软件形式化方法

学院：软件学院　　**姓名：**刘智文　　**学号：**12016000755

在层叠样式表中检测重构机会

1、摘要

层叠样式表（CSS）是用于描述的 HTML 文档的外观和格式的语言。 CSS 已经广泛应用于网络和移动开发实践，因为它能够清晰地分离内容和演示。语言表现出复杂的特点，如继承，层叠和特异性，使 CSS 代码难于维护。因此，找到提高 CSS 代码的可维护性的方法很重要。在本文中，我们提出了一种自动化的方法来消除重复的 CSS 代码。更具体来说，我们开发了检测三种 CSS 声明的重复，并建议重构的机会，消除这些重复的技术。我们的方法使用的前提条件是保证重构的应用程序将保存原文件的造型。我们评估我们的技术在 38 个当今网络中的 Web 系统和总共 91 个 CSS 文件。我们的研究结果表明，CSS 代码中的重复是较为常见的。此外，有大量的都有保留重构机会，可以减少 CSS 文件的大小和增加代码的可维护性。

2、前言

层叠样式表（CSS）是用于描述的 HTML 文档的外观和格式的语言。CSS 广泛应用于当今的 Web 开发，超过 90%的 Web 开发人员在 90%的网站中使用 CSS。CSS 也越来越多地用于移动应用开发，通过生成混合移动应用的框架。因此，CSS 已经成为在许多不同领域中的应用的重要语言。

虽然 CSS 有一个相对简单的语法，它的一些复杂的功能，如继承，级联和规范，这将

使开发和维护 CSS 代码繁琐的任务交给开发人员。 此外，由于缺乏既定的设计原则和有效的工具支持，CSS 开发远不是严格和严格的过程。一个不规范的开发实例是通过复制和修改现有代码而不是重用已经定义的代码来定义新的 CSS 规则。

有经验证据表明，用程序或面向对象语言开发的软件系统中的重复代码与增加的维护效率，更高的错误倾向以及在改变频率和新近度方面更高的不稳定性相关联。我们认为 CSS 代码的开发和维护也会遇到与代码重复相同的问题。此外，重复的问题甚至可能在 CSS 代码更强烈，因为 CSS 语言缺乏许多功能可用于其他编程范例，可以启用代码重用。例如，在 CSS 中没有变量和函数的概念来构建可重用的代码块。

此外，CSS 代码必须通过网络从服务器传输到多个客户端。大量的代码重复增加了传输数据的大小，导致大的网络负载开销。一旦在客户端，CSS 代码必须由 Web 浏览器处理。大量的代码重复增加了必须由浏览器处理的 CSS 代码的大小，导致计算开销可能显着考虑移动设备中可用的有限计算，存储器和功率资源。以前的研究表明，通过分析 CSS 代码执行的网页的可视化布局消耗浏览器的平均处理时间的 40-70%。

在本文中，我们提出了一种自动化技术，（1）分析和检测各种类型的 CSS 重复，（2）发现和引用重构机会，以根除重复的 CSS 代码。 这项工作做出以下主要贡献：

1、我们定义 CSS 代码中的各种类型的重复，并提出一种检测重复的技术；

2、我们提出了一个重构可以消除 CSS 代码重复机会的推荐技术。此外，我们提供了一种基于每个建议重构可能实现的大小减小的排序机制，通过关注具有更高影响的重构来帮助 CSS 开发人员优化其维护效率；

3、我们描述在应用重构之后保留 CSS 样式的前提条件；

4、我们执行一个研究来评估我们的方法的效率使用 38 个当今网络中的网站，用到总共 91 个 CSS 文件。

我们的结果表明，CSS 代码中的重复程度确实非常强烈，从绝对大多数被检查的 CSS 文件的 40%到 90%。 平均来说，我们在经过检查的 CSS 文件中找到 165 个重构机会，其中 62 个可以通过保留网页的样式以应用。 最后，通过保留呈现的重构来实现的平均尺寸减小为 8%，而最高减少为 35%。

3、CSS 语言

CSS 是万维网联盟（W3C）批准的第一个标准以及标记语言 HTML。 CSS 通过断开样式代码与标记之间的关系分离。 因此，相同的样式表可以应用于不同的标记，从而能够重用样式代码。 在本文中，我们将应用样式的 HTML 页面称为目标文档。 样式表可以写在目标文档内部，也可以写在与目标文档链接的单独的外部 CSS 文件中。 CSS 文件由一组 CSS 规则组成，语法如下图所示：

```
selector {
    property_1: value_1;
    property_2: value_2;
    ...
    property_n: value_n[;]
}
```

CSS 规则中的选择器的部分定义应当应用样式规则的目标文档的元素。例如，P 选择器选择文档中的每个段落元素<p>。属性：值被称为样式声明。声明定义由相应选择器选择的元素的指定属性的样式值。例如，声明颜色：blue; 将对所选元素的文本应用蓝色。每个选择器中可能有多个样式声明。

3.1、CSS 选择器

可以使用 ID 属性为目标文档中的元素设置 ID。例如，HTML 标签<div id =“toolbar”

/>对应于 ID 等于"toolbar"的 div 元素。 #toolbar CSS 选择器（称为 ID 选择器）可用于选择具有此 ID 的 DOM 元素。

此外，一组声明可以定义为 CSS 中的类。然后可以将相同的类应用于许多不同的元素，从而避免声明的重复。 如下图显示了一个类选择器的示例：

HTML	CSS

content
 	
content
 	
	.class1 {
	color: red;
	font: 10pt tahoma;
	}

还有可能在 CSS 中分组不同的选择器，使用“，”字符。 例如，如果我们要对所有 h1 和 h2 HTML 元素应用相同的样式，我们可以使用 h1，h2 {/ * declarations * /} CSS 规则。

我们还可以向选择器添加属性条件。如果选择器 S 选择一组元素 S，则格式 S [attr operator value]的选择器选择 S 的子集元素，在目标文档中，属性 attr 被定义并设置为带值的子串。运算符可以定义子串的条件。例如，[target]选择所有<a>具有设置为任何值的目标属性的元素，而 img [src $ = “.png”]选择所有具有“.png”后缀的<img 元素 src 属性值。

伪类可以用于过滤由给定选择器选择的元素。 例如：: not（div）选择除 div 元素之外的所有元素。 还存在结构伪类，例如 tr：nth-child（2n + 1），其选择目标文档的每个表中的每个奇数行。 伪元素创建关于目标文档中元素的抽象，超出了 HTML 标准规定的元素[34]。 例如，p::first-line 选择器只选择每个<p>元素中文本的第一行。

最后，我们可以结合各种选择器来实现更具体的选择器，使用不同的组合器。 假设我们有两个选择器 A 和 B，我们可以如下组合它们：

A B（后代组合器）选择由 B 选择的所有元素，它们是由 A 选择的元素的后代；

A > B（子组合器）选择由 B 选择的所有元素，它们是由 A 选择的元素的直接子元素；

A ~ B（一般兄弟组合器）选择由 B 选择的所有元素，其具有由 A 选择的元素作为兄弟；

A + B（相邻兄弟组合器）选择由 B 选择的所有元素，其直接在由 A 选择的兄弟元素之前。

3.2、继承，级联和规范

目标文档中的元素以分层方式组织。 对于一些特定的元素属性，CSS 通过利用这种层次结构支持值的继承。 例如，如果我们应用颜色：蓝色; 到<body>元素，<body>标签的所有子元素都将用蓝色样式。

可能有不同的 CSS 规则选择相同的 HTML 元素。如果这些选择器将值分配给相同的属性，则 Web 浏览器只能将其中一个属性值应用于所选元素。在这种情况下，Web 浏览器遵循级联起源规则。基于这些规则，创作的样式规则（即，由 web 应用开发者编写并链接到目标文档的规则）将覆盖用户样式规则（即，网站的读者可能已经定义的规则浏览器）。类似地，用户样式规则覆盖浏览器的默认样式规则。在选择器的原点相同（例如，连接选择器在同一 CSS 文件中）的情况下，规范确定“获胜”选择器，即，更具体的选择器优先于较不具体的选择器。如果两个选择器的原点和指定是相同的，CSS 文件中的选择器的位置决定了获胜选择器（即，最后一个选择器覆盖先前的选择器）。还可以在声明的末尾添加！important 注释，以允许选择器绕过特定规则并始终应用。

4、CSS 中的重复

基于两个代码片段的文本或功能相似性，已经为过程和面向对象代码[28]定义了不同类

型的重复。 由于缺乏可用于构建可重用的代码块的变量和函数，因此代码重复问题在 CSS 中预计会更有效。 因此，许多常见的样式声明必须在多个选择器中重复。

4.1、重复类型

在这项工作中，我们专注于重复的 CSS 声明，这只能通过更改 CSS 代码来避免。 此外 ,通过消除这种重复 ,我们可以减少必须通过网络传输并在将来保持的 CSS 代码的大小。因此，我们在本节中的声明级别定义三种不同类型的重复。

类型 I：对于给定属性，具有词汇相同的值的声明

类型 I 重复的一个极端例子可以在图 3 中看到 ,它来自 Gmail 的收件箱页面的主要 CSS 文件。 在这个文件中，有 23 个声明在三个选择器中重复。 如下图仅显示了两个选择器的这些声明的子集。

```
.z-b-V {
    -webkit-box-shadow:
       0 1px 0 rgba(0, 0, 0, .05);
    box-shadow:
       0 1px 0 rgba(0, 0, 0, .05);
    background-color: #fff;
    color: #404040;
    cursor: default;
    font-size: 11px;
    font-weight: bold;
    text-align: center;
    white-space: nowrap;
    border: 1px solid transparent;
    border-radius: 2px;
    ...
}
```

```
.z-b-G {
    -webkit-box-shadow:
       0 1px 0 rgba(0, 0, 0, .05);
    box-shadow:
       0 1px 0 rgba(0, 0, 0, .05);
    background-color: #fff;
    color: #404040;
    cursor: default;
    font-size: 11px;
    font-weight: bold;
    text-align: center;
    white-space: nowrap;
    border: 1px solid transparent;
    border-radius: 2px;
    ...
}
```

注意，类型 I 复制的定义仅考虑属性值的等同性，而忽略值的顺序。 例如，考虑图 3 中的两个边界声明。如果其中一个定义为 border：transparent solid 1px（即，不同顺序中的相同值），我们仍然将它们视为 I 型重复的实例，因为基于 CSS 规范，浏览器将以相同的方式解释两个声明。

类型 II：具有给定属性的等价值的声明

在 CSS 中，对于具有替代表示的属性，我们可能具有相同的值。 字体大小，颜色，长度，角度和频率值是代表性的情况。例如，下表所示出了相同颜色的替代表示，“紫”颜色的不同表示。我们将所有这些不同的表示值视为等价值。

Representation	Value
HTML Named Color	Violet
Hexadecimal	#EE82EE
RGB	rgb(238, 130, 238)
RGBA	rgba(238, 130, 238, 1)
HSL	hsl(300, 76%, 72%)
HSLA	hsla(300, 76%, 72%, 1)

如果两个或多个声明对于相同的属性具有相同的值，我们将它们视为类型 II 复制的实例。在 a 图我们可以看到 Gmail 收件箱页面的 CSS 文件中，类型 II 重复的示例。 注意，带有 color 属性的声明是重复的。

此外，某些属性有一些默认值，在缺少显式值时的应用。 例如，基于 CSS 规范，声明 padding：2px 4px 2px 4px; 也可以写成较短的版本作为 padding：2px 4px; 具有相同的效果。 这种情况也被视为等同声明，因此构成 II 型重复的情况。

类型 III：一组个体属性声明等同于一个简写属性声明

一些 CSS 属性（如边距，填充和背景）称为速记属性。 使用这些属性，我们可以在单个声明中定义一组属性的值。 例如，可以使用边际速记属性来定义边缘顶部，边缘右边，边缘底部和边缘左边的值，如下图所示：

```
margin: 3px 4px 2px 1px;  ⟺  { margin-top: 3px;
                               margin-right: 4px;
                               margin-bottom: 2px;
                               margin-left: 1px; }
```

如果选择器包含一组单独的声明，这相当于另一个选择器的简写声明，我们将这些声明视为 III 类复制的实例。B 图显示了 Gmail 收件箱页面的 CSS 文件中，类型 III 复制的示例。

```
.fJ {
    ...
    color: white;
    ...
}
.Ik {
    ...
    color: #fff;
    ...
}
```

(a) Type II

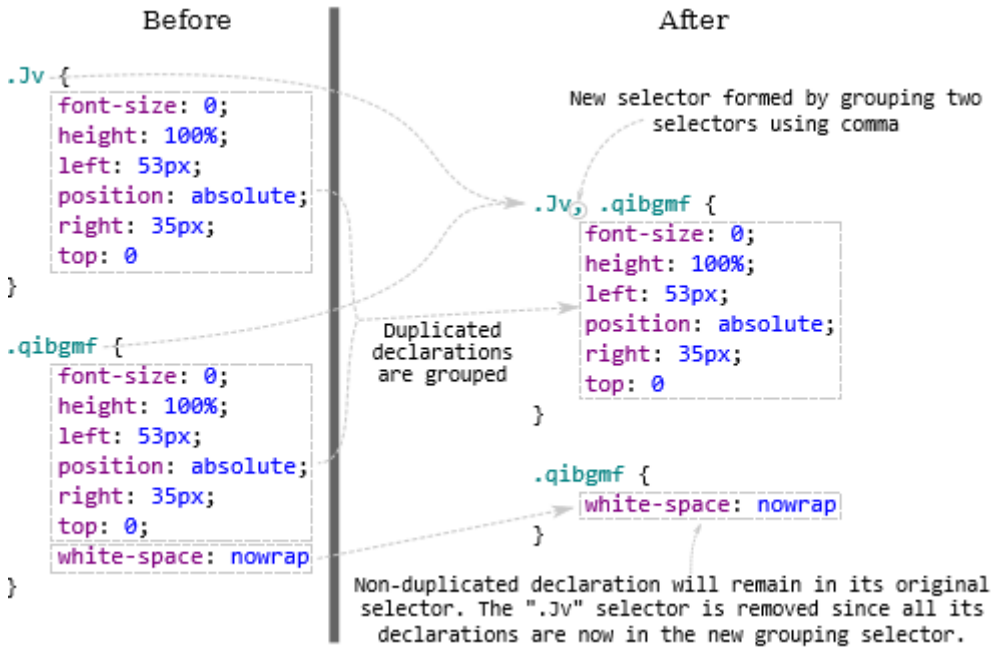
```
.sG0wIf {
    border-bottom-color: #e5e5e5;
    border-bottom-style: solid;
    border-bottom-width: 1px
}

body .azewN {
    ...
    border-bottom: 1px solid #e5e5e5;
    ...
}
```

(b) Type III

4.2、消除重复

上述类型的复制可以直接在 CSS 代码中消除，而不通过提取分组选择器来改变目标文档。 如果在一组选择器 S1，S2，...，Sn 中复制声明集合 D（以类型 I，II，III 复制的形式），我们可以为组 S1，S2 ...创建新的选择器。 ..，Sn 并将 D 移动到此新选择器。 例如，下图左侧的 CSS 代码片段可以重构，下图右侧所示：



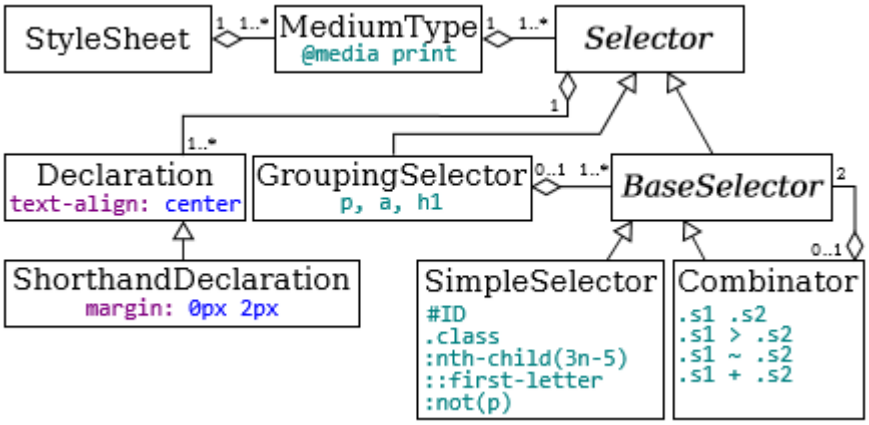
另一个可能的解决方案，再次基于分组，是为重复的声明创建一个公共类，并将该类分配给目标元素。 然而，该解决方案还需要更新目标文档，以便它们利用新定义的类。

5、方法

我们在 CSS 中检测复制和重构机会的提取的方法分为以下讨论的四个主要步骤。

5.1、抽象模型生成

要查找重复，我们首先需要解析给定 Web 应用程序的所有 CSS 代码。 为了支持所有可用的 CSS 标准，我们修改了 W3C Flute 解析器，使其符合 CSS3 规范。 我们的方法然后分析代码并生成下图所示的抽象模型的实例，它表示应用程序的 CSS 代码的高级结构。



如图所示，每个样式表都可能绑定到一些 Medium 类型。 这标识了样式表定义的目标显示介质。 例如，可以区分用于在移动设备中打印和显示的样式。 也可以在给定样式表中为不同的媒体类型定义相同的选择器。

在此模型中，BaseSelector 表示不执行分组的选择器。 SimpleSelector 表示类型选择器或通用选择器（* selector，它选择每个元素）。 类 SimpleSelector 具有指定属性的特殊属性，如元素 ID，类标识符，伪类，伪元素和属性条件。 最后，Combinator 表示组合器选择器，它可以通过组合两个 BaseSelector 来形成。 这些选择器中的每一个的示例可以在图中看到。

5.2、预处理

类型 I 重复的检测不需要任何预处理。然而，为了便于检测 II 型和 III 型重复实例，我们执行三个单独的预处理步骤。

属性值的规范化。在此步骤中，我们用通用的参考格式或单位替换可用不同格式或单位（例如颜色和尺寸）表示的值。例如，命名，十六进制或 HSL 格式的每个颜色都将替换为其等效的 rgba（）值。以厘米，英寸或点指定的每个维度都将转换为其等效像素值。所有应用的转换均基于 CSS 规范提供的指南。用常用表示法替换值称为归一化，并且已在传统的代码克隆检测技术中用于在标识符，文字和类型中找到具有差异的克隆。我们的动机是使用相同属性值的替代格式或单位来声明。这种情况构成 II 型重复实例。

添加缺少的默认值。 正如我们在第 3 节中讨论的，CSS 开发人员有时会省略一些多值属性的值，以便具有更短的声明。 在这一步中，我们有一些基于 CSS 规范的预定义规则，将隐含的缺失值添加到模型中的属性。 例如，margin 属性通常应该有 4 个值。 我们丰富了声明边距：2px 4px，两个缺失的隐含值为 margin：2px 4px 2px 4px。 这允许基于一组完整的显式值的声明的比较，使得能够检测 II 型复制实例。

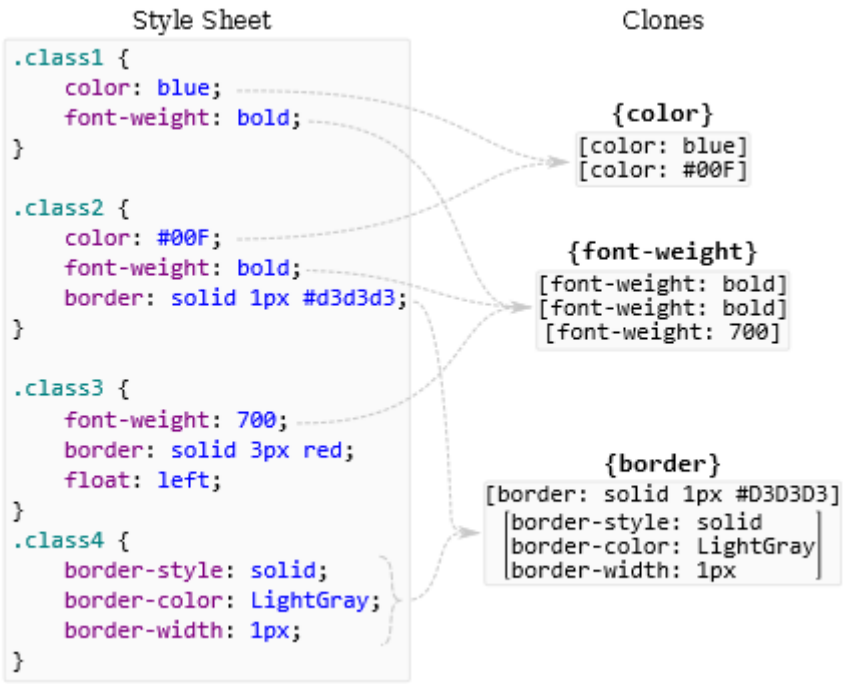
虚拟速记声明。 检测类型 III 复制实例需要在一个选择器中的速记声明与在另一个选择器中的单个声明的等效集合之间的比较。 为了方便这个任务，我们向模型添加了“虚拟”简写声明。 我们检查每个选择器的声明，以找到可以表示为等价简写声明的个体声明集。对于每个这样的单独声明的集合，我们生成相应的简写声明，并将其作为“虚拟”声明添加到模型中的相应选择器。 这些虚拟速记声明将与“真实”速记声明进行比较，以检测类型 III 复制实例。

5.3、重复检测

通过比较 CSS 模型中的每个可能的声明对并检查它们是否相等（对于类型 I）或等效（对于类型 II 和 III），可以找到重复实例。 注意，在我们的方法中，我们考虑在 CSS 模型中存在的声明，因为它们是在预处理步骤之后形成的，以允许检测类型 II 和 III 的重复实例。

算法 1 中总结了我们的检测方法。算法接收预处理的 CSS 样式表作为输入，并返回一组克隆，其中每个克隆是一组等于或等价的声明，D = {d1，d2，...，dn }，并且每个声明属于所分析的 CSS 样式表的选择器 si∈S= {s1，s2，...，sn}。

如图描述了样式表和从算法 1 的应用中提取的相应克隆的示例。第一两个克隆包含 II 型重复的实例，而最后一个克隆包含 III 型重复的实例。



5.4、提取重构机会

如第 5.3 节所定义的一个克隆，可以通过提取单个声明 di∈D 到一个新的选择器来直接重构，该选择器将 S 中的所有选择器分组，然后从它们所属的原始选择器中删除 D 中的

所有声明。 CSS 代码的重构版本将包含| D | - 少一些声明，但在应用于所选元素的样式方面应具有完全相同的效果。 因此，克隆越大（即 D 的基数），对应的重构越有利，因为通过应用重构将消除更多的声明。

检测到的克隆构成用于提取更高级和更高影响重构机会的“构件”。 例如，可能存在具有多个公共声明的选择器（即，涉及多个克隆的选择器）。 在一组选择器之间共享的一组通用声明构成一个克隆集。 在这种情况下，可以将克隆集中的所有声明提取到单个分组选择器（或类选择器）中，从而显着减少声明的重复。在 4.2 小节中提出了这种情况的一个例子。通常，在更大的选择器集合中更多的克隆是共同的，相应重构机会在减少重复声明中的影响越高。