**一.重构**

1.什么是重构？

重构是在不改变程序行为的基础上进行小的改动，是代码逐渐改善的过程。

2.为何重构？

（1）难以辨认的代码（2）自家开发的网站，自己都用不了。（3）缓慢的页面呈现速度

（4）不同的浏览器，不同的呈现。（5）网页需要危险或者和非标准技术的支持（6）网站被黑

（7）在Google搜索结果中首次排在17页之后（8）访问者致信投诉网站失灵。

3.何时重构？我

重构的第一个时机出现在进行任何重新设计之前。

4.重构什么？

（1）XHTML

XHTML是HTML简洁、清晰的一个变体，它不容许犯错，XHTML虽然编写难度大，但

是却简化了XHTML的处理，现在接力棒已经从浏览器到了编写者的手中。

1.程序处理XHTML要比处理HTML容易得多。

2.跨浏览器的兼容性

3.在页面中加入未来的新技术。

（2）CSS

1. 把表现从内容中分离出来是HTML的基本设计原则。

2. HTML文档应该侧重表达的是文档的意义而不是文档的表现。

3.CSS布局更精美、更少偏差、更高效、更好用、页面加载变得更快、显示的更好。

4.CSS让每个人在互不干扰的情况下各司其职发挥所长。

5.页面变得非常简洁。

6.可以组合通用的样式，并保持整站一致的观感。

（3）REST（Representational State Transfer表述性状态转移)

REST是HTTP的架构。

1.所有的资源通过URL定位。

2.通过GET进行诸如查询或者浏览等安全无副作用的操作。

3.通过POST进行诸如购买或者添加评论等非安全的操作。

4.各个请求彼此独立。

5.重构HTML的步骤：

（1）确定问题（2）修复问题（3）检验已修复的问题（4）确保没有引入新问题（5）部署解

决方案

**二.良构**

1.概念：良构是一个来自XML的概念。从技术角度而言，它表示文档遵循某种严格的约束。

2.将名称改为小写：

（1）动机：XHTML只使用小写的名称。 所有元素和属性的名称都要小写

（2）风险：转换为小写并没有太大的风险。

（3）做法：1.手动修改 2.使用Tidy或TagSoup或者正则表达式 3.使用能够处理大小写的替换工

具，如Perl或者BBEdit.

3.给属性值加引号：

（1）动机：在XHTML中，所有的属性值，即使没有空格，也都要加上引号。

（2）风险：浏览器绝对不会因为属性值被用引号括起来而产生混淆的情况了。（当然每一个属

性值添加的引号会给文件增加两个字节的大小，如果是Google或者因为服务器每

秒就有上千兆字节的流量请求而需计算主页的每一个字节，才需要考虑是否加引

号）

（3）做法：为属性值前后加上单引号或双引号。

4.补足遗漏的属性值：

（1）动机：XHTML不支持只有属性名称而没有值的语法。

（2）风险：很小。

（3）做法：使用Tidy或者TagSoup.

5.将空标签替换为空元素标签：

（1）动机：XML解析器要求所有的起始标签都有匹配的结束标签，还可以使用空元素标签的语

法。

（2）风险：大部分现代浏览器对空元素标签的处理不存在问题，但是在一些旧版本浏览器会有

问题。

（3）做法：1.写成起始标签和结束标签对，如：<br></br>

2.写成空元素标签，如：<br/>(方法2与1比，2对旧版的浏览器和编写者更友好些）

3.在最后一个斜线之前加上属性或者空格。（能达到最大的兼容性）如：<br

class="empty" />

6.添加结束标签:

(1）动机：1.XML的兼容性2.很多传统的HTML文档在遗漏结束标签时，并不能按照预想的方式

去显示。不正确关闭的元素，CSS的创建和排错都非常困难。

（2）风险：非常小。最终可能时文档容量稍微变大。

（3）做法：1.手工处理。2.Tidy和TagSoup都可修复此问题，但是它们通常猜错起始标签的位

置，Tidy虽然不会关闭空的段落，但是也找不到第一个段落的开始位置。可以使用

把值为y的 --enclose-block-text选项把一块独立的文本包入段落中; $ tidy -asxhtml

--enclose-block-text y endtag.html这些对于基本的浏览器呈现没多大影响，但是

如果对p元素 赋予了特定的样式规则，则影响非常大。进一步说，本想为某一章或

某一节的第 一段应用特别格式，但却应用到了第二段。因此，在一个页面上运行

TagSoup之后，务必搜索下一个空的段落。只要发现了一个，意味着文档前面有一

块非段落 的文本，它应该放入新的p元素中。

7.消除重叠：在父级元素上关闭每一个子元素

(1）动机：不同的浏览器对包含重叠元素的文档所构建的DOM树是不一样的，因此JS在不同浏

览器间的运作不一致，JS/CSS以及其他需要读取DOM的程序都会变得难以创建、排

错和维护。

（2）风险：有时候文本本身避免不了重叠的情况。

（3）做法： Tidy和TagSoup可以修复重叠问题，但效果不是那么好。

8.把文本转换为UTF-8

(1）动机：如果页面使用非基本ASCII字符集的内容，很容易出现跨平台的问题。

（2）风险：为了正确显示UTF-8编码，必须控制Web服务器的HTTP响应，这在一些共享的主机

环境中配置起来可能有些麻烦。另外，差劲的工具并不总是能认出UTF-8的。

（3）做法：第一，重编码为UTF-8,第二，把你的编码告诉客户端。

9.转义小于号： &lt

(1）动机：一个未转义的小于号更有可能导致浏览器中的内容被隐藏。

（2）风险：没有。

（3）做法：正则表达式、xmllint或HTML验证工具。

10.转义&号：把&转义为&amp;

动机：如果没有空格会迷惑一小部分浏览器，未转义的&号会对读者隐藏内容。

风险：没有。

做法：搜索工具

11、转义属性值中的引号：把“转义为&quot； 把‘转义为&apos;

动机：在属性值内出现的引号，如果跟作为定界的引号是一样的，就会过早关闭属性值。

风险：没有

做法：手工修复

12.加入XHTML DOCTYPE声明：

动机：DOCTYPE声明指明了DTD，用来处理实体引用。

风险：会导致很多浏览器关闭怪异模式。

做法：为XHTML使用3中DTD：框架型、过度型、严格型

13.结束每一个实体引用：在每一个实体引用后加上分号。

动机：XML要求实体引用以分号结尾。Web浏览器并不是所有情况下都可以自己补充遗漏分号的

实体引用。

风险：没有

做法：搜索分号之前有空格的实体引用即可。

14.替换虚构的实体引用：确保文档中用到的实体引用都是有定义的。

动机：偶然情况下，编写者会使用不存在的实体引用。

风险：没有

做法：直接搜索和替换或者验证器

15.加入根元素：确保每一个文档都有一个html根元素。

动机：XML要求每个文档都有一个独立的根元素。在XHTML中，这个根元素就是html

风险：没有

做法：搜索，然后在开头和结尾加上html即可。

16.加入XHTML命名空间：为每一个html元素添加xmlns="http://www.w3.org/1999/xhtml"属性

动机：命名空间不同，XSLT和其他基于XML的工具可以对同一元素做不同的处理。

XHTML工具在XHTML命名空间下会按照预期找到HTML元素，但是没有命名空间的话可

能会导致功能失效。

很多浏览器只能在正确命名空间的XHTML文档中才能正确操作。

风险：没有

做法：搜索和替换来修复。

**三.有效性**

1.概念： 有效性比良构稍微严格，不仅要确保文档的语法正确，还要保证语义正确。

2.加入过渡型DOCTYPE声明

动机：过渡型DTD让你不必完全语义化标记就能让文档通过验证。

风险：浏览器在使用现有的TOCTYPE或者根本没有使用DOCTYPE的情况下，如果启用浏

览器的怪异模式，添加这个DOCTYPE会导致文件的样式化有所偏差。

做法： 文件搜索和替换

3.删除所有不该存在的标签：消灭所有异数

动机：现代浏览器不再支持大量过时的、不推荐使用的、厂商私有的标签。

风险：一些还依赖这些标签的旧浏览器可能会在页面样式上缺失。

做法：如果发现一个异数，最好是整站搜索并改掉它。

4,添加alt属性：

动机：帮助视力受损的用户；搜索引擎化

风险：添加alt文本没有捷径，需要大量的时间和精力。

做法：验证器、正则表达式3

5.用object替代embed

动机：它允许embed元素使用大量未定义的属性，这让网页无法通过验证；

embed不是任 何HTML规范的组成部分；

object元素可以被更好的文档化，现代浏览器的支持更一致，对载入的内容种类及其

渲染对象更为明确；

跟embed拥有的大量属性相比，object只是很少的一部分；

object有可验证的方式，而embed没有。

风险：有些旧的浏览器，并不认识object标签，补救办法：object元素内加入embed元素。

做法： 验证器、搜索

6.加入严格型DOCTYPE声明：

动机：严格型DTD删除了所有不推荐使用的表现性元素。

风险：严格型DTD的限制很多，习惯使用的大量元素和属性都不再允许使用。

做法：第一个需要做的改动是把DOCTYPE声明指向严格型DTD。如果已经通过了过渡型

DTD的验证，只需简单搜索和替换。

7.用CSS替代center：

动机：严格型XHTML不再允许center元素，因为居中表达的是呈现而非语义。

风险：很旧的浏览器可能不支持CSS规则。

做法：可以为所有指定类型的元素赋予这个属性 text-align:center;

8.用CSS替代font:

动机：严格型XHTML不再允许font元素，因为它表达的是呈现而非语义。

风险：很旧的浏览器可能不支持CSS规则。

做法：验证器

9.用em或CSS替代i

动机：严格型XHTML不再允许em元素，因为它表达的是呈现而非语义。

风险：很旧的浏览器可能不支持CSS规则。

做法：使用严格型DTD验证。

10.用strong或者CSS替换b

动机：严格型XHTML不再允许strong元素，因为它表达的是呈现而非语义。

风险：很旧的浏览器可能不支持CSS规则。

做法：使用严格型DTD验证。

11.用CSS替换各种颜色属性：

动机：严格型XHTML不再允许各种颜色属性，因为它表达的是呈现而非语义。

风险：很旧的浏览器可能不支持CSS规则。

做法：把原有的HTML属性值复制到CSS属性中或者升级为HTML4或CSS的颜色名称。

12.把img元素的属性转换为CSS属性：

动机：严格型XHTML把与img元素相关的大部分属性都转换为css属性了。

风险：很旧的浏览器可能不支持CSS规则。

做法：注意在转化为css时，padding-left 和padding-right需要设置为hspace，padding-top

和padding-bottom需要设置为vspace.

13.用object替代applet

动机：严格型XHTML不再允许applet元素

风险：没有一种object语法能运作在所有的浏览器中。补救方法：嵌套使用object元素，并

使用条件注释组织IE看到第二个object元素。

做法：验证器

14,用CSS替换表现性元素：

动机：严格型XHTML不再允许这些表现型元素

风险：很旧的浏览器可能不支持CSS规则。

做法：使用严格DTD验证。

15.把行内（Inline）元素放到块（block）元素中

动机：要做到有效，文档中所有的元素不但要有有效的严格型XHTML元素，还必须保证它

们之间的正确关系。

风险：没有

做法：大多数块元素可以嵌套。p只能包含纯文本或者其他行内元素。pre只能包含行内元

素。

行内元素通常可以包含其他行内元素。

一些杂项元素如object、ins既可以当行内元素，也可以当块元素。

一些上下文相关的元素只能出现在相应的父元素中。

**四.布局**

良构和有效性检查是基本的语法约束，接下来要保证语义是恰当的。

1.替换表格布局：

（1）三种比较普遍的布局：

两栏布局：左边是一个固定宽度的侧栏，右边是一个弹性宽度的内容栏。

两栏布局：右边是一个固定宽度的侧栏，左边是一个弹性宽度的内容栏。

（有三种可能的宽度：两栏宽度都固定；左栏宽度固定，内容栏是百分比宽度；两栏都是

百分比宽度）

三栏布局：左右各有固定宽度的侧栏，中间是内容。

（2）两栏，侧栏在左：

#content{

float:right;

width:80%;

}

#pages{

float:left;

width:18%;

}

为了解决ie6的错误，需要添加：

\*html{

left：18%；

}

两栏，侧栏在右：

#content{

float:left;

width:80%;

}

#pages{

float:right;

width:18%;

}

三栏布局：左右各有固定宽度的侧栏，中间是内容。

body{

margin：0；

}

#content{

width:100%;

position ：relative;

float:left;

}

#pages{

margin-left:-100%;

position:relative;

float:left;

}

#content{

margin-right:-100%;

position:relative;

float:left;

}

2.使用CSS定位替代框架：把框架转化为单一页面

移除非框架->使用div取代框架->使用css的width和height取代cols和rows。->使用css

的overflow替代scrolling->完成的无框架文档。->使用css替换框架重现原始布局

3.把内容放到最前面：

把主要内容放到最前面，侧栏、页头、页脚、导航栏等部分跟在内容之后。

4.正确标记列表

5.用CSS替换blockquote/ul的缩进：

对于只是为了缩进才使用的blockquote或ul元素，把它们转为div， 然后使用CSS

的margin-left或者margin -right属性进行缩进操作。

6.替换占位图片：

把所有用于占位的GIF都删除，必要时使用CSS的margin/padding/position/indent属

性来复现这些效果。

7.添加id属性：

为每个元素赋予唯一的id，以便可以快速定位。

8.为图片添加width 和height属性;

为所有没有width和height属性的img元素添加这两个属性。

**五.可访问性：**

1.把图片转换为文本：

使用图片本身包含的文本取代图片，然后使用标记和CSS规则模拟相关的样式。

2.为表单输入框添加标签：

对于非隐藏的input/textarea/select 等表单元素，确保它们都有相应的标签。

3.使用标准的字段名称：

重写表单和处理表单的脚本，为如电子邮件地址、信用卡号、电话号码等使用惯用的名

称。

4.开启自动完成：

在适当的情况下把autocomplete="off"删除。

5.为表单添加Tab索引：

为每一个可编辑的、非隐藏的表单字段添加tabindex属性。

6.加入“跳过”机制：

在每个页面的开头放置一个“不可见”的链接，它可以直接跳转到页面的主要内容上。

7.添加内部标题：

对于比较长的网页，在合适的位置放置h1到h6元素。

8.把链接和标题里重要的内容放到前面：

在每个标题和链接里，最重要的、最突出的内容放到前面。

9.加大输入框：

保证所有的表单都有足够的空间，让用户更容易填写所有必要的字段。不确定空间的话。

更多的空间比更少的好。

10.加入表格描述：

为每个表格添加caption元素和/或summary属性。给每一行或每一列的标签使用th元

素。caption元素包含更完整地描述表格的标记文本，只可以包含行内元素。

summary属性包含描述表格的纯文本。所有人都可以看到描述文本，一般只有屏幕阅

读器才会注意到summary属性。

11、加入acronym元素：

给缩写词（abbreviation）和字首组合词（acronym）分别包一层abbr和acronym标签。

12.加入lang属性：

给每个根元素添加lang和xml:lang属性，用于指出文档使用的主要语言，在用其他语言编写的

文档中的任何元素，也应该添加值为该语言的lang和xml:lang属性。

**六.Web应用程序：**

在web2.0的大潮中。很多网站已经变成数据录入、字处理、日历管理、人力资源、游戏和其

他你所能想象粗的功能的应用程序。

1.用POST替代不安全的GET。

通过GET访问的URL可以是而且应该是用于爬虫、预抓取、缓存、重用和其他自动访问

上。不安全的操作（如确定订购、下订单、达成协议、删除页面等）应通过POST来执

行，避免没有显示用户请求和同意的情况下发上以外的操作。

2.用GET替换安全的POST：重新设计安全的操作，通过GET而不是POST来访问数据。

GET形式的URL让用户可以使用后退键； GET形式的URL对搜索引擎更友好，可以提高

搜索引擎排名。 GET形式的URL这种表示法是可以缓存的。

3.重定向POST为GET:

把响应POST请求的脚本重新设计为重定向到浏览器，而不是直接执行结果。

4.启用缓存：

启用缓存控制头，可以在恰当的时候缓存动态生成的页面。

5.阻止缓存：

应用缓存控制头，让那些迅速变化的页面总是可以从服务器载入最新的版本。

阻止缓存的一个做法是使用POST处理所有请求。这是错误的解决方案，正确方案是在不应

缓存的页面内发送HTTP缓存控制头。

6.使用ETag：

为Web应用程序生成的静态页面提供ETag.

7.用HTML替换Flash.

把Flash网站转为HTML，为所有的Flash内容提供一个纯HTML的版本。

8.增加Web Forms 2.0类型

为表单的文本输入框添加属性

type="email/url/date/time/datetime/localdatetime/month/week/number",识别预期 的

数据类型。

9.用mailto链接取代联系表单：

把联系表单改为实在的mailto链接。

10.封禁机器人：

在希望封禁机器人的目录中加上robots.txt文件。

11.转义用户的输入：

把所有用户提供的数据都进行转义。

**七.内容：**

1.修正拼写：

使用拼写检查器检查所有的文本。

2.修复坏链接：

尽可能修复所有坏链接，修复不了的就删除掉。

3.移动页面：

重新组织URL结构，对开发者、访问者、搜索引擎更透明，但是要保证就得URL能够继续工

作。

4.删除入口页面：

不要浪费访问者的时间。

5.隐藏电子邮件地址：

公布在Web页面上的电子邮件地址应该进行编码，防止垃圾信息机器人的采集。