诞客网络科技有限公司前端规范文档

Version 1.0.0

版本修订纪录

Date	Version	Description	Author
2013-08-19	1.0.0	初稿	周文康
2013-11-01	1.0.1	加入 css 规范	周文康

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

目录

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

第一部分 编程风格

1.基本的格式化

1.1 缩进层级

推荐使用 4 个字符为一个缩进层级。

1.2 语句结尾

推荐不要省略分号。

1.3 行的长度

推荐行的长度限定在 80 个字符。

1.4 换行

通常我们在运算法后换行，下一行增加两个层级的缩进。

1.5 空行

1. 方法之间。
2. 方法中局部变量和第一行语句之间。
3. 多行或者单行注释之前。
4. 方法中逻辑代码块之间以提升代码的可读性。

1.6 命名

推荐使用驼峰式大小写命名法。

1.6.1 变量和函数

变量：小驼峰式大小写，前缀是名词。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

函数：小驼峰式大小写，前缀是动词。

1.6.2 常量

大写，用下划线分隔单词。

1.6.3 构造函数

大驼峰式大小写。

1.7 直接量

javascript 中的原始类型：字符串、数字、布尔型、null、undefined。

1.7.1 字符串

建议使用双引号来括住字符串。

1.7.2 数字

小数需书写完整（小数部分和整数部分）。
担心 0 打头的为 8 进制，0x 为 16 进制。

1.7.3 null

特殊值 null 除了下述情况下应当避免使用。

1. 用来初始化一个变量,这个变量可能被赋值为一个对象。
2. 用来和一个已经初始化的变量比较,这个变量可以是也可以不是一个对象。
3. 当函数的参数期望是对象时,被用作参数传入。
4. 当函数的返回值期望是对象时,被用作返回值传出。

还有一些场景不应当使用 null。

1. 不要使用 null 来检测是否传入了某个参数。
2. 不要使用 null 来检测一个未初始化的变量。

1.7.4 undefined

容易和”null”混淆。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

不管是值是 undefined 的变量还是未声明的变量，typeof 运算符结果都是“undefined“
当把变量初始值赋为 null，typeof null 返回”object”，这样就可以和 undefined 区分开。

1.7.5 对象直接量

当定义对象直接量时，常常在第一行包含左花括号，每一个属性的名值对都独占一行，并保持一个缩进，最后右花括号也独占一行。

1.7.6 数组直接量

不赞成显式地使用 Array 构造函数来创建数组。可以使用两个方括号将数组元素括起来代替构造函数的方式。

2.注释

2.1 单行注释

单行注释以两个斜线开始。应当用来说明一行代码或者一组相关的代码。单行注释可能有三种使用方法。

1. 独占一行的注释，用来解释下一行代码。
2. 在代码行的尾部注释，用来解释它之前的代码。
3. 多行，用来注释掉一个代码块。

2.2 多行注释

多行注释应当在代码需要更多文字去解释的时候使用。每个多行注释都在至少有如下三行。

1. 首行仅仅包括/*注释开始。该行不应当有其他文字。
2. 接下来的行以*开头并保持左对齐。这些行可以有文字描述。
3. 最后一行以*/开头并同先前行保持对齐。也不应当有其他文字。

多行注释的首行应当保持同它描述代码的相同层次的缩进。后续的每行应当有同样层次的缩进并附加一个空格（为了适当保持*字符的对齐）。每一个多行代码之前应当预留一个空行。

2.3 使用注释

添加注释的一般原则是，在需要让代码变得更清晰时添加注释。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

2.3.1 难于理解的代码

难于理解的代码通常都应当就加注释。

2.3.2 可能被误认为错误的代码

另一个适合添加注释的好时机是当代码看上去有错误时。

2.3.3 浏览器特性 hack

Javascript 程序员常常会编写一些低效的、不雅的、彻头彻尾的肮脏代码，用来让低级浏览器正常工作。实际上这种情形是一种特殊的“可能被误认为错误的代码”，所以确保碰到这种情形要有相应的注释。

3.语句和表达式

不论块语句包含多行代码还是单行代码，都应当总是使用花括号，包括：

1. if
2. for
3. while
4. do...while...
5. try...catch...finally

3.1 花括号的对齐方式

将左括号放置在块语句中第一句代码的末尾。

3.2 块语句间隔

在括左圆括号之前和右圆括号之后各添加一个空格。

3.3 switch 语句

switch 语句中可以使用任意类型值，任何表达式都可合法地用于 case 从句。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

3.3.1 缩进

1. 每条 case 语句相对于 switch 关键字都缩进一个层级。
2. 从第二条 case 语句开始，每条 case 语句前后各有一个空行。

3.3.2 case 语句的“连续执行”

当你的代码出现 case 语句的连续执行时，建议添加注释（/*fall through*/）。

3.3.3 defalut

更倾向于在没有默认行为且写了注释的情况下省略 default。

3.4 with 语句

在严格模式中，with 语句是被明确禁止的，避免使用 with 语句。

3.5 for 循环

避免使用 continue，但也没有理由完全禁止使用，它的使用应当根据代码可读性来决定。

3.6 for-in 循环

1. for-in 循环式用来遍历对象属性的，它不仅遍历对象的实例属性，同样遍历从原型继承来的属性，所以最好使用 hasOwnProperty()方法来为 for-in 循环过滤出实例属性。
2. 避免将 for-in 循环使用在数组成员上，因为 for-in 循环式无序的。

4.变量、函数和运算符

4.1 变量声明

1. 合并 var 语句。
2. 将局部变量的定义作为函数内第一条语句。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

4.2 函数声明

和变量一样，函数声明在解析阶段也会被 Javascript 引擎提前，但推荐的写法还是先申明函数然后使用函数。

4.3 函数调用间隔

一般情况下，对于函数调用写法推荐的风格是，在函数名和左括号之间没有空格。这样做为了将它和块语句区分开来。

4.4 立即调用的函数

为了让立即执行的函数一眼被看出来，可以将函数用一对圆括号包裹起来。

4.5 严格模式

1. 避免使用全局的严格模式。
2. 如果你希望在多个函数中应用严格模式而不必写很多行"use strict"，可以使用立即执行函数。

4.6 相等

==和!==会强制转换类型，带来很多意想不到的后果，建议用===和!==替代。

4.6.1 eval()

eval()有着天生的安全漏洞问题，应该寻找其它的方法替代它（例如解析 JSON 时，可以用 json.parse()替代 eval()）。

4.6.2 原始包装类型

避免使用原始包装类型。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

第二部分 编程实践

5.UI 层的松耦合

5.1 什么是松耦合

确保一个组件的修改不会经常性的影响其它部分，这就达到了松耦合。本质上讲，每个组件需要保持足够瘦身来确保松耦合。组件知道的越少，就越有利于形成整个系统。

5.2 将 JavaScript 从 CSS 中抽离

避免使用 css 表达式。

5.3 将 CSS 从 JavaScript 中抽离

将 css 从 Javascript 中抽离意味着所有样式信息都应当保持在 css 中。当需要通过 Javascript 来修改元素样式的时候，最佳的方法是操作 css 的 className。

5.4 将 JavaScript 从 HTML 中抽离

1.避免将脚本嵌入 html 中运行。

```
1.    //不好的写法
2.    <button onclick="doSomething()"/>
3.    //好的写法
4.    btn.addEventListener("click",doSomething,flase)
```

2.最好将所有的 Javascript 代码都放入外置文件中。

5.5 将 HTML 从 JavaScript 中抽离

5.5.1 方法 1 ： 从服务器加载

将模板放在远程服务器上，使用 XMLHttpRequest 对象来获取外部标签。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

5.5.2 方法 2 ： 简单客户端模板

客户端模板是一些带“插槽”的标签片段，这些“插槽”会被 Javascript 程序替换为数据以保证模板的完整可用。

5.5.3 方法 3 ： 复杂客户端模板

参见诸如 Handlebars(http://handlebarsjs.com/)所提供的解决方案。

6.避免使用全局变量

全局作用域中声明的变量和函数都是 window 对象的属性。

6.1 全局变量带来的问题

6.1.1 命名冲突

当脚本中的全局变量和全局函数越来越多时，发生命名冲突额概率也随之增高。

6.1.2 代码的脆弱性

一个依赖于全局变量的函数即是深耦合于上下文环境之中。如果环境发生改变，函数很可能就失效了。所以定义函数时，最好尽可能多地将数据置于局部作用域内。在函数内定义的任何“东西”都应当采用这种写法。任何来自函数外部数据都应当以参数形式传递来。这样做可以将函数和其外部环境隔离开来，并且你的修改不会对程序其他部分造成影响。

6.1.3 难以测试

确保你的函数不会对全局变量有依赖，这将增强你的代码的可测试性。

6.2 意外的全局变量

不小心省略 var 语句意味着你在不知情的情况下创建了一个全局变量。最好的规则就是总是使用 var 来定义变量，哪怕是定义全局变量。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

6.2.1 避免意外的全局变量

使用严格模式。

6.3 单全局变量方式

依赖尽可能少的全局变量，即只创建一个全局变量。

6.3.1 命名空间

确保不重复的命名空间。

```
1.  var YourGlobal = {
2.      namespace: function (ns) {
3.          var parts = ns.split("."),
4.          object = this,
5.          i , len;
6.
7.          for (i=0, len=parts.length; i < len; i++) {
8.              if (!object[parts[i]]) {
9.                  object[parts[i]] = {};
10.             }
11.             object = object[parts[i]];
12.         }
13.
14.         return object;
15.     }
16. };
17. //调用
18. YourGlobal.namespace("book.maintainable");
```

6.3.2 模块

模块是一种通用的功能片段，它并没有创建新的全局变量或命名空间。相反，所有的这些代码都存放于一个表示执行一个任务或发布一个接口的单函数中。可以用一个名称来表示这个模块，同样这个模块可以依赖其他模块。

6.4 零全局变量方式

这种模式的使用场景有限。只要你的代码需要被其他代码依赖，就不能使用零全局变量这种

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

方式。如果你的代码需要在运行时不断扩展或者修改也不能使用。但是，如果你的脚本非常短，且不需要和其他代码产生交互，可以考虑这种方式实现。

```
1.              //一个零全局变量的例子
2.              (function (win {
3.                  "use strict"
4.
5.                  var doc = win.document;
6.                  //这里定义其他变量、代码
7.
8.               }(window));
```

7.事件处理

7.1 典型用法

event 对象。

7.2 规则 1：隔离应用逻辑

将应用逻辑从所有事件处理程序中抽离出来的做法是一种最佳实践。

```
1.      //好的写法-拆分应用逻辑
2.      var MyApplication = {
3.          handleClick: function (event) {
4.              this.showPopup(event);
5.          }
6.
7.          showPopup: function (event) {
8.              var popup = document.getElementById("popup");
9.              popup.style.left = event.clientX + "px";
10.             popup.style. top = event.clientY + "px";
11.             popup.className = "reveal";
12.         }
13.      };
14.      addListener(element, "click", function(event)) {
15.          MyApplication.handleClick(event);
16.          MyApplication.handleClick(event);
17.     }
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

7.3 规则 2：不要分发事件对象

最佳的办法是让事件处理程序使用 event 对象来处理事件，然后拿到所有需要的数据传给应用逻辑。

```
1.      //好的写法
2.      var MyApplication = {
3.          handleClick: function (event) {
4.              //阻止事件冒泡
5.              event.preventDefalut();
6.              event.stopPropagation();
7.              this.showPopup(event.clientX , event.clientY);
8.          }
9.
10.         showPopup: function (x, y) {
11.             var popup = document.getElementById("popup");
12.             popup.style.left = x + "px";
13.             popup.style. top = y + "px";
14.             popup.className = "reveal";
15.         }
16.      };
17.      addListener(element, "click", function(event)) {
18.          MyApplication.handleClick(event);
19.     }
```

8.避免“空比较”

8.1 检测原始值

Javascript 中有五种原始值：字符串、数字、布尔值、null、undefined。

1. 对于字符串，typeof 返回“String”。
2. 对于数字，typeof 返回“number”。
3. 对于布尔值，typeof 返回“boolean”。
4. 对于 undefined，typeof 返回“undefined”。
5. 对于 null，typeof 返回“object”。

注意两点：

1.未定义的变量和值位 undefined 的变量通过 typeof 都将返回“undefined”。

2.null，一般不用做检测语句，倘若真需要比较，应该使用===或!==。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

8.2 检测引用值

有这样几种内置的引用类型：Object、Array、Date、Error。使用 typeof 检测时，返回的都是“object”。
检测某个引用类型值得类型的最好方法是使用 instanceof 运算符。instanceof 的一个有趣特征是它不仅能检测构造这个对象的构造器，还检测原型链、自定义类型。

8.2.1 检测函数

检测函数最好的方法是使用 typeof，因为它可以跨帧使用。

8.2.2 检测数组

检测数组最好的方式是使用 isArray(),它也可以检测跨帧传递的值。

```
1.      function isArray (value) {
2.          if (typeof Array.isArray === "function") {
3.              return Array.isArray(value);
4.          } else {
5.              return Object.prototype.toString.call(value) === "[object Array]";
6.          }
7.      }
```

8.3 检测属性

判断属性是否存在的最好方法使用 in 运算符。in 运算符仅仅会简单的判断属性是否存在，而不会去读属性的值。

```
1.        if ("related" in object)
```

如果你只想检查实例对象的某个属性是否存在，则使用 hasOwnProperty()方法。

9.讲配置数据从代码中分离出来

9.1 什么是配置数据

1. URL。
2. 需要展现给用户的字符串。
3. 重复的值。

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

4. 设置（比如每页的配置项）。
5. 任何可能发生变更的值。

9.2 抽离配置数据

将配置数据从代码中抽离出来的第一步是将配置数据拿到外部，这样任何人都可以修改它们，而不会导致应用逻辑出错。同样，我们可以将整个 config 对象放到单独的文件中，这样对配置数据的修改可以完全和使用这些数据的代码隔离开来。

9.3 保存配置数据

1. 一种方式是采用 java 的属性文件来存放配置数据。

```
# 面向用户的消息
MSG_INVALID_VALUE = Invalid value

# URL
URL_INVALID = /errors/invalid.php
```

2. 第二种是 JSONP，是将 JSON 结构用一个函数（调用）包装起来。

```
myfunc({"MSG_INVALID_VALUE ":"Invalid value"," URL_INVALID ":"/errors/invalid.php "})
```

3. 最后一种是纯 Javascript，这种方式是将 JSON 对象赋值给一个变量，这个变量会序用到。

```
var config = ({"MSG_INVALID_VALUE ":"Invalid value"," URL_INVALID ":"/errors/invalid.php
                "})
```

10.抛出自定义错误

10.1 错误的本质

当某些非期望的事情发生时程序就引发一个错误。

10.2 在 JavaScript 中抛出错误

就牢记一件事情，如果没有通过 try-cctch 语句捕获，抛出任何值都将引发一个错误。针对所有浏览器，唯一不出差错的显示自定义的错误消息的方式就是用一个 Error 对象。

```
throw new Error("Something bad happended")
```


上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

10.3 抛出错误的好处

抛出自己的错误可以使用确切的文本供浏览器显示。除了行和列的号码，还可以包含任何你需要的有助于调试问题的信息。我推荐总是在错误消息中包含函数名称，以及函数失败的原因。

10.4 何时抛出错误

1. 一旦修复了一个很难调试的错误，尝试增加一两个自定义错误。当再次发生错误时，这将有助于更容易的解决问题。
2. 如果正在编写代码，思考一下：“我希望【某些事情】不会发生，如果发生，我的代码会一团糟糕”。这是，如果“某些事情”发生，就抛出一个错误。
3. 如果正在编写的代码别人也会使用，思考一下他们使用的方式，在特定的情况下抛出错误。

10.5 try-catch 语句

可能引发错误的代码放在try中，处理错误的代码放在catch块中。当然还可以增加一个finally块，finally块中的代码不管是否有错误发生，最后都会被执行。在某些情况下，finally块工作起来有点复杂。例如，如果try块中包含了一个return语句，实际上它必须等到finally块中的代码执行后才返回。
throw只应该在应用程序栈中最深的部分抛出，通常的异常捕获用try-catch更合适。千万不要将try-catch中的catch块留空，你应该总是写点什么来处理错误。

10.6 错误类型

ECMA-262指出7种错误类型。

1. Error ： 所有错误的基本烈性。
2. EvalError ： 通过eval()函数执行代码发生错误时抛出。
3. RangeError ： 一个数字超出它的边界是抛出。
4. ReferenceError ： 期望的对象不存在时抛出。
5. SyntaxError ： 给eval()函数传递代码中有语法错误时抛出。
6. TypeError ： 变量不是期望的类型时抛出。
7. URIError ： 给encodeURI()/encodeURIComponent()、decodeURI()或者decodeURIComponent()等函数传递格式非法的URI字符串是抛出。

上海市闸北区恒丰路600号7楼 邮编200070
Tel: +86-21-60540799

11.不是你的对象不要动

11.1 什么是你的

请牢记，如果你的代码没有创建这些对象，不要修改它们，包括：

1. 原生对象（例如：Object、Array）。
2. DOM 对象（例如：document）。
3. BOM 对象（例如：window）。
4. 类库的对象。

11.2 原则

11.2.1 不覆盖方法

11.2.2 不新增方法

11.2.3 不删除方法

11.3 更好的途径

在 javascript 之外，最受欢迎的对象扩充的形式是继承。但继承仍然有一些很大的限制。首先，不能从 DOM 或 BOM 对象继承。其次，由于数组索引和 length 属性之间错综复杂的关系，继承自 Array 是不能正常工作的。

11.3.1 基于对象的继承

ECMAScript5 的 Object.create()方法是实现这种继承的最简单的方式。

```
1.   var person = {
2.       name: "jason",
3.       sayName: function () {
4.           alert(this.name);
5.       }
6.   };
7.  var myPerson = Object.create("person");
8.  myPerson.sayName();   //弹出 jason"
```

重新定义 myPerson.sayName()会自动切断对 person.sayName()的访问。

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

```
1.      myPerson.sayName: function () {
2.          alert("blabla");
3.      }
4.      myPerson.sayName(); //弹出 blabla
5.      person.sayName();   //弹出 jason
```

Object.create()方法可以指定第二个参数，该参数对象中的属性和方法将添加到新对象中。

```
1.      var myPerson = Object.create(person, {
2.          name: {
3.              value: "Greg"
4.          }
5.      });
6.      myPerson.sayName(); //弹出 Greg
7.      person.sayName();   //弹出 jason
```

11.3.2 基于类型的继承

基于类型继承一般需要两个步骤：首先，原型继承；然后，构造器继承。

```
1.      function Person (name) {
2.          this.name;
3.      }
4.      function Author (name) {
5.          Person.call(this, name) //继承构造器
6.      }
7.
8.      Author.prototype = new Person();
```

11.3.3 门面模式

门面模式是一种流行的设计模式，它为一个已存在的对象创建一个新的接口。门面是一个全新的对象，其背后有一个已存在对象在工作。门面有时候也叫包装器，它们用不同的接口来包装已存在的对象。

```
1.      function DOMWapper (element) {
2.          this.element = element;
3.      }
4.
5.      DOMWapper.prototype.addClass = function (className) {
6.         element.className += "" + className;
7.      }
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

```
8.
9.      DOMWapper.prototype.remove = function () {
10.        this.element.parentNode.removeChild(this.element);
11.     }
12.     //用法
13.     var wrapper = new DOMWrapper(document,getElementById("myDiv"));
14.     //添加一个 className
15.     wrapper.addClass("selected");
16.     //删除元素
17.     wrapper.remove();
```

11.4 关于 polyfill 的注解

polyfill 是对某种功能的模拟，这些功能在新版本的浏览器中有完整的定义和原生的实现。pollyfills 的优点是，如果浏览器提供原生实现，可以非常轻松地移除它们。如果你使用了 polyfills,你需要搞清楚那些浏览器提供了原生实现。并确保 polyfills 的实现和浏览器原生保持完全一致，并再三检查类库是否提供严重这些方法正确性的测试用例。polyfills 的缺点是，和浏览器的原生实现相比，它们的实现可能不精确，这回给你带来很多麻烦，还不如不实现它。从最佳的可维护性角度而言，避免使用 polyfills,相反可以在已存在的功能之上创建门面来实现。

11.5 阻止修改

ECMAScript5 引入几个方法防止对对象的修改。

1. 防止扩展：禁止为对象“添加”属性和方法，但已存在的属性和方法是可以被修改或删除。
2. 密封：类似“防止扩展”，而且禁止为对象“删除”已存在的属性和方法。
3. 冻结：类似“密封”，而且禁止为对象“修改”已存在的属性和方法（所有字段均只读）。

每种锁定的类型都拥有两个方法：一个用来实施操作，另一个用来检测是否应用了相应的操作。如防止扩展，Object.preventExtension()和 Object.isExtensible()两个函数可以使用。

```
1.      var person = {
2.          name:  "Jason";
3.      }
4.      //锁定对象
5.      Object.preventExtension(person);
6.      console.log(Object.isExtensible(person)); //false
7.      person.age = 25; //正常情况下悄悄地失败，在严格模式下会抛出异常
```

使用 Object.seal()函数来密封一个对象。
使用 Object.freeze()函数来冻结一个对象。

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

12.浏览器嗅探

12.1 User-Agent 检测

倘若你选择使用用户代理（user-agent）检测，最安全的方法是只检测旧的浏览器.

12.2 特性检测

特性检测的原理是为特定浏览器的特性进行测试，并仅当特性存在时即可应用特性检测。

```
1.      function getById (id) {
2.          var element =null;
3.
4.          if (document.geElementById) {// DOM
5.              element = document.getElementById(id);
6.          } else if (document.all) {  //IE
7.              element = document.all[id];
8.          } else if (document.layers) { //Netscape <= 4
9.             element = document.layers[id];
10.         }
11.
12.         return element;
13.     }
```

1. 探测标准的方法。
2. 探测不同浏览器的特定方法。
3. 当被探测的方法均不存在时提供一个合乎逻辑的备用方法。

12.3 避免特性推断

一种不当的使用特性检测的情况是“特性推断”。特性推断尝试使用多个特性但仅验证了其中之一。你不能从一个特性的存在来推断另一个特性是否存在。,

12.4 避免浏览器推断

通过特性检测从而推断出是某个浏览器同样是很糟糕的做法。这叫做浏览器推断，是一种错误的实践。

1. Internet Explorer 8，同时支持 window.ActiveXObject 和 window.XMLHttpRequest,也会被识别为 Internet Explorer7。

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

2. 任何实现 document.childNodes 的浏览器，如果没有识别为 Internet Explorer 的话，都有可能识别为 Webkit 内核。
3. 识别出的 WebKit 版本号太小了，再者，WebKit422 或者更高的版本也将会被误认为 WebKit420.
4. 没有坚持 Opera，所以 Opera 要么被无检测为其他浏览器，要么根本不会被检测到。
5. 当有新的浏览器发布时，此代码需要更新。

12.5 应当如何取舍

特性推断和浏览器推断都是糟糕的做法，应当不惜一切代价避免使用。纯粹的特性检测是一种很好的做法，而且几乎在任何情况下，都是你想要的结果。通常，你仅需要在使用前检测特性是否可用。不要试图推断特性间的关系，否则最终得到的结果页不是可靠的。
如果你想使用用户代理嗅探，记住这一点：这么做唯一安全方式是针对旧的或者特定版本的浏览器。而绝不应当针对最新版本或者未来的浏览器。
建议是尽可能的使用特性检测。如果不能这么做的时候，退而求其次，考虑使用用户代理检测。永远不要使用浏览器推断，因为你会被这样维护性很差的代码缠身，而且随着新的浏览器的出现，你需要不断的更新代码。

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

第三部分 附录

A JavaScript 编程风格指南

A.1 缩进

每一行的层级由 4 个空格组成（使用 Tab 键时注意 Tab 键是否设置为 4 个空格）

```
1.          //好的写法
2.          if (true) {
3.              dosomething();
4.          }
```

A.2 行的长度

每行长度不应该超过 80 字符。如果一行多于 80 个字符，应当在一个运算符（逗号，分号等）后换行。下一行应当增加两个缩进（8 个字符）。

```
1.          //好的写法
2.          dosomething (argument1, argument2, argument4, argument4,
3.                  ) ;
```

A.3 原始值

字符串应当始终使用双引号（避免使用单引号）且保持一行。避免在字符串中使用斜线另起一行。

```
1.      //好的写法
2.      var name = "hello" ;
```

数字应当使用十进制整数，科学计算法表示整数，十六进制整数，或者十进制浮点小数，小数点前后应当至少保留一位数字。避免使用八进制直接量

```
1.      //好的写法
2.      var count = 10 ;
3.      var price = 10.0 ;
4.      var num = 0xA2 ;
5.      var price = 1e23 ;
6.      //不好的写法
```


上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

```
var price = 10. ;
var price = .1 ;
var num = 010 ;
```

特殊值 null 除了下述情况下应当避免使用。

3. 用来初始化一个变量,这个变量可能被赋值为一个对象。
4. 用来和一个已经初始化的变量比较,这个变量可以是也可以不是一个对象。
5. 当函数的参数期望是对象时,被用作参数传入。
6. 当函数的返回值期望是对象时,被用作返回值传出。

```
//好的写法
var person = null;
//好的写法
function getPerson(){
    if(condition){
        return new Person("John");
    } else {
        return null;
    }
}
 //好的写法
 var person = getPerson();
 if(person !==null){
     doSomething();
}
//不好的写法，和一个为被初始化的变量比较
var person ;
if(person !==null){
    doSomething();
}
//不好的写法，通过测试判断某个参数是否被传递
function doSomething(arg1,arg2){
    if(arg !==null){
        doSomething();
    }
}
```

避免使用特殊值 undefined，判断一个变量是否定义应当使用 typeof 操作符。

```
//好的写法
if(typeof variable == "undefined"){
    doSomething();
}
//不好的写法：使用了 undefined 直接量
if(typeof variable == undefined){
```


上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

```
8.          doSomething();
9.      }
```

A.4 运算符间距

二元运算符前后必须使用一个空格来保持表达式的整洁。操作符包括赋值运算符和逻辑运算符。

```
1.      //好的写法
2.      var found = (values[i] === item);
3.      //好的写法
4.      if(found && (count >10)) {
5.          doSomething();
6.      }
7.      //不好的写法，丢失了空格
8.      var found = (values[i]===item);
```

A.5 括号间距

当使用括号时，紧接左括号之后和紧接右括号之前不应该有空格。

```
1.      //好的写法
2.      var found = (values[i] === item);
3.      //好的写法
4.      if(found && (count >10)) {
5.          doSomething();
6.      }
7.      //不好的写法，左括号之后有额外的空格
8.      var found = ( values[i] === item);
9.      //不好的写法，右括号之前有额外的空格
10.     if(found && (count >10) ) {
11.     }
12.     //不好的写法，参数两边有额外的空格
13.     if(i =0; i < 3; i++) {
14.         process(  i );
15.     }
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

A.6 对象直接量

对象直接量应当使用如下格式。

1. 起始左花括号应当同表达式保持同一行。
2. 每个属性的名值对应当保持一个缩进，第一个属性应当在左花括号后另起一行。
3. 每个属性的名值对应当使用不含引号的属性名（含引号的是 JSON），其后紧跟一个冒号（之前不含空格）,而后是值。
4. 倘若属性值是函数类型。函数体应当在属性名之下另起一行，而且其前后均应保留一个空行。
5. 一组相关的属性值前后可以插入空行以提升代码额可读性。
6. 结束的右花括号应当独占一行。

```
//好的写法
var object = {

    key1: value1,

    key2: value2,

    func: function() {
        doSomething();
    },

    key3: value3
};
```

当对象字面量作为函数参数时，如果值是变量，其实花括号应当同函数名在同一行。所以其余先前列出的规则同样适用。

```
//好的写法
func: function() {
    doSomething({
        key1:  value1,
        key2: value2
    });
};
//不好的写法：  所有代码写在一行
doSomething({key1: value1,key2: value2});
```

补充：JSON 和直接量唯一的区别就是引号（例：age 和"age"）。
它们是可以相互转换地，对象直接量可以转换为 JSON，JSON 也可以转换为对象直接量。
转换的方法如下：
对象直接量 —> JSON ，这个需要用到 W3C 官方提供的 JOSN.js 进行转换。
var JSON = JSON.stringify(对象直接量)，这样变量 JSON 就是对象直接量了。

JSON—> 对象直接量：

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

方法一：eval (' ('+JSON+')');
方法二：JSON.parse(JSON) ;
建议使用第二种，更安全。

A.7 注释

频繁的使用注解有助于他人理解你的代码。如下情况应当使用注解。

1. 代码晦涩难懂。
2. 可能被误认为错误的代码。
3. 必要但并不明显的针对特点浏览器的代码。
4. 对于对象、方法或者属性，生成文档时有必要的（使用恰当的文档注释）。

A.7.1 单行注释

单行注释应当用来说明一行代码或者一组相关的代码。单行注释可能有三种使用方法。

4. 独占一行的注释，用来解释下一行代码。
5. 在代码行的尾部注释，用来解释它之前的代码。
6. 多行，用来注释掉一个代码块。

```
1.        //好的写法
2.        func: function() {
3.
4.            //注释之前 有空行 有缩进
5.            doSomething({
6.                key1： value1,
7.                key2: value2
8.            });
9.        };
```

对于代码行尾单行注释的情况，应确保代码结尾同注释之前至少一个缩进

```
1.        //好的写法
2.        var reslut = something; // something xxx
3.        //不好的写法
4.        var reslut = something;// something xxx
```

注释一个代码块时在连续多行使用单行注释是唯一可以接受的情况。多行注释不应当出现在这种情况下使用。

```
1.        //好的写法
2.        //if (xxx) {
3.        //    doSomething();
4.        //}
```


上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

A.7.2 多行注释

多行注释应当在代码需要更多文字去解释的时候使用。每个多行注释都在至少有如下三行。

2. 首行仅仅包括/*注释开始。该行不应当有其他文字。
3. 接下来的行以*开头并保持左对齐。这些行可以有文字描述。
4. 最后一行以*/开头并同先前行保持对齐。也不应当有其他文字。

多行注释的首行应当保持同它描述代码的相同层次的缩进。后续的每行应当有同样层次的缩进并附加一个空格（为了适当保持*字符的对齐）。每一个多行代码之前应当预留一个空行。

```
1.          //好的写法
2.          if (xxx) {
3.              /*
4.               * 哈哈
5.               * 嘻嘻
6.               */
7.              doSomething();
8.          }
```

A.7.3 注释声明

注释有时候也可以用来给一段代码声明额外的信息。这些声明的格式以单个单词打头并紧跟一个冒号。课使用的声明如下。

TODO

说明代码还未完成。应当包含下一步要做的事情。

HACK

表明代码实现走了一个捷径。应当包含为何使用 hack 的原因。这也可能表明该问题可能会有更好的解决方法。

XXX

说明代码是有问题的并应当尽快修复。

FIXME

说明代码是有问题的并应当尽快修复。重要性略次于 XXX。

REVIEW

说明代码任何可能的改动都需要评审。

这些声明可能在一行或多行注释中使用，并且应当遵循同一般注释类型相同的格式规则。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

```
1.          //好的写法
2.          /*
3.           * HACK:不得不针对 IE6 进行特殊处理,在下一个版本会重
4.           * 写这段代码
5.           */
6.          if (xxx) {
7.              doSomething();
8.          }
```

A.8 变量声明

所有变量在使用前都应当事先定义。变量定义应当放在函数开头。使用一个 var 表达式每行一个变量。除了首行，所有行都应当多一层缩进以使变量名能够垂直方向对齐。变量定义时应当初始化，并且赋值操作符应当保持一致的缩进。初始化的变量应当在未初始化变量之前。

```
1.          //好的写法
2.          var  count = 10,
3.               name = "Jook",
4.               found = false,
5.               empty;
```

A.9 函数声明

函数应当在使用前提前定义。一个不是作为方法的函数（也就是说没有作为一个对象的属性）应当使用函数定义的格式（不是函数表达式和 Function 构造器格式）。函数名和开始圆括号之间不应当有空格。结束的圆括号和右边的花括号之间应该留一个空格。右侧的花括号应当同 function 关键字保持同一行。开始和结束括号之间不应该有空格。参数名之间应当在逗号之后保留一个空格。函数体应当保持一级缩进。

```
1.          //好的写法
2.          function doSomething(arg1, arg2) {
3.              return null;
4.          }
5.          //不好的写法：函数表达式
6.          var doSomething  = function (arg1, arg2) {
7.              return null;
8.          }
9.          //不好的写法：左侧花括号位置不对
10.         function doSomething(arg1, arg2)
11.         {
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

```
12.             return null;
13.         }
14.         //不好的写法：使用了 Function 构造器
15.         var doSometing = new Function("arg1", "arg2");
```

其他函数内部定义的函数应当在 var 语句后立即定义

```
1.          //好的写法
2.          function doSomething() {
3.
4.              var count = 10,
5.                  name ="xxx",
6.                  found = false;
7.
8.              function inner() {
9.                 doSomeOtherThing();
10.             }
11.         }
```

匿名函数可能作为方法赋值给对象，或者作为其他函数的参数。function 关键字同开始括号之间不应有空格。

```
1.      //好的写法
2.      object.method = function() {
3.         doSomething();
4.      }
5.      //不好的写法：不正确的空格
6.      object.method = function () {
7.         doSomething();
8.      }
```

立即被调用的函数应当在函数调用的外层用圆括号包裹。

```
1.      //好的写法
2.      var value = (function() {
3.          return{
4.              aa:"a"
5.          }
6.      }());
7.      //不好的写法：函数调用的外层没用圆括号包裹
8.      var value = function() {
9.          return{
10.             aa:"a"
11.         }
12.     }();
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

A.10 命名

变量和函数在命名是应当小心。命名应仅限于数字、字母、字符，某些情况下也可以使用下划线。最好不要在任何命名中使用美元符号（$）或者反斜杠（\）。

变量命名应当采用驼峰命名格式，首字母小写，每个单词首字母大写。变量名的第一个单词应当是一个名词（而非动词）以避免同函数混淆。不要在变量命名中使用下划线。

```
1.  //好的写法
2.  var account = "8401-1";
3.  //不好的写法：大写字母开头
4.  var Account = "8401-1";
5.  //不好的写法:动词开头
6.  var getAccount = "8401-1";
7.  //不好的写法:使用下划线
8.  var account_number = "8401-1";
```

函数命名也应当采用驼峰命名格式。函数名的第一个单词应当是动词（而非动词）来避免同变量混淆。函数名中最好不要使用下划线。

```
1.  //好的写法
2.  function doSomething() {
3.  }
4.  //不好的写法：大写字母开头
5.  function DoSomething() {
6.  }
7.  //不好的写法:名词开头
8.  function car() {
9.  }
10. //不好的写法:使用下划线
11. function do_something() {
12. }
```

构造函数—通过 new 运算符创建对象的函数—也应当以驼峰格式命名并且首字符大写。构造函数名名称应当以非动词开头，因为 new 代表这创建一个对象实例的操作。

```
1.  //好的写法
2.  function MySomething() {
3.  }
4.  //不好的写法：小写字母开头
5.  function mySomething() {
6.  }
7.  //不好的写法:名词开头
8.  function getCar() {
9.  }
10. //不好的写法:使用下划线
```


上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

```
11.    function get_something() {
12.    }
```

常量（值不会被改变的变量）的命名应当是所有字母大写，不同单词之间用单个下划线隔开。

```
1.     //好的写法
2.     var TOTAL_COUNT = 10;
3.     //不好的写法：带有小写
4.     var total_COUNT = 10;
```

对象的属性同变量的命名规则相同。对象的方法同函数的命名规则相同。如果属性或者方法是私有的，应当那个之间加一个下划线。

```
1.     //好的写法
2.     var object = {
3.         _count: 10,
4.
5.         _getCount: function () {
6.             return this._count;
7.         }
8.     };
```

A.11 严格模式

严格模式应当仅限在函数内部使用，千万不要在全局使用。（严格模式可以提高 javascript 执行效率）

```
1.     //不好的写法
2.     "use strict"
3.
4.     function doSomething()  {
5.     }
6.     //好的写法
7.     function doSomething()  {
8.         "use strict";
9.         doSomething();
10.    }
```

如何你期望在多个函数中使用严格模式而不需要多次声明"use strict"，可以使用立即被调用额函数。

```
1.     //好的写法
2.     (function ()  {
3.         "use strict";
4.         function doSomething(){
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

```
        };
        function doSomeOtherthing(){
        };
    }());
```

A.12 赋值

当给变量赋值时，如果右侧是含有比较语句的表达式，需要用圆括号包裹。

```
    //好的写法
    var flag = (i < conut);
    //不好的写法：遗漏圆括号
    var flag = i < conut;
```

A.13 等号运算符

使用===（严格相等）和！==（严格不相等）代替==（相等）和！=（不相等）来避免 弱类型转换错误。

```
    //好的写法
    var flag = (a === b);
    //不好的写法：使用 ==
    var flag = (a == b);
```

A.15 三元操作符

三元运算符应当仅仅在条件赋值语句中，而不要作为 if 语句的替代品。

```
    //写法
    var value = condition ? value1 : value2;
    //不好的写法：没有赋值，应当使用 if 表达式
    condition ? doSomething() : doSomeOtherThing();
```

A.16 语句

简单语句

每一行最多只包含一条语句。所有简单的语句都应该以分号（;）结束。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

```
//好的写法
count++;
a = b;
//不好的写法：多个表达式写在一行
count++; a = b;
```

返回语句

返回语句当返回一个值的时候不应当使用圆括号包裹，除非在某些情况下这么做可以让返回值更容易理解。例如：

```
return;
return collection.size();
return (siz > 0 ? size : defaultSize);
```

复合语句

复合语句是大括号起来的语句列表。

1. 括起来的语句应当较复合语句多缩进一个层级。
2. 开始的大括号应当在复合语句所在行的末尾；结束的大括号应当独占一行且同复合语句的开始保持同样的缩进。
3. 当语句是控制结构的一部分是，诸如 if 或者 for 语句，所有语句都需要用大括号括起来，也包括单个语句。这个约定使得我们更方便地添加语句而不用担心忘记加括号而引起 bug。
4. 像 if 一样的语句开始的关键词，其后应该紧跟一个空格，其实大括号应当在空格之后。

```
//好的写法
if (condition) {
    doSomething();
};
//不好的写法：遗漏花括号
if (condition)
    doSomething();
```

for 语句

for 语句的初始化部分不应当有变量声明。

```
//好的写法
var i,
    len;
for (i=0, len=10; i < len; i++ ) {
    doSomething();
};
//不好的写法：初始化时声明变量
for (var i=0, len=10; i < len; i++ ) {
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

```
9.          doSomething();
10.     };
```

当使用 for-in 语句是，记得使用 hasOwnProperty()进行双重检查来过滤对象的成员。还有当心 for-in 是无序的，注意合理使用。

while 语句

```
1.      while (condition) {
2.      }
```

do 语句

```
1.      do {
2.          statements
3.      } while (condition);
```

switch 语句

switch 下的每一个 case 都应当保持一个缩进。除第一个之外包括 default 在内 meiyigecase 都应当在之前保持一个空行。每一组语句（除了 default）都应当以 break、return、throw 结尾，或者用一行注释表示跳过。

```
1.      switch (expression) {
2.          case 1:
3.              /*falls through*/
4.
5.          case 2:
6.              doSomething();
7.              break;
8.
9.          case 3:
10.             return true;
11.
12.         default:
13.             throw new Error("This shouldn't happen")
14.     }
```

如果一个 switch 语句不包含 default，应当用一行注释替代。

```
1.       switch (value) {
2.          case 1:
3.              /*falls through*/
4.
5.          case 2:
```


上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

```
            doSomething();
            break;

        case 3:
            return true;

       //没有 default
    }
```

try 语句

```
    try {
        statements
    } catch (vaeiable) {
       statements
    }

    //含 finally
    try {
        statements
    } catch (vaeiable) {
       statements
    } finally {
        statements
    }
```

A.17 留白

在逻辑相关的代码块之间添加空行可以提高代码的可读性。

两行空行仅限在如下情况中使用。

1. 在不同的源代码文件之间。
2. 在类和接口定义之间

单行空行仅限在如下情况中使用。

1. 方法之间。
2. 方法中局部变量和第一行语句之间。
3. 多行或者单行注释之前。
4. 方法中逻辑代码块之间以提升代码的可读性。

空格应当在如下情况中使用

1. 关键词后跟括号的情况应当使用空格隔开。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

2. 参数列表中逗号之后应当保留一个空格。
3. 所有除了点（.）之外的二元运算符，其操作数都应当用空格隔开。单目运算符的操作数之间不应该用空白隔开，诸如一元减号，递增（++），递减（——）。
4. for 语句中表达式之间应当用空格隔开。

A.18 需要避免的

1. 切勿使用像 String 一类原始包装类型创建新的对象。
2. 避免使用 eval()。
3. 避免使用 with 语句。该语句在严格模式中不复存在，可能在未来的 ECMAScript 标准中也将去除。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

B 糟粕

B.1 全局变量

三种方式定义全局变量：

（1）var foo = value;

（2）window.foo = value;

（3）foo = value;

建议：尽量少用全局变量。

B.2 作用域

Javascrip 中无块级作用域,最好的方式是在在每个函数开头部分声明所有变量。

B.3 自动插入分号

小心 return 语句中自动插入分号导致的后果。

建议：{ 要紧跟在 return 后，返回值与 return 在同一行。

B.4 保留字

它们不能被用来命名变量或参数。当保留字被用作对象字面量的键值对，它们必须被引号括起来。它们不能用在点表示法中，所以有时候必须用括号表示法。

```
1.  var method; //ok
2.  var class; //非法
3.  object = {box: value}; //ok
4.  object = {case: value}; //非法
5.  object = {"case": value}; //ok
6.  object.box = value; //ok
7.  object.case = value; //非法
8.  object[“case”]; //ok
```

B.5 Unicode

javascript 字符是 16 位的,只能覆盖 65535 个字符。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

B.6 typeof

typeof null 返回 "object" 而不是 “null”。所以更好的检测方法是 myValue === null

检测对象的值（typeof 不能辨别出 null 与对象，但 null 值为假，而对象为真）：

```
if(myValue && typeof myValue ==="object") {
    // myValue 是一个对象或数组
}
```

B.7 parseInt

增加第二个参数,明确进制,parseInt("08",10)。

B.8 +

如果你打算用 + 去做加法运算，请确保两个运算符都是整数。

B.9 浮点型

二进制浮点数不能正确的处理十进制小数,0.1+0.2 不等于 0.3,不过整数部分是精确的。

B.10 NaN

```
typeof NaN === "number"  //true;
NaN === NaN //false
NaN !== NaN //true
```

自定义一个 isNumber 函数

```
function isNumber(value) {
    rerurn typeof value ==="number" && isFinite(value);
}
```

B.11 伪数组

typeof 不能辨别数组和对象。要判断一个值是否是数组，要检测它的 constructor 属性：

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

```
if(myValue && typeof myValue === "object" && typeof myValue.length ===
        "number" && typeof myValue. constructor === Array){
        //数组
}
```

B.12 假值

javascript 拥有众多假值

值	类型
0	Number
NaN（非数字）	Number
''（空字符串）	String
false	Boolean
null	Object
undefined	Undefined

==会强制转换类型，用===更可靠。

B.13 hasOwnProperty

hasOwnProperty 是一个方法，不是一个运算符，所以会被函数或非函数值替换。
a. hasOwnProperty = null; //触雷

B.14 对象

因为原型链的存在,javascript 对象永远不会有真的空对象。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

C 鸡肋

C.1 ==

运算符 ==和!= 会试图强制转化判断值的类型,规则复杂。请使用===和！==

C.2 with 语句

可能出现歧义,并影响处理器速度，不推荐使用。

C.3 eval

降低安全性,影响性能，增加阅读复杂性，不推荐使用。

C.4 continue 语句

重构移出 continue 后,性能得到改善。

C.5 switch 贯穿

除非明确中断流程，否则每次条件判断后都贯穿到下一个 case 条件。

C.6 缺少块的语句

始终使用代码块（加{}）会让代码更容易理解。

C.7 ++ --

使代码变得更隐晦。

C.8 位运算符

javascript 没有整数类型,位操作符将它们的运算数先转换成整数,接着执行运算,然后再转化回去,非常慢。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

C.9 function 语句对比函数表达式

不在 if 语句中使用 function,尽量用 var func = function （）{} 的形式声明。

C.10 类型的包装对象

避免使用 new Object 和 new Array。可使用{} 和[] 代替。

C.11 new

更好的应对策略是根本不去使用 new。

C.12 void

将返回 undefined,没有什么用,而且让人困惑，尽量避免使用。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

D 优化法则

D.1 Yslow-23 条规则

1. 减少 HTTP 请求次数

合并图片、CSS、JS，改进首次访问用户等待时间。

2. 使用 CDN

就近缓存==>智能路由==>负载均衡==>WSA 全站动态加速。

3. 避免空的 src 和 href

当 link 标签的 href 属性为空、script 标签的 src 属性为空的时候，浏览器渲染的时候会把当前页面的 URL 作为它们的属性值，从而把页面的内容加载进来作为它们的值。

4. 为文件头指定 Expires

使内容具有缓存性。避免了接下来的页面访问中不必要的 HTTP 请求。

5. 使用 gzip 压缩内容

压缩任何一个文本类型的响应，包括 XML 和 JSON，都是值得的。

6. 把 CSS 放到顶部

避免空白页。

7. 把 JS 放到底部

防止 js 加载对之后资源造成阻塞。

8. 避免使用 CSS 表达式

影响性能。

9. 将 CSS 和 JS 放到外部文件中

目的是缓存，但有时候为了减少请求，也会直接写到页面里，需根据 PV 和 IP 的比例权衡。

10. 权衡 DNS 查找次数

减少主机名可以节省响应时间。但同时，需要注意，减少主机会减少页面中并行下载的数量。

IE 浏览器在同一时刻只能从同一域名下载两个文件。当在一个页面显示多张图片时，IE 用户的图片下载速度就会受到影响。所以新浪会搞 N 个二级域名来放图片。

11. 精简 CSS 和 JS

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

12. 避免跳转

同域：注意避免反斜杠“/”的跳转；

跨域：使用 Alias 或者 mod_rewirte 建立 CNAME（保存域名与域名之间关系的 DNS 记录）。

13. 删除重复的 JS 和 CSS

重复调用脚本，除了增加额外的 HTTP 请求外，多次运算也会浪费时间。在 IE 和 Firefox 中不管脚本是否可缓存，它们都存在重复运算 JavaScript 的问题。

14. 配置 ETags

它用来判断浏览器缓存里的元素是否和原来服务器上的一致。比 last-modified date 更具有弹性，例如某个文件在 1 秒内修改了 10 次，Etag 可以综合 Inode(文件的索引节点(inode)数)，MTime(修改时间)和 Size 来精准的进行判断，避开 UNIX 记录 MTime 只能精确到秒的问题。服务器集群使用，可取后两个参数。使用 ETags 减少 Web 应用带宽和负载。

15. 可缓存的 AJAX

“异步”并不意味着“即时”：Ajax 并不能保证用户不会在等待异步的 JavaScript 和 XML 响应上花费时间。

16. 使用 GET 来完成 AJAX 请求

当使用 XMLHttpRequest 时，浏览器中的 POST 方法是一个“两步走”的过程：首先发送文件头，然后才发送数据。因此使用 GET 获取数据时更加有意义。

17. 减少 DOM 元素数量

是否存在一个是更贴切的标签可以使用？人生不仅仅是 DIV+CSS。

18. 避免 404

有些站点把 404 错误响应页面改为“你是不是要找***”，这虽然改进了用户体验但是同样也会浪费服务器资源（如数据库等）。最糟糕的情况是指向外部 JavaScript 的链接出现问题并返回 404 代码。首先，这种加载会破坏并行加载；其次浏览器会把试图在返回的 404 响应内容中找到可能有用的部分当作 JavaScript 代码来执行。

19. 减少 Cookie 的大小

20. 使用无 cookie 的域

比如图片 CSS 等，Yahoo! 的静态文件都在主域名以外，客户端请求静态文件的时候，减少了 Cookie 的反复传输对主域名的影响。

21. 不要使用滤镜

png24 的在 IE6 半透明那种东西，别乱使，淡定的切成 PNG8+jpg。

22. 不要在 HTML 中缩放图片

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

23. 缩小 favicon.ico 并缓存

上海市闸北区恒丰路 600 号 7 楼　　邮编 200070
Tel: +86-21-60540799

E JavaScript 工具集

E.1 编辑器

Sublime Text 2 （http://www.sublimetext.com/2）

E.2 性能检测工具

YSlow（http://developer.yahoo.com/yslow/）

E.3 构建工具

Ant（http://ant.apache.org/）
Grant（https://github.com/cowboy/grunt）

E.4 文档生成器

Js Doc ToolKit(http://code.google.com/p/jsdoc-toolkit-ant-task)
YUI Doc(http://yui.github.io/yuidoc/)

E.5 代码检查工具

JSLint(http://www.jslint.com/)
JSHint(http://www.jshint.com/)

E.6 压缩工具

YUI Compressor（http://yuilibrary.com/download/#yuicompressor）
JSMin（http://www.crockford.com/javascript/jsmin.html）

E.7 测试工具

JsTestDriver(https://code.google.com/p/js-test-driver/)
YUI Test(http://yuilibrary.com/projects/yuitest/)

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

F CSS 规范大全

F.1 文件规范

1、文件均归档至约定的目录中（具体要求以豆瓣的 CSS 规范为例进行讲解）：

1-1、所有的 CSS 分为两大类：通用类和业务类。通用的 CSS 文件，放在如下目录中：

1-1-1：基本样式库 /css/core
1-1-2：通用 UI 元素样式库 /css/lib
1-1-3：JS 组件相关样式库 /css/ui

1-2、业务类的 CSS 是指和具体产品相关的文件，放在如下目录中：

1-2-1：读书 /css/book/
1-2-2：电影 /css/movie/
1-2-3：音乐 /css/music/
1-2-4：社区 /css/sns/
1-2-5：小站 /css/site/
1-2-6：同城 /css/location/
1-2-7：电台 /css/radio/

1-3、外联 CSS 文件适用于全站级和产品级通用的大文件。内联 CSS 文件适用于在一个或几个页面共用的 CSS。另外，对具体的 CSS 进行文档化的整理。如：

1-3-1：util-01 reset /css/core/reset.css
1-3-2：util-02 通用模块容器 /css/core/mod.css
1-3-3：ui-01. 喜欢按钮 /css/core/fav_btn.css
1-3-4：ui-02. 视频/相册列表项 /css/core/media_item.css
1-3-5：ui-03. 评星 /css/core/rating.css
1-3-6：ui-04. 通用按钮 /css/core/common_button.css
1-3-7：ui-05. 分页 /css/core/pagination.css
1-3-8：ui-06. 推荐按钮 /css/core/rec_btn.css
1-3-9：ui-07. 老版对话框 /css/core/old_dialog.css
1-3-10：ui-08. 老版 Tab /css/core/old_tab.css
1-3-11：ui-09. 老版成员列表 /css/core/old_userlist.css
1-3-12：ui-10. 老版信息区 /css/core/notify.css
1-3-13：ui-11. 社区用户导航 /css/core/profile_nav.css
1-3-14：ui-12. 当前大社区导航 /css/core/site_nav.css
1-3-15：ui-13. 加载中 /css/lib/loading.css

2、文件引入可通过外联或内联方式引入

2-1、link 和 style 标签都应该放入 head 中，原则上，不允许在 html 上直接写样式。避免在

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

CSS 中使用@import，嵌套不要超过一层。
2-2、外联方式：<link rel=”stylesheet” href=”…” />（类型声明 type=”text/css”可以省略）
2-3、内联方式：<style>…</style> （类型声明 type=”text/css”可以省略）

3、文件名、文件编码及文件大小

3-1、文件名必须由小写字母、数字、中划线组成。
3-2、文件必须用 UTF-8 编码，使用 UTF-8（非 BOM），在 HTML 中指定 UTF-8 编码，在 CSS 中则不需要特别指定因为默认就是 UTF-8。
3-3、单个 CSS 文件避免过大（建议少于 300 行）。

F.2 注释规范

1、文件顶部注释（推荐使用）

Css 代码

```
/*
* @description: 中文说明
* @author: name
* @update: name (2013-04-13 18:32)
*/
```

2、模块注释

模块注释必须单独写在一行

Css 代码

```
/* module: module1 by 张三 */
…
/* module: module2 by 张三 */
```

3、单行注释与多行注释

单行注释可以写在单独一行，也可以写在行尾，注释中的每一行长度不超过 40 个汉字，或者 80 个英文字符。

Css 代码

```
/* this is a short comment */
```


上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

多行注释必须写在单独行内

Css 代码

```
/*
* this is comment line 1.
* this is comment line 2.
*/
```

4、特殊注释

用于标注修改、待办等信息

Css 代码

```
/* TODO: xxxx by name 2013-04-13 18:32 */
/* BUGFIX: xxxx by name 2012-04-13 18:32 */
```

5、区块注释

对一个代码区块注释（可选），将样式语句分区块并在新行中对其注释。

Css 代码

```
/* Header */
/* Footer */
/* Gallery */
```

F.3 命名规范

使用有意义的或通用的 ID 和 class 命名：ID 和 class 的命名应反映该元素的功能或使用通用名称，而不要用抽象的晦涩的命名。反映元素的使用目的是首选；使用通用名称代表该元素不表特定意义，与其同级元素无异，通常是用于辅助命名；使用功能性或通用的名称可以更适用于文档或模版变化的情况。

- /* 不推荐: 无意义 */ #yee-1901 {}
- /* 不推荐: 与样式相关 */ .button-green {}.clear {}
- /* 推荐: 特殊性 */ #gallery {}#login {}.video {}
- /* 推荐: 通用性 */ .aux {}.alt {}

常用命名（多记多查英文单词）：page、wrap、layout、header(head)、footer(foot、ft)、content(cont)、menu、nav、main、submain、sidebar(side)、logo、banner、title(tit)、popo(pop)、icon、note、btn、txt、iblock、window(win)、tips 等。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

ID 和 class 命名越简短越好，只要足够表达涵义。这样既有助于理解，也能提高代码效率。

- /* 不推荐 */ #navigation {}.atr {}
- /* 推荐 */ #nav {}.author {}

类型选择器避免同时使用标签、ID 和 class 作为定位一个元素选择器；从性能上考虑也应尽量减少选择器的层级。

- /* 不推荐 */ul#example {}div.error {}
- /* 推荐 */#example {}.error {}

命名时需要注意的点：

- 规则命名中，一律采用小写加中划线的方式，不允许使用大写字母或 _
- 命名避免使用中文拼音，应该采用更简明有语义的英文单词进行组合
- 命名注意缩写，但是不能盲目缩写，具体请参见常用的 CSS 命名规则
- 不允许通过 1、2、3 等序号进行命名
- 避免 class 与 id 重名
- id 用于标识模块或页面的某一个父容器区域，名称必须唯一，不要随意新建 id
- class 用于标识某一个类型的对象，命名必须言简意赅。
- 尽可能提高代码模块的复用，样式尽量用组合的方式
- 规则名称中不应该包含颜色（red/blue）、定位（left/right）等与具体显示效果相关的信息。应该用意义命名，而不是样式显示结果命名。

1、常用 id 的命名：

(1) 页面结构

- 容器: container
- 页头：header
- 内容：content/container
- 页面主体：main
- 页尾：footer
- 导航：nav
- 侧栏：sidebar
- 栏目：column
- 页面外围控制整体布局宽度：wrapper

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

- 左右中：left right center

(2) 导航

- 导航：nav
- 主导航：mainbav
- 子导航：subnav
- 顶导航：topnav
- 边导航：sidebar
- 左导航：leftsidebar
- 右导航：rightsidebar
- 菜单：menu
- 子菜单：submenu
- 标题: title
- 摘要: summary

(3) 功能

- 标志：logo
- 广告：banner
- 登陆：login
- 登录条：loginbar
- 注册：regsiter
- 搜索：search
- 功能区：shop
- 标题：title
- 加入：joinus
- 状态：status
- 按钮：btn
- 滚动：scroll
- 标签页：tab
- 文章列表：list
- 提示信息：msg
- 当前的: current
- 小技巧：tips
- 图标: icon
- 注释：note
- 指南：guild

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

- 服务：service
- 热点：hot
- 新闻：news
- 下载：download
- 投票：vote
- 合作伙伴：partner
- 友情链接：link
- 版权：copyright

2、常用 class 的命名：

(1) 颜色:使用颜色的名称或者 16 进制代码，如

- .red { color: red; }
- .f60 { color: #f60; }
- .ff8600 { color: #ff8600; }

(2) 字体大小,直接使用”font+字体大小”作为名称，如

- .font12px { font-size: 12px; }
- .font9pt {font-size: 9pt; }

(3) 对齐样式,使用对齐目标的英文名称，如

- .left { float:left; }
- .bottom { float:bottom; }

(4) 标题栏样式,使用”类别+功能”的方式命名，如

- .barnews { }
- .barproduct { }

F.4 书写规范

1、排版规范

(1) 使用 4 个空格，而不使用 tab 或者混用空格+tab 作为缩进；
(2) 规则可以写成单行，或者多行，但是整个文件内的规则排版必须统一；

单行形式书写风格的排版约束：

- 如果是在 html 中写内联的 css，则必须写成单行；

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

- 每一条规则的大括号 { 前后加空格 ；
- 每一条规则结束的大括号 } 前加空格；
- 属性名冒号之前不加空格，冒号之后加空格；
- 每一个属性值后必须添加分号；并且分号后空格；
- 多个 selector 共用一个样式集，则多个 selector 必须写成多行形式 ；

多行形式书写风格的排版约束：

- 每一条规则的大括号 { 前添加空格；
- 多个 selector 共用一个样式集，则多个 selector 必须写成多行形式；
- 每一条规则结束的大括号 } 必须与规则选择器的第一个字符对齐；
- 属性名冒号之前不加空格，冒号之后加空格；
- 属性值之后添加分号；

2、属性编写顺序

- 显示属性：display/list-style/position/float/clear …
- 自身属性（盒模型）：width/height/margin/padding/border
- 背景：background
- 行高：line-height
- 文本属性：color/font/text-decoration/text-align/text-indent/vertical-align/white-space/content…
- 其他：cursor/z-index/zoom/overflow
- CSS3 属性：transform/transition/animation/box-shadow/border-radius
- 如果使用 CSS3 的属性，如果有必要加入浏览器前缀，则按照 -webkit- / -moz- / -ms- / -o- / std 的顺序进行添加，标准属性写在最后。
- 链接的样式请严格按照如下顺序添加： a:link -> a:visited -> a:hover -> a:active

3、规则书写规范

- 使用单引号，不允许使用双引号;
- 每个声明结束都应该带一个分号，不管是不是最后一个声明;
- 除 16 进制颜色和字体设置外，CSS 文件中的所有的代码都应该小写;
- 除了重置浏览器默认样式外，禁止直接为 html tag 添加 css 样式设置;
- 每一条规则应该确保选择器唯一，禁止直接为全局.nav/.header/.body 等类设置属性;

4、代码性能优化

- 合并 margin、padding、border 的-left/-top/-right/-bottom 的设置，尽量使用短名称。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

- 选择器应该在满足功能的基础上尽量简短，减少选择器嵌套，查询消耗。但是一定要避免覆盖全局样式设置。
- 注意选择器的性能，不要使用低性能的选择器。
- 禁止在 css 中使用*选择符。
- 除非必须，否则，一般有 class 或 id 的，不需要再写上元素对应的 tag。
- 0 后面不需要单位，比如 0px 可以省略成 0，0.8px 可以省略成.8px。
- 如果是 16 进制表示颜色，则颜色取值应该大写。
- 如果可以，颜色尽量用三位字符表示，例如#AABBCC 写成#ABC 。
- 如果没有边框时，不要写成 border:0，应该写成 border:none 。
- 尽量避免使用 AlphaImageLoader 。
- 在保持代码解耦的前提下，尽量合并重复的样式。
- background、font 等可以缩写的属性，尽量使用缩写形式 。

5、CSS Hack 的使用

请不用动不动就使用浏览器检测和 CSS Hacks，先试试别的解决方法吧！考虑到代码高效率和易管理，虽然这两种方法能快速解决浏览器解析差异，但应被视为最后的手段。在长期的项目中，允许使用 hack 只会带来更多的 hack，你越是使用它，你越是会依赖它！

推荐使用下面的：

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

区别属性：

IE6	_property:value
IE6/7	*property:value
IE6/7/8/9	property:value\9

区别规则：

IE6	* html selector { ... }
IE7	*:first-child+html selector { ... }
非IE6	html>body selector { ... }
firefox only	@-moz-document url-prefix() { ... }
saf3+/chrome1+	@media all and (-webkit-min-device-pixel-ratio:0) { ... }
opera only	@media all and (-webkit-min-device-pixel-ratio:10000),not all and (-webkit-min-device-pixel-ratio:0) { ... }
iPhone/mobile webkit	@media screen and (max-device-width: 480px) { ... }

6、字体规则

- 为了防止文件合并及编码转换时造成问题，建议将样式中文字体名字改成对应的英文名字，如：黑体(SimHei) 宋体(SimSun) 微软雅黑(Microsoft Yahei，几个单词中间有空格组成的必须加引号)
- 字体粗细采用具体数值，粗体 bold 写为 700，正常 normal 写为 400
- font-size 必须以 px 或 pt 为单位，推荐用 px（注：pt 为打印版字体大小设置），不允许使用 xx-small/x-small/small/medium/large/x-large/xx-large 等值
- 为了对 font-family 取值进行统一，更好的支持各个操作系统上各个浏览器的兼容性，font-family 不允许在业务代码中随意设置

F.5 测试规范

1、了解浏览器特效支持

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

为了页面性能考虑，如果浏览器不支持 CSS3 相关属性的，则该浏览器的某些特效将不再支持，属性的支持情况如下表所示（Y 为支持，N 为不支持）：

浏览器	圆角	阴影	动画	文字阴影	透明	背景渐变	空间变换
Chrome 5+	Y	Y	Y	Y	Y	Y	Y
Firefox 4+	Y	Y	Y	Y	Y	Y	Y
Safari 5+	Y	Y	Y	Y	Y	Y	Y
Opera	Y	Y	Y	Y	Y	N	Y
IE9+	Y	Y	N	N	Y	N	Y
Chrome 5-	N	N	Y	Y	Y	Y	Y
Firefox 4-	N	N	N	Y	Y	N	N
Safari 5-	N	Y	Y	Y	Y	N	Y
IE8	N	N	N	N	N	N	N
IE7	N	N	N	N	N	N	N
IE6	N	N	N	N	N	N	N

2、 设定浏览器支持标准

浏览器	浏览量占比	WinXP	Win7
IE 8.0	37.75%	A	A
IE 6.0	17.65%	B	B
Chrome	17.63%	A	A
IE 9.0	10.58%	B	A
搜狗高速	6.26%	B	B
IE 7.0	1.66%	B	B
奇虎360	1.40%	B	b
Maxthon	1.27%	C	C
Firefox	1.19%	C	C
其他	4.62%	C	C

- A 级－交互和视觉完全符全设计的要求
- B 级－视觉上允许有所差异，但不破坏页面的整体效果
- C 级－可忽略设计上的细节，但不防碍使用

3、常用样式测试工具

- W3C CSS validator：http://jigsaw.w3.org/css-validator/
- CSS Lint：http://csslint.net/
- CSS Usage：https://addons.mozilla.org/en-us/firefox/addon/css-usage/

F.6 其他规范

- 不要轻易改动全站级 CSS 和通用 CSS 库。改动后，要经过全面测试。
- 避免使用 filter

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799

- 避免在 CSS 中使用 expression
- 避免过小的背景图片平铺。
- 尽量不要在 CSS 中使用!important
- 绝对不要在 CSS 中使用”*”选择符
- 层级(z-index)必须清晰明确，页面弹窗、气泡为最高级（最高级为 999），不同弹窗气泡之间可在三位数之间调整；普通区块为 10-90 内 10 的倍数；区块展开、弹出为当前父层级上个位增加，禁止层级间盲目攀比。
- 背景图片请尽可能使用 sprite 技术，减小 http 请求，考虑到多人协作开发, sprite 按照模块、业务、页面来划分均可。

上海市闸北区恒丰路 600 号 7 楼 邮编 200070
Tel: +86-21-60540799