Web应用界面跨浏览器兼容性的自动检测方法

王欢欢 吴毅坚 赵文耘
(复旦大学软件学院 上海 201203) (复旦大学上海市数据科学重点实验室 上海 200433)

摘 要 随着Web应用被越来越广泛地使用,其稳定性也受到开发人员及用户的重视,其中很重要的一项指标是Web应用在不同浏览器中的兼容性问题。为了保证应用在所有的浏览器中都可以正常使用,在开发阶段对浏览器兼容性隐患进行检测就显得非常重要。为此,提出了一个在开发阶段可以自动检测Web应用界面跨浏览器兼容性问题的一项新技术,它可以自动浏览Web应用的所有页面,通过对同一页面在不同浏览器里所提取出的代码结构信息和相关属性的分析,生成差异报告,从而帮助开发人员更快地找到有兼容性问题的元素。完成了该方法的具体实现,并将其应用于一个具体的开发项目中来,收集相关的数据并验证该方法的可行性。最后根据实验数据归纳了常见的Web应用界面兼容性问题。

关键词 自动化,跨浏览器,兼容性,检测
中图法分类号 TP311.5 **文献标识码** A

Automatic Detection Method of Cross-browser Web Application

WANG Huan-huan WU Yi-jian ZHAO Wen-yun
(School of Software, Fudan University, Shanghai 201203, China)
(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 200433, China)

Abstract With the Web applications used more and more widely, the stability of these Web applications has been paid more and more attention. One of the most important issue is the compatibility of different browsers. To make sure that the application can be used in all browsers, it is very important to detect the browser compatibility in the development stage. This paper proposed a new automated technology to detect browser compatibility problems in the development stage, which can automatically browse all the pages of the Web application. After the analysis of the extracted information structure code and the related property, we got a report of cross browser compatibility problems for the developers to help them solve these questions faster. This paper implemented the method and applied it to a specific developed project to collect the relevant data and demonstrated the feasibility of the method. In the end, this paper analyzed the common Web application compatibility issues.

Keywords Automation, Cross-browser, Compatibility, Detection

1 引言

Web应用是随着Internet的发展而兴起的软件产品,相较于传统的软件,它具有只需要一个浏览器就能自由使用的优势。如今的生活中,Web应用也已经得到越来越广泛的使用,用户可以在不同的终端上依据自己的习惯和偏好选择不同的浏览器。但分散的浏览器给开发人员带来了一个很现实的问题,就是如何使项目在不同的浏览器中能够保持一致的表现。不同浏览器的内核在渲染、解析和调用代码上的标准各不相同,这使得在一种浏览器中能够运行的代码就不一定能在另外一种浏览器中运行[1]。现有的很多Web应用存在不同程度上的跨浏览器兼容性问题。Web应用在不同浏览器上的兼容性差异主要体现在各浏览器对CSS、JavaScript以及Ajax、Flash等插件的解析和支持程度不同,从而反映为界面显示和功能的差异。

现有的对浏览器兼容性的研究主要集中在对Web应用的自动化检测上,自动化测试具有开销最小、耗时短、具有一致性、可重复等特点[2]。现有的工具主要有3种类型:1)捕获屏幕的截图工具:捕获不同的浏览器,或者不同平台下的网站项目的截图,然后比较其差异;2)取网页DOM信息:浏览所有浏览器下的网页,抓取网页的DOM信息,对比结构差异;3)提供仿真环境的工具:它允许开发人员能和特定环境、特定平台下的Web应用程序互动,发现并解决问题。

本文提出了一个新的自动检测Web应用界面跨浏览器兼容性问题的技术,通过基准浏览器在运行容器中浏览所有的页面,生成操作序列的测试脚本。在运行容器中的所有浏览器依次运行该脚本后即可获取不同页面的快照,对其比较后即可获得差异报告,帮助开发者更快地发现兼容性问题。本文完成了该方法的具体实现,并应用于一个具体的开发项目中,收集相关的数据并验证该方法的可行性。最后根据获

王欢欢(1989—),女,硕士,主要研究方向为软件产品线,E-mail:12212010017@fudan.edu.cn;**吴毅坚**(1979—),男,博士,副教授,CCF会员,主要研究方向为软件体系结构、软件复用和产品线等;**赵文耘**(1964--),男,硕士,教授,CCF会员,主要研究方向为软件复用、电子商务。

取的实验文章归纳了常见的 Web 应用界面兼容性问题。

本文第 2 节分详细阐述了相关研究工作的现状；第 3 节介绍了基于网页快照的 Web 应用界面浏览器兼容性检测方法；第 4 节将该测试工具用于具体的实验项目中，并对获得的数据结果进行分析；最后是结论和对未来的展望。

2 相关工作

根据网页在不同浏览器中的截图是否一致来判断 Web 应用界面是否一致是最直接的检测方法，相关的学术上的研究也较多：Choudhary 等人抽取网页的 DOM 信息和页面的屏幕截图，再结合屏幕截图做针对网页外观的视觉分析，根据网页外观的差异找到相应的浏览器 DOM 信息并报告给开发人[3,4]。Semenenko 等人使用机器学习方法判断浏览器中显示的页面截图是否一致，并向检测者报告出现显示差异的截图及对应代码[5]。但这种根据页面截图找到对应的 DOM 结构差异的过程较为繁琐，且不同的浏览器对 Web 应用解析后的 DOM 结构基本一致，因此获取到的差异会比较少。

相对于界面的兼容性差异，验证 Web 应用的功能也是现有的一个测试方向：Mesbah 等人将浏览器在每次用户操作后的页面显示作为一个状态节点，建立起一个有限自动机，然后再对生成的模型成对地做等价性检验以展现可观察的功能差异性[6]。边耐政等人实现的 Web 自动化测试低耦合框架则通过录制脚本、回放脚本来简化测试人员的工作[7]。IBM 公司推出针对图形用户界面的自动化测试软件：RFT，其能够实现自动化录制的功能和回归测试，通过页面能否通过脚本的回放来验证 GUI 的完善性[8]。

现有的 Web 应用的自动化测试一般都是基于一些自动化测试工具[9,10]。Sahi 是一个 Web 自动化测试工具，它通过注入 JavaScript 来获取页面中的元素，而且它还可以录制脚本，自动播放。相对于 Selenium[11,12]、用 Ruby 实现的 Watir[13]等工具，Sahi 内置了智能的页面等待机制，能够自动判断 Ajax 请求是否已经处理完毕。但是以往的工作都是利用 Sahi的自动化功能实现跨浏览器的功能验证，我们可以利用其功能实现 Web 应用跨浏览器的页面兼容性测试，它可以实现对 Web 应用操作正确、一致的捕捉与回放，并允许其他用户对已有操作记录进行定制[14,15]。

3 基于网页快照的 Web 应用界面浏览器兼容性检测方法

3.1 方法预览

基于网页快照的 Web 应用界面浏览器兼容性检测方法由生成测试用的操作脚本、在多浏览器上执行操作脚本、分析页面快照并生成差异报告 3 个步骤组成。在生成操作脚本步骤中，采用 Sahi 作为容器运行被测 Web 应用，获得 Web 应用的页面操作序列，形成测试操作脚本。在执行操作脚本步骤中，将基于 JavaScript 的页面探针工具引入到被测 Web 应用中，然后在待测浏览器中依次打开被测应用，并回放测试操作脚本。在各个浏览器回放过程中，页面探针 JS 代码将自动获得每个浏览器中 Web 应用的页面快照，并按页面分类整理，以便后续进行跨浏览器的比较。最后，利用启发式规则对各个页面在多个浏览器下的快照进行比较后，生成带有不同差异类型的页面元素差异报告，供开发人员调试参考。

该方法的整体流程如图 1 所示，详细步骤将在后续部分中介绍。其中，页面快照是指在某一浏览器下某个页面在加载完毕后，记录页面上所有的标签及其重要的属性信息的文本文件。

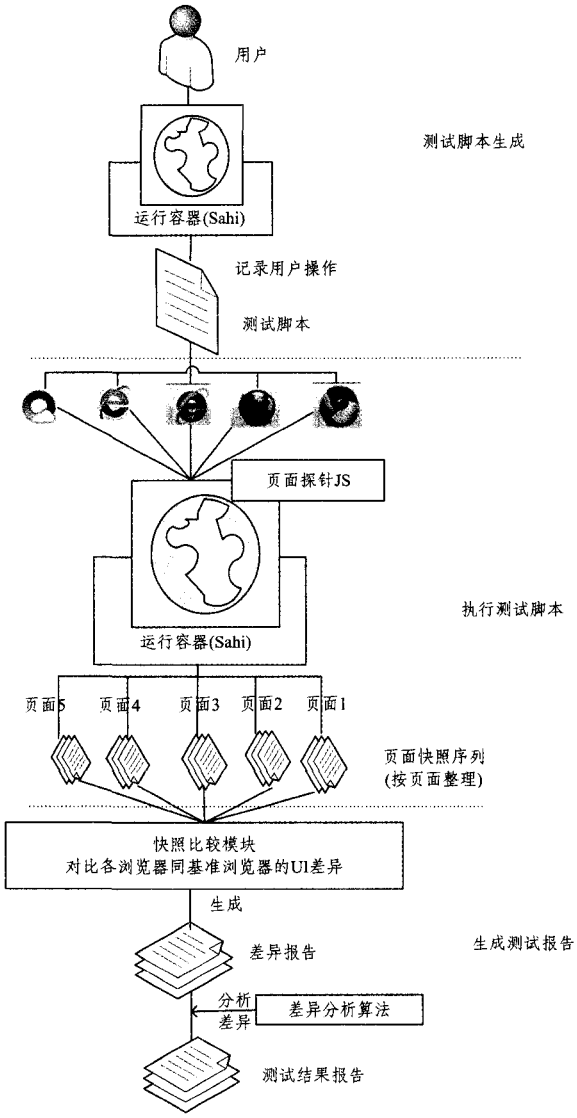


图 1 基于网页快照的 Web 应用界面浏览器兼容性检测方法预览

本文测试的浏览器有 Chrome、Firefox、IE10、IE9、IE8、QQ、360 安全浏览器 6.0。其中 Chrome 使用了开源 chromium 内核，Firefox 使用了 Gecko 内核，IE8、IE9、IE10 则分别使用了 Trident4.0，5.0，6.0 的内核，360 安全浏览器 6.0 版为 Trident＋Webkit 内核，QQ 浏览器搭载了 IE 内核，因此，以上浏览器能很好地覆盖主流的浏览器内核。鉴于苹果公司已不再维护 Windows 下 safari，本文暂不考虑 safari。

3.2 生成测试脚本

为了在多个浏览器中自动获取 Web 应用各个页面的元素及属性，通过在基准浏览器上进行手工操作的方式，获取被测 Web 应用的操作序列作为浏览器兼容测试的操作脚本。该脚本将用于在多种浏览器中自动地对被测 Web 应用进行相应的操作。由于存在浏览器的兼容性问题，每个页面元素在浏览器中的位置可能存在差异，因此脚本中对于页面元素的定位是基于元素的 id 等信息来唯一识别，而不是根据其在屏幕/页面上的绝对位置。

采用 Sahi 作为运行容器，在基准浏览器上录制被测 Web 应用的操作序列，得到测试操作脚本，表 1 所列为一个测试操作脚本的片段。该脚本以函数语言的形式记录了对哪些元素（URL、文本框、按钮、超链接等）进行了哪些操作，用于在对多浏览器的测试过程中自动化地浏览页面。

表 1　测试操作脚本片段

```
_navigateTo("http://localhost:8080/selfstudying");
_setValue(_textbox("idno"),"min@sj. sh");
_setValue(_password("password"),"sjmr");
_click(_submit(0));
_click(_link("开课申请"));
_click(_link("新增课程"));
_click(_span("下一步"));
```

3.3　页面快照获取

页面快照的获取分为两个步骤：首先获取页面信息，生成页面快照并存储于本地；其次，将生成的所有快照按页面整理，即不同浏览器在一个页面下的所有快照放在一个文件夹里，便于比较和查看。

当生成测试脚本后，在每个加载完毕后的页面中引入页面探针，该探针能够获取页面在各个浏览器下的所有标签及其信息。对于页面的标签信息，从以下两个方面来获取：1）标签的唯一标识 xpath 用来记录标签在页面中的路径，同时，有的标签的兼容性问题会影响到父级元素的显示样式，考虑元素的父子关系更有利于我们分析兼容性问题；2）标签的属性信息，由于浏览器兼容性最直观的差别是控件的长宽、在页面中的位置等信息，因此获取了 clientWidth、clientHeight、clientLeft、clientTop、offsetLeft、offsetTop 这 6 个属性，而标签的 scrollWidth、scrollHeight[16] 等其他属性对于在页面显示上没有直观差异，因此暂不考虑。表 2 详细描述了需要获取的 6 个属性。

表 2　属性信息说明

属性名	属性值
tagName	标签名称
clientWidth	对象可见的宽度，不包滚动条等边线，会随窗口的显示大小改变
clientHeight	对象可见的高度，不包滚动条等边线，会随窗口的显示大小改变
clientLeft	元素左边框的宽度，如果不指定一个边框或者不定位该元素，它的值就是 0
clientTop	元素上边框的宽度，如果不指定一个边框或者不定位该元素，它的值就是 0
offsetLeft	对象元素边界的左侧相对于 offsetParent 的水平偏移量
offsetTop	对象元素边界的上侧相对于 offsetParent 的水平偏移量

通过递归的方式获取页面的所有标签。在获取标签时不需要考虑以下标签：1）不占据页面空间的元素：script、style、head、title、meta、link、base、br[17]，这些标签无具体的显示，不会占据页面空间，因此将其排除以精简页面快照的大小。2）加载顺序不同的元素：option 元素，在 chrome 中 option 是懒加载，因此在页面快照中它的高度、宽度都为 0；而在 Firefox 中，页面加载的同时会加载 option，因此会获取到其具体的高度、宽度值。但这种差异是不影响页面显示的，所以将其排除以提高准确率。

当获取到所有页面的标签信息后即可生成一个页面快照。每一个页面快照都需要有一个唯一标示的名称来标示它是哪个浏览器下的哪个页面。由于页面 title 是唯一一个可以识别是哪个页面的信息，因此将快照名称设为“浏览器名称＋页面 title”。表 3 所列为页面快照的格式样式。

表 3　页面快照格式

```
{ pageTag:"Chrome_登录-松江区学生自主学习平台"
{"tagName": "HTML", "clientWidth": 1349, "clientHeight": 667, "clientLeft":0,"clientTop":0,"offsetLeft":0,"offsetTop":0,"xpath":"HTML"}
{"tagName": "BODY", "clientWidth": 1349, "clientHeight": 667, "clientLeft":0,"clientTop":0,"offsetLeft":0,"offsetTop":0,"xpath":"HTML/BODY"}
...
}
```

在页面快照进行本地化存储时，由于不同的浏览器对于本地文件的操作权限不一样，因此需要区分不同的浏览器类型。表 4 列出了不同类型浏览器实现本地文件存储功能的伪代码。

表 4　不同类型浏览器实现本地文件存储功能的伪代码

```
if(IE 内核)) {
  创建文件系统组件:new ActiveXObject("Scripting. FileSystemObject");
  创建文件. CreateTextFile(文件路径);
  写入文件
} else {
  在页面中创建一个 a 标签;
  设置标签的 href 属性
href="data:application/octet-stream;base64,"+Base64. encode(content);
  设置标签的 download=存储的文件名;
  点击 a 标签
}
```

对于使用 IE 内核的浏览器，提供了 ActiveX[18] 组件实现本地文件的操作。用 ActiveXObject 函数创建文件系统组件，CreateTextFile 函数可以创建一个本地文件，再将获取到的所有标签信息写入，生成页面快照。

对于非 IE 内核的浏览器，没有直接的本地文件操作方法，为此，使用了浏览器的下载功能将页面快照存储到本地。首先创建一个隐藏的 *a* 标签，*href* 属性值代表为以流文件形式对数据进行 base64 编码存储，*download* 值则为要存储的快照名称。当 *a* 标签创建后，添加一个点击事件，以下载方式实现文件存储的目的。

为了能直观地看到每个 Web 页面在不同浏览器下的快照，以及方便地将其他浏览器与基准浏览器界面进行差异对比，当所有浏览器的快照都生成后，利用 IE 内核的文件系统操作组件对其按页面整理：将同一个页面下在不同浏览器下的所有快照放到以“页面 title”命名的文件夹内。表 5 给出了文件整理实现的伪代码。

表 5　文件整理实现的伪代码

```
创建文件系统组件:new ActiveXObject("Scripting. FileSystemObject");
获取所有的页面快照;
For(一个页面快照 in 所有页面快照){
得到该快照的页面 title;
If(有名为页面 title 的文件夹){
  将页面放入文件夹内;
  If(文件夹内已有该文件的副本){
    保留最新版本的快照,删除旧快照
    }
}else{
  若没有则创建名为页面 title 的文件夹;
  将页面放入文件夹内;}
}
```

3.4　页面快照的匹配与比较

当所有浏览器的页面快照都获取到之后，将同一页面在不同浏览器下的快照进行匹配。此时需要设置一个阈值来控

制标签的匹配程度，阈值可以为绝对值，也可以为百分比。把两个标签的匹配程度分为以下 3 种。

Equal：如果两个标签的所有属性值都相等，则认为这两个标签是一致的；

Warning：如果两个标签的所有属性值差值都小于阈值且至少有一属性差值大于零，则认为这两个标签基本一致，可以忽略，不用报告给开发人员；

Error：如果两个标签的某个属性值差值大于阈值，则认为这两个标签在显示上存在兼容性问题，需要报告给开发人员。

对于阈值的设置，使用启发式规则，根据项目的实际情况设置多个阈值：

1. 对大部分元素来说，需要设置一个绝对值或百分比作为普通元素阈值，该阈值可以作为大部分标签的匹配标准；

2. 对于宽度和高度容易受到浏览器实际大小影响的元素，如 html、body 等，则需要设置浏览器元素阈值；

3. 对于一些粗粒度元素，如宽度为整个页面宽度的 90% 以上的元素，在不同浏览器之间相差的像素值大于标准阈值时也不会有明显的兼容性问题，需要设置粗粒度元素阈值。

在查看了大部分的页面快照后，发现同一 Web 应用界面在不同浏览器下的 DOM 节点数目和结构信息是一致的，因此当对两个页面快照中的元素匹配时，只要读取两个快照中行数相同的数据，即可获得不同浏览器下的同一元素。表 6 列出了读取两个页面快照中相对应的两个标签，并对其进行匹配的源代码。函数 get_max_attribute_diff 用来获取标签中所有属性差值的最大值。函数 set_matching_level 用来判断每对标签的匹配程度，如果这对标签为 HTML 或 BODY 标签，则使用浏览器元素阈值，对于粗粒度的元素，使用粗粒度元素阈值；对于其他标签，使用普通元素阈值。最后根据匹配级别设置不同的 class 值，标示不同级别的警示。

表 6　标签匹配实现

```
function get_max_attribute_diff(标签 1,标签 2) {
var max=0;
var attribute;
  for(标签的某个属性 in 标签的所有属性)
{
  if(Math. abs(标签 1 与标签 2 的该属性值的差) > max)
  max=Math. abs(标签 1 与标签 2 的该属性值的差);
  attribute=属性名称;
}
  Return {max:max;attribute:attribute};
}
function set_matching_level(标签 1,标签 2) {
      var result=get_max_attribute_diff(标签 1,标签 2). max;
      Var attribute=get_max_attribute_diff(标签 1,标签 2). attribute;
  if(标签 tagName=='HTML'||标签 tagName=='BODY'){
   standard 的值为浏览器元素阈值
  }else if(标签为粗粒度元素){
   standard 的值为粗粒度元素阈值
  }else{
        standard 的值为普通元素阈值
      }
  if(result==0) {
      设置为 Equal;
  } else if(result <=standard) {
      设置为 Warning;
          高亮 attribute 属性;
  } else {
      设置为 Error;
          高亮 attribute 属性;
}
}
```

4　实验结果和分析

4.1　实验系统及实验结果

将该工具应用于一个已开发完成的自主学习平台系统中。该平台包含了如 layer、ckeditor、jwplayer、uploadify 等常用的主流的插件，具有上传文件、富文本框编辑、播放视频等常用的功能。该项目包含 42 个非重复页面。首先使用 Sahi 运行容器在基准浏览器 IE10 下浏览了 29 个代表性页面，生成测试操作脚本，并在这些页面中分别引入页面探针 JS。之后分别用运行容器打开 Chrome、Firefox、QQ、360、IE8、IE9、IE10 运行测试操作脚本并生成页面快照。IE 内核的浏览器最后运行脚本可以实现页面的整理，形成页面序列。最后快照比较模块将页面快照自动进行比较，形成差异报告。截取报告片段如图 2 所示。差异报告中会统计每个浏览器和基准浏览器的比较结果：标签总数、匹配数目(Equal)、警告数目(Warning)、差异数目(Error)，在差异报告中：

深灰色的部分为“Error”标签，该标签中存在着差值大于阈值的不可忽略的差异，表示其在对应浏览器的页面显示上存在着较大的差异。

浅灰色的部分为“Warning”标签，该标签中存在着差异但是可以忽略，表示其在对应浏览器的页面上的差异基本无法辨别。

白色的部分为“Equal”标签，表示两个标签完全匹配，且在不同浏览器的显示上完全一致。

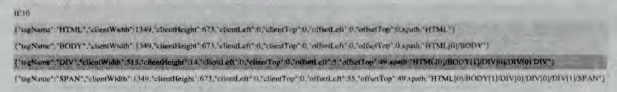


图 2　差异报告片段

实验中，该应用在每个浏览器下都获取 5793 个标签元素，当设置浏览器元素阈值为 10px，粗粒度元素阈值为 8px，普通元素阈值为 5px 时，实验结果结果如表 7 所列。其中，匹配标签数为测试浏览器与基准浏览器的所有匹配程度为 Equal 的标签数目。警示标签数为测试浏览器与基准浏览器中匹配程度为 Warning 的标签数目。差异标签数为测试浏览器与基准浏览器中匹配程度为 Error 的标签数目。兼容性差异数是对差异标签数(Error)去重后的数目，即把由于子元素差异而导致兄弟、父元素出现的差异归为一个差异。有效兼容性差异数目是指：根据兼容性差异能在界面中找到相应界面问题的数目。有效性是指可以找到对应的界面问题的兼容性差异数占总数目的比例。

表 7　实验结果统计

待测浏览器	标签总数目	匹配标签数目	警示标签数	差异标签数	兼容性差异数	有效兼容性差异数	有效性
Chrome	5753	1316	3710	727	125	118	94.4%
Firefox	5793	1193	3564	996	165	143	86.7%
IE8	5793	5790	3	0	0	0	/
IE9	5793	5785	8	0	0	0	/
360	5793	1306	3771	816	114	108	94.7%
QQ	5793	5609	169	15	11	10	90.9%

从上述结果可以看到：

1. 该方法能报告给开发人员有价值的兼容性问题。由于被测应用中表格较多，且当表格的某个 td 出现高度或宽度差异时，相应的 tr、tbody/thead、table 都会出现差异，因此差异标签数目较大，但是去重后的兼容性差异数较少，不会给开发

人员报告冗余信息。在报告的兼容性差异数中，能找到对应页面的兼容性问题的平均有效性高于90%。

2. 不同的浏览器对界面的支持差异也不同。该应用在IE10的显示与IE8、IE9的差异非常小，而与Firefox的差异最大。Chrome以及采用Chrome内核的360浏览器的表现比较接近。国内常用的QQ浏览器尽管采用IE内核，但也和IE10存在少量的差异。

3. 该实验中存在10%左右的假阳性数据。这些假阳性数据和浏览器样式有很大关系。用运行容器打开时，默认都是最大化状态，但是每个页面在浏览器最大化状态下的宽、高都是有差异的。对于html、body标签，尽管使用了浏览器元素阈值，仍会找到一些不匹配的情况。其次，快照中会有一些隐藏的标签，当这些标签存在差异时，对界面显示无影响，但仍然会出现在差异报告中。

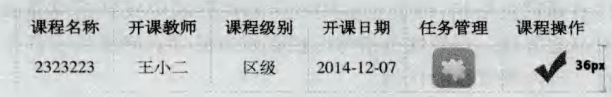
4. 对于阈值的选取，根据不同的项目有所不同。由于该Web应用是固定布局，因此获取的页面标签属性值是绝对值。设置的阈值也是绝对值，本次实验的普通标签阈值为5px，通过多次的调整阈值，当设置为1px至4px中的某个值时，都会出现一定的假阳性数据，兼容性差异数的有效性会降低；当大于5px时，则无法检测出很多明显的浏览器兼容性问题。

4.2 实验分析及讨论

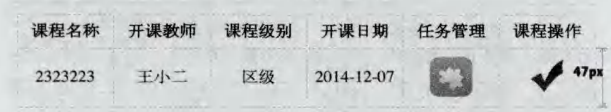
通过对以上实验数据的分析，根据报告中的兼容性差异标签找到对应的界面，对应用中出现的差异进行分类统计，并把找到的常见的显示差异进行了分类总结。

4.2.1 不影响显示的界面差异

在该实验项目中，有部分页面内有表格，表格的每行都有图片，表格在不同浏览器中的位置、显示差异较小，但会因为行内图片的原因存在行高不一致的情况，如教师课程界面在Firefox和IE中存在如图3所示的问题。在Firefox中表格里的图片撑高了该行，导致整个表格偏高，可以看到IE和Firefox中的两个表格的td高度差11px，相应的tr高度也相差11px。这种差异在表格行数较少时不会影响界面显示，但当表格行数达到一定数量时，在Firefox中会出现滚动条，而IE10中则无。这是最常见的界面兼容性问题，在该应用的差异报告中占了43.1%，显示上细微的差异影响了美观，对于Web应用的影响最小。



(a)IE10中的显示效果



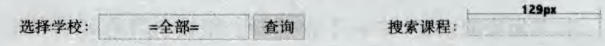
(b)Firefox中的显示效果

图3

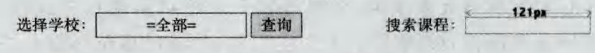
4.2.2 存在潜在问题但不影响显示的界面差异

在该实验项目中，每个有列表的页面都有搜索框，它们在不同浏览器页面中的位置、显示差异较小，但是当输入的搜索数据过长时，可能会有潜在的数据换行、溢出或错位问题。如学校统计页面在Chrome、Firefox、IE10中存在着图4所示的差异，页面中的搜索框在页面的位置基本无差异，但是输入框的长度存在差异，可以看到输入框的长度分别为139px，129px，121px。当输入的数据过长时，在Firefox中会出现换行问题，而在Chrome和IE中数据会溢出。这种差异在该Web应用中出现的概率达26.8%，出现的主要原因是没有定义div控件的宽度，或input输入框的size，或控件受父元素的影响而拉长或缩短。在其潜在的问题出现后，有可能会变形，进而影响其他元素的显示。

(a)Chrome中的显示效果



(b)Firefox中的显示效果

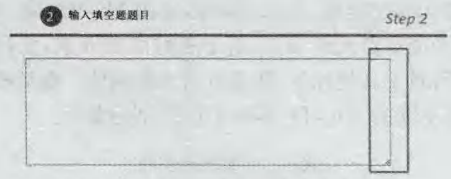


(c)IE10中显示效果

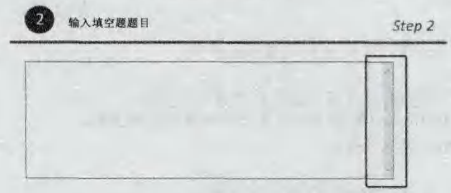
图4

4.2.3 影响显示的界面差异

不同浏览器对于控件的解释也不一致。部分控件需要CSS的兼容处理。教师添加填空题的界面上在Chrome和IE10中存在着如图5所示的差异。由于页面使用了Textarea控件，虽然该控件在页面中的位置没有差异，但在Chrome中不存在、右侧的滚动条，而在IE中即使行数没有超过Textarea的高度，仍然存在滚动条，影响了界面美观。该应用中由于浏览器对控件的解释不一致出现的界面兼容性问题的概率为15.3%，可以通过CSS hack技巧或引用不同的CSS文件来解决。



(a)Chrome中的显示效果



(b)IE10中的显示效果

图5

4.2.4 存在潜在问题且影响显示的界面差异

表格是展示数据的很好的控件，但是当表格中某一格内数据异常时，会对整个表格的显示有很大的影响。该实验中，有超过50%的界面差异和表格显示有关。如开放课程学习页面在Chrome和IE10中存在着如图6所示的兼容性差异。该页面中使用了表格插件对所有的课程进行展示，由于没有直接设置表格列的宽度，其列宽和行高都是自适应的，因此会出现图片换行现象，严重影响界面美观。图中基本每一行的高度、每一列的宽度都存在着差异。控件错位是一种比较多见且严重的界面兼容性问题，在该应用的差异报告中占了12.4%。在使用div+css页面布局或项目使用流动式布局时，更易出现这类差异。

(a)Chrome 中的显示效果

(b)IE10 中的显示效果

图 6

结束语 本文介绍了一种基于网页快照的 Web 应用界面浏览器兼容性检测方法，该方法基于自动化测试工具来实现项目中大部分页面的自动浏览，具有轻便、易用的优势。将获取到的不同页面在不同的浏览器里的快照进行比较，通过多次的实验调优，给开发人员提供有意义的支持来解决兼容性问题。该方法在常用的几个浏览器中被证明有效。

文中的自动化运行容器使用开源的 Sahi 工具，其存在部分不符合本文要求的功能，因此在后面的工作中，我们可以做一个更适用于本实验的自动化运行容器。其次，可增加一些更友好的提示给开发人员。鉴于本方法是基于引入 JS 文件到项目中的方法来实现的，可实验的 Web 应用比较少，之后会考虑如何通过注入的方式实现在非开发应用中的使用。最后，本方法中浏览器页面快照是在网页加载完成之后获取的，若页面存在着局部刷新或异步加载部分，则无法对其兼容性进行验证，未来工作中会使用延时等方法完善此功能。

参 考 文 献

[1] Dallmeier V, Burger M, Orth T, et al: WebMate: a tool for testing Web 2.0 applications[C]// Proc. Workshop on JavaScript Tools(JSTools 2012). 2012: 11-15

[2] Fan Fu-xing, Huang Da-qing, Zhou Mo. Research and implementation of automated testing framework based on Web[J]. Electronic design engineering, 2012(20): 36-38

[3] Choudhary S R, Versee H, Orso A. WEBDIFF: Automated identification of cross-browser issues in web applications[C]// 2010 IEEE International Conference on Software Maintenance (ICSM). 2010: 1-10

[4] Choudhary S R, Prasad M R, Orso A. CrossCheck: Combining Crawling and Differencing to Better Detect Cross-browser Incompatibilities in Web Applications[C]// 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation(ICST). 2012: 171-180

[5] Semenenko N, Dumas M, Saar T. Browserbite: Accurate Cross-Browser Testing via Machine Learning Over Image Features[C]// Proceedings of the 28th International Conference on Software Maintenance(ICSM). IEEE Computer Society, 2014: 528-531

[6] Mesbah A, Prasad M R. Automated cross-browser compatibility testing[C]// Proceedings of the 33rd International Conference on Software Engineering. 2011: 561-570

[7] Bian Nai-zheng. A low coupling Web automation test framework based on Selenium [J]. Computer applications and software, 2014, 31(8): 13-16, 37

[8] Zhang Fan. The implementation of WEB automation test design based on RFT software [D]. Wuhan: Huazhong University of Science and Technology, 2013

[9] Qing Xin. Web UI automated testing based on Selenium [D]. Guangzhou: South China University of Technology, 2012

[10] Gang Xiao-ming. Reasonable application of software test automation [J]. Computer Applications and software, 2010, 27(8): 172-174, 214

[11] Wu Ling-lin. The research and application of software test automation based on Selenium [J]. Computer and Modernization, 2013, 1(2): 65-68

[12] Xing Wei-chao, Gao Xiao-tong. Design and implementation of Webpage test automation framework [J]. Computer Applications and Software, 2012, 29(9): 167-170, 211

[13] Kong Ying-hui. Research and implementation of network page real-time information acquisition based on Watir [J]. Computer Applications and Software, 2014, 31(5): 103-105, 144

[14] Chen Xiao-yu, Huang Zhen, Liu Xuan-zhe, et al. A Chrome browser based user capture and playback tool [J]. Computer Science, 2014, 41(11): 112-117

[15] Ye Xin-ming, Liu Liang. The design, implementation and comparison of Web performance testing replay browser [J]. Computer Science, 2006, 33(9): 58-60

[16] Ma Yong-heng, Xiong Qian-xing, Yang Jine, et al. Study on design method of W3C XML Schema mode [J]. The Research and Application of Computer, 2006(5)

[17] Wang Qing. Research and development of a Web automation testing framework Watir [D]. Wuhan: Wuhan University of Technology, 2011

[18] Zhang Zhen. Research and implementation of ActiveX control operation support in Firefox [D]. Dalian: Dalian University of Technology, 2008

(上接第 424 页)

是开源云技术都已经具有了规模服务的技术和能力，云平台的性能、管理和安全是云计算研究领域下一步值得研究的课题。

参 考 文 献

[1] Laurent S, Andrew M. Understanding Open Source and Free Software Licensing[M]. USA: O'Reilly Media, 2008

[2] Free software for cloud computing[OL]. http://en.wikipedia.org/wiki/Category:Free_software_for_cloud_computing

[3] Moreno-Vozmediano R, Montero R S, Llorente I M. IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures[J]. Computer, 2012, 45(12): 65-72

[4] Daniel N, Wolski R, Grzegorczyk C, et al. The eucalyptus open-source cloud-computing system[C]// 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009(CCGRID'09). 2009: 124-131

[5] NIST Cloud Computing Reference Architecture[OL]. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505

[6] OpenStack Security Domains[OL]. http://docs.openstack.org/security-guide/content/ch005_security-domains.html#ch005_security-domains-idp39040