...">
</span>
```

正例：

```
<img class="avatar" src="...">
```

8. 【强制】严格嵌套约束规则
a 元素里不可以嵌套交互式元素(<a>、<button>、<select>等)
<p>里面不可以嵌套 、<h1>~<h6>、<p>、<ul>/<ol>/<li>、<dl>/<dt>/<dd>、<form>等
反例：

```
<p><div></div></p>
```

正例：

```
<p><span><span></p>
```

9. 【推荐】结构、表现、行为三者分离
尽量在文档和模板中只包含结构性的 HTML；而将所有表现代码，移入样式表中；将所有动作行为，移入脚本之中，确保相互耦合最小化。

10. 【推荐】属性顺序
HTML 属性应当按照以下给出的顺序依次排列，确保代码的易读性。
class
id, name

data-*
src, for, type, href, value
title, alt
role, aria-*
class 用于标识高度可复用组件，因此应该排在首位。id 用于标识具体组件，应当谨慎使用，因此排在第二位。

11.【推荐】布尔（boolean）型属性
布尔型属性可以在声明时不赋值。XHTML 规范要求为其赋值，但是 HTML5 规范不需要。
正例：

```
<input type="checkbox" value="1" checked disabled>
```

12.【推荐】避免一行代码过长
使用 HTML 编辑器，左右滚动代码是不方便的。每行代码尽量少于 80 个字符。

13.【推荐】HTML 注释
注释写在<!--和-->中，比较长的评论可以在<!--和-->中分行写，长评论第一个字符缩进两个空格，更易于阅读。
正例：

```
<!-- 这是注释 -->
<!--
  这是一个较长评论。这是一个较长评论。这是一个较长评论。
  这是一个较长评论。这是一个较长评论。这是一个较长评论。
-->
```

14.【推荐】alt 标签不为空
图片 alt 属性不要留空，用描述性文字代替最佳：

15.【推荐】a 标签的 title 属性不为空
锚文字是告诉搜索引擎被链接页面主题内容的重要依据之一，也可为用户做信息提示；
正例：

```
<a title="返回首页 " href="http://es.tes-sys.com/">首页</a>
```

16.【推荐】空行和缩进
不要无缘无故添加空行。
为每个逻辑功能块添加空行，这样更易于阅读。
比较短的代码间不要使用不必要的空行和缩进。
用两个空格来代替制表符（tab），这是唯一能保证在所有环境下获得一致展现的方法。

17.【强制】元素定义
使用小写元素名，关闭所有 HTML 元素。

18.【强制】属性定义
对于属性的定义，确保全部使用双引号，绝不要使用单引号。

反例：

```
<div class='bottombar-item' data-bottombar-item='teaching' title='教学过程'></div>
```

正例：

```
<div class="bottombar-item" data-bottombar-item="teaching" title="教学过程"></div>
```

19.【推荐】其他

不要在自闭合元素的尾部添加斜线，HTML5 规范中明确说明这是可选的。

不要省略可选的结束标签（例如，
或</body>）。

二、CSS

1. 【强制】声明样式表的字符编码
   在每个 CSS 文档的第一行添加字符编码的声明。
   正例：

```
@charset "UTF-8";
```

2. 代码风格
   1) 【强制】不要使用 ID 选择器；
   2) 【强制】属性选择器中的值必须用双引号包围；
   3) 【强制】不要在 URI 值（url()）中使用引号；
   4) 【推荐】在一个规则声明中应用了多个选择器时，每个选择器独占一行；
   5) 【强制】在规则声明的左大括号 { 前加上一个空格；
   6) 【强制】在属性的冒号 : 后面加上一个空格，前面不加空格；
   7) 【强制】规则声明的右大括号 } 独占一行；
   8) 【推荐】规则声明之间用空行分隔开；
   9) 【强制】所有声明语句都应当以分号结尾；
   10) 【强制】>、+、~ 选择器的两边各保留一个空格；
   11) 【强制】对于以逗号分隔的属性值，每个逗号后面都应该插入一个空格；
   12) 【推荐】不要在 rgb()、rgba()、hsl()、hsla()或 rect()值的内部的逗号后面插入空格。
   说明：这样利于从多个属性值（既加逗号也加空格）中区分多个颜色值（只加逗号，不加空格）。
   反例：

```
.selector{
   border :2px solid white;
   border-radius:50%}
.selector1,.selector2 {
   // ...
}
#selector {
   // ...
}
```

   正例：

```
.selector {
   border: 2px solid white;
   border-radius: 50%;
}
.selector1,
.selector2 {
   // ...
}
```

3. class 命名
   1) 【强制】class 名称中只能出现小写字符和破折号（不是下划线，也不是驼峰命名法）。破折

号应当用于相关 class 的命名（类似于命名空间）（例如，.btn 和.btn-danger）。
2) 【强制】避免过度任意的简写。.btn 代表 button，但是.s 不能表达任何意思。
3) 【强制】class 名称应当尽可能短，并且意义明确。
4) 【强制】使用有意义的名称。使用有组织或目的明确的名称，不要使用表现形式的名称。
5) 【强制】基于最近的父 class 或基本 class 作为新 class 的前缀。
6) 【参考】使用.js-*来标识行为（与样式相对），并且不要将这些 class 包含到 CSS 文件中。

反例：

```
.t { ... }
.red { ... }
.header { ... }
```

正例：

```
.tweet { ... }
.important { ... }
.tweet-header { ... }
```

4. 选择器
   1) 【强制】对于通用元素使用 class，这样利于渲染性能的优化。
   2) 【强制】对于经常出现的组件，避免使用属性选择器（例如，[class^="..."]）。
   3) 【强制】选择器要尽可能短，并且尽量限制组成选择器的元素个数，建议不要超过 3。
   4) 【强制】只有在必要的时候才将 class 限制在最近的父元素内（也就是后代选择器）。

   反例：

```
span { ... }
.page-container #stream .stream-item .tweet .tweet-header .username { ... }
.green { ... }
```

   正例：

```
.avatar { ... }
.tweet-header .username { ... }
.tweet .avatar { ... }
```

5. 【推荐】声明顺序

   相关的属性声明应当归为一组，并按照下面的顺序排列：

   Formatting Model
   Box Model
   Typographic
   Visual

   由于定位（positioning）可以从正常的文档流中移除元素，并且还能覆盖盒模型（box model）相关的样式，因此排在首位。盒模型排在第二位，因为它决定了组件的尺寸和位置。其他属性只是影响组件的内部或者是不影响前两组属性，因此排在后面。

   说明：

   Formatting Model 相关属性包括：position / top / float / display / overflow 等
   Box Model 相关属性包括：border / margin / padding / width / height 等
   Typographic 相关属性包括：font / line-height / text-align / word-wrap 等
   Visual 相关属性包括：background / color / transition / list-style 等

另外，如果包含 content 属性，应放在最前面。

6. 【推荐】单行规则声明

对于只包含一条声明的样式，为了易读性和便于快速编辑，建议将语句放在同一行。对于带有多条声明的样式，还是应当将声明分为多行。

说明：

这样做的关键因素是为了错误检测，例如，CSS 校验器指出在 183 行有语法错误。如果是单行单条声明，你就不会忽略这个错误；如果是单行多条声明的话，你就要仔细分析避免漏掉错误了。

7. 【参考】简写形式的属性声明

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有：margin、padding、font、background、border、border-radius 等。过度使用简写形式的属性声明会导致代码混乱，并且会对属性值带来不必要的覆盖从而引起意外的副作用。

在 MDN 上一篇非常好的关于 shorthand properties[1]的文章，建议熟悉简写属性声明及其行为。

8. 【强制】不要使用@import

与<link>标签相比，@import 指令要慢很多，不光增加了额外的请求次数，还会导致不可预料的问题。

9. 【强制】媒体查询（Media query）的位置

将媒体查询放在尽可能相关规则的附近。不要将他们打包放在一个单一样式文件中或者放在文档底部。如果你把他们分开了，将来只会被大家遗忘。

10. 【推荐】注释

代码是由人编写并维护的。

请确保你的代码能够自描述、注释良好并且易于他人理解。

好的代码注释能够传达上下文关系和代码目的。

不要简单地重申组件或 class 名称。

对于较长的注释，务必书写完整的句子；对于一般性注解，可以书写简洁的短语。

建议注释独占一行，避免行末注释。

11. 【推荐】代码组织

1) 以组件为单位组织代码段。

2) 使用一致的空白符将代码分隔成块，这样利于扫描较大的文档。

3) 如果使用了多个 CSS 文件，将其按照组件而非页面的形式分拆，因为页面会被重组，而组件只会被移动。

12. 【推荐】编辑器配置

将你的编辑器按照下面的配置进行设置，以避免常见的代码不一致和差异：

用两个空格代替制表符（即用空格代表 tab 符）。

保存文件时，删除尾部的空白符。

设置文件编码为 UTF-8。

在文件结尾添加一个空白行。

[1] https://developer.mozilla.org/zh-CN/docs/Web/CSS/Shorthand_properties

13.【推荐】格式

1) 对于属性值或颜色参数，省略小于 1 的小数前面的 0（例如，.5 代替 0.5；-.5px 代替-0.5px）。

2) 十六进制值应该全部小写，例如，#fff。在扫描文档时，小写字符易于分辨，因为他们的形式更易于区分。

3) 尽量使用简写形式的十六进制值，例如，用#fff 代替#ffffff。

4) 避免为 0 值指定单位，例如，用 margin: 0;代替 margin: 0px;

14.【推荐】阅读

https://cssguidelin.es/

三、JavaScript

1. 对象

1) 【强制】使用直接量创建对象。

反例：

```
var item = new Object();
```

正例：

```
var item = {};
```

2) 【强制】不要使用保留字作为键名。

3) 【强制】使用同义词替换需要使用的保留字。

反例：

```
var superman = {
    class: 'alien'
};
var superman = {
    klass: 'alien'
};
```

正例：

```
var superman = {
    type: 'alien'
};
```

2. 数组

1) 【强制】使用直接量创建数组。

反例：

```
var items = new Array();
```

正例：

```
var items = [];
```

2) 【推荐】向数组增加元素时使用 Array#push 来替代直接赋值。

反例：

```
var stack = [];
stack[stack.length] = 'abracadabra';
```

正例：

```
var stack = [];
stack.push('abracadabra');
```

3) 【推荐】当你需要拷贝数组时，使用 Array#slice。

反例：

```
var len = items.length;
var itemsCopy = [];
var i;

for (i = 0; i < len; i++) {
  itemsCopy[i] = items[i];
}
```

正例：

```
itemsCopy = items.slice();
```

4) 【推荐】使用 Array#slice 将类数组对象转换成数组。

正例：

```
var argsArr = Array.prototype.slice.call(arguments);
```

3. 字符串

1) 【推荐】使用单引号"包裹字符串。

2) 【推荐】超过 100 个字符的字符串应该使用连接符写成多行。

说明：

若过度使用，通过连接符连接的长字符串可能会影响性能。

3) 【推荐】程序化生成的字符串使用 Array#join 连接而不是使用连接符。

4. 函数

1) 【强制】永远不要在一个非函数代码块（if、while 等）中声明一个函数，浏览器允许你这么做，但它们的解析表现不一致，正确的做法是：在块外定义一个变量，然后将函数赋值给它。

说明：

ECMA-262 把 块 定义为一组语句，函数声明不是语句。

反例：

```
if (currentUser) {
    function test() {
        console.log('Hello World!');
    }
}
```

正例：

```
var test;
if (currentUser) {
    test = function test() {
        console.log('Hello World!');
    };
}
```

2) 【强制】永远不要把参数命名为 arguments。这将取代函数作用域内的 arguments 对象。

反例：

```
function nope(name, options, arguments) {
    // ...stuff...
}
```

正例：

```
function yup(name, options, args) {
    // ...stuff...
}
```

5. 属性

1) 【强制】使用 . 来访问对象的属性。

说明：

ECMA-262 把 块 定义为一组语句，函数声明不是语句。

反例：

```
var person = {
    age: 28
};
var age = person['age'];
```

正例：

```
var age = person.age;
```

2) 【强制】当通过变量访问属性时使用中括号[]。

正例：

```
var person = {
    age: 28
};

function getProp(prop) {
    return person[prop];
}

var age = getProp('age');
```

6. 变量

1) 【强制】总是使用 var 来声明变量，避免污染全局命名空间。

反例：

```
superPower = new SuperPower();
```

正例：

```
var superPower = new SuperPower();
```

2) 【推荐】在作用域顶部声明变量，避免变量声明提升相关的问题。

3) 【推荐】使用 var 声明每一个变量。这样做的好处是增加新变量将变的更加容易，而且你永远不用再担心调换错;跟,。

4) 【推荐】最后再声明未赋值的变量。当你需要引用前面的变量赋值时这将变的很有用。

反例：

```
var i, len, dragonball,
        items = getItems(),
        goSportsTeam = true;

var i;
var items = getItems();
var dragonball;
var goSportsTeam = true;
var len;
```

正例：

```
var items = getItems();
var goSportsTeam = true;
var dragonball;
var length;
var i;
```

其他

文件命名
文件名中只可由英文字母 a~z、排序数字 0~9 或间隔符-组成，禁止包含特殊符号，比如空格、$等
文件名区分大小写，统一使用小写字母
为更好的表达语义，文件名使用英文名词命名，或英文简写。

图片命名

图片后缀命名一律小写。
使用间隔符-进行连接。*一般背景图片以 bg-开头，按钮图片以 btn-开头，图标图片以 icon-开头，聚合图以 spr-开头，后跟英文单词，不推荐使用汉语拼音，如果名称过长，适当使用缩写
bg-body.jpg spr-home.png　btn-submit.png　icon-game.png