前端代码规范：
一般规范

1. 使用 2 个空格符缩进

建议使用 2 个空格符进行缩进，不要使用 tab，因为当在不同的环境或者说是编辑器当中，只有使用 2 个空格符才能保障在所有环境下具有一致的表现。

2. 文件/资源命名

我们的文件命名需要遵循统一的命名规定，以可读性而言采用（-）减号来分隔文件名。确保文件名总是以字母开头而不是数字，且尽量都使用小写字母，因为在大小写敏感的操作系统中有可能会参数异常且难以察觉
不推荐：

```
MyScript.js

myScript.js

100Script.js

my-file-min.css
```

推荐：

```
my-script.js

my-camel-case-name.js

my-file.min.css
```

3. 变量/函数命名规范

变量的命名尽量采用匈牙利命名法，驼峰式书写格式，其中，函数和变量以小写字母开头，类（class）则用大写字母开头，尽量少用缩写，即便用也尽量采用常见的一些缩写（如：num = number，btn = button），常量则采用全部字母大写的书写的方式。更多内容可以参见 google 开源项目风格指南。对于 css 的书写，尽量赋予选择器以意义，不要出现没有意义的颜色、数字，应把他们与相应的意思对应起来。
不推荐：

```
var n; //不明白 n 的意义

.text-red { color: red; } //红色表示什么意思
```

推荐：

```
var number;
var num;
function btnChangeColor(color){...}
.text-danger { color: red; }
```

4. 注释

编写代码时需要适当的注释用以说明文件的作者、函数的功能、甚至是某些变量值的含义等等。良好的注释可以提高代码的可读性，也便于代码维护和团队协作。我们主要使用 js 来编写，请仔细查看 http://usejsdoc.org/ 来编写符合规范的注释。
例如：

```
/**
 * Represents a book. 函数描述
 * @param {string} title - The title of the book. 参数 title
 * @param {string} author - The author of the book. 参数 author
 */
function Book(title, author) {
}
```

HTML/CSS 编写规范

1. 协议引入

当引入图片或者其他媒体文件时，url 所指向的具体路径中不需要指定协议，除非不是 http 或 https 协议。
不推荐：

```
<script src="http://cdn.com/jquery.min.js"></script>
```

推荐：

```
<script src="//cdn.com/jquery.min.js"></script>
```

不推荐：

```
.example {
  background: url("http://example.com/image/bg.jpg");
```

```
}
```

推荐：

```
.example {
  background: url("//example.com/image/bg.jpg");
}
```

### 2. 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）。

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

### 3. 引入 CSS 和 JavaScript 文件

根据 HTML5 规范，在引入 css 和 JavaScript 文件时一般不需要制定 type 属性，因为 text/css 和 text/javascript 分别是他们的默认值。

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

### 4. 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
title, alt
aria-*, role
```

例如：

```
<a class=".." id=".." data-modal="toggle" href="#">
  example link
</a>
```

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。
不推荐：

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

推荐：

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

CSS 规范

- 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。
- 为选择器分组时，将单独的选择器单独放在一行。
- 为了代码的易读性，在每个声明块的左花括号前添加一个空格。
- 声明块的右花括号应当单独成行。
- 每条声明语句的 : 后应该插入一个空格。
- 为了获得更准确的错误报告，每条声明都应该独占一行。
- 所有声明语句都应当以分号结尾。最后一条声明语句后面的分号是可选的，但是，如果省略这个分号，你的代码可能更易出错。
- 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（例如，box-shadow）。
- 不要在 rgb()、rgba()、hsl()、hsla() 或 rect() 值的内部的逗号后面插入空格。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）。
- 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）。
- 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。
- 尽量使用简写形式的十六进制值，例如，用 #fff 代替 #ffffff。
- 为选择器中的属性添加双引号，例如，input[type=” text”]。只有在某些情况下是可选的，但是，为了代码的一致性，建议都加上双引号。
- 避免为 0 值指定单位，例如，用 margin: 0; 代替 margin: 0px;。

```
/* Bad CSS */
.selector, .selector-secondary, .selector[type=text] {
  padding:15px;
  margin:0px 0px 15px;
  background-color:rgba(0, 0, 0, 0.5);
  box-shadow:0px 1px 2px #CCC,inset 0 1px 0 #FFFFFF
}

/* Good CSS */
.selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

**1. 声明顺序**

相关属性的声明应该归为一组，能使用简写尽量使用简写，并按照下面的顺序排列：

1. Position
2. Box model
3. Typographic
4. Visual

   完整的属性列表及其排列顺序请参考 Recess

```
.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. 媒体查询（Media query）的位置**

因为我们的项目要做响应式，所以肯定会用到媒体查询。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。如果将他们分开，容易遗忘。
推荐：

```
.element { ... }
.element-avatar { ... }
.element-selected { ... }
```

```
@media (min-width: 480px) {
  .element { ...}
  .element-avatar { ... }
  .element-selected { ... }
}
```

### 3. 注释

尽可能描述出该声明的作用和意图

```
/* Wrapping element for .modal-title and .modal-close */
.modal-header {
  ...
}
```

### 4. class 命名

- class 名称中只能出现小写字符和破折号（dashe）（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）。
- 避免过度任意的简写。.btn 代表 button，但是 .s 不能表达任何意思。
- class 名称应当尽可能短，并且意义明确。
- 使用有意义的名称。使用有组织的或目的明确的名称，不要使用表现形式（presentational）的名称。
- 基于最近的父 class 或基本（base） class 作为新 class 的前缀。

```
/* Good example */
.tweet { ... }
.important { ... }
.tweet-header { ... }
```

### 5. 选择器

- 对于通用元素使用 class ，这样利于渲染性能的优化。
- 对于经常出现的组件，避免使用属性选择器（例如，[class^="…"]）。浏览器的性能会受到这些因素的影响。
- 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 ，太多选择器会造成性能查询上的问题。

```
/* Bad example */
```

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


/* Good example */
.avatar { ... }
.tweet-header .username { ... }
.tweet .avatar { ... }
```

### 6. 代码组织

- 以组件为单位组织代码段。
- 制定一致的注释规范。
- 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。
- 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。

### 7. id 还是 class

- class 应该应用于概念上相似的元素，这些元素可以出现在同一页面上的多个位置，而 id 应该应用于不同的唯一的元素。
- 只有在目标元素非常独特，绝不会对网站上其他地方别的东西使用这个名称时，才会使用 id.
- 如果你认为以后可能需要相似的元素，就使用 class。

**后端（Java）代码规范文档：**http://www.hawstein.com/posts/google-java-style.html
**关键点：**
**1 命名**
类名、接口名使用 Pascal 命名法，变量、方法名使用 CamelCase 命名法，常量名使用全大写字母加下划线的方式命名。
**2 缩进**
缩进大小为 4 个空格。
**3 变量声明**
所有声明的变量必须初始化。
**4 使用大括号（即使是可选的）**
大括号与 if, else, for, do, while 语句一起使用，即使只有一条语句（或是空），也应该把大括号写上。
**5 大括号换行**
- 左大括号前不换行
- 左大括号后换行
- 右大括号前换行

- 如果右大括号是一个语句、函数体或类的终止，则右大括号后换行；否则不换行。例如，如果右大括号后面是 else 或逗号，则不换行。