**关注”千锋互联”微信公众号( ID: qianfengjiaoyu ),更多学习资源(电子书/面试题/简历模板/视频/公开课等等)与你分享**



## CSS 的调试流程与方法

我经历过许多 CSS 代码的调试工作，有别人写的也有自己写的，有移动端平台的也有标准桌面浏览器的，从陈旧的 IE 到最新的基于 Webkit 的每日构建。经验告诉我，很多人并没有一个标准的 CSS 调试流程。

我发现在大多数情况下，拥有专业的解决问题的方法，能够节省花在 bug 上的时间。

下面是我总结的经验。

我不保证这是最适合的调试 CSS 的方法，但是确实对我很有效。如何 CSS 不是你的主要编程语言，调试它可能就像暗黑艺术一样；遵循下面的指南能够帮助你更有效地定位和解决 bug。

概括地说，我把调试流程分为 3 个阶段：

* **评估并快速修复**
* **还原和重现**
* **定位根源并修复**

我们挨个解释每个阶段并实践一个例子。

**一、评估并快速修复**

如果 CSS 是你的主要工作语言，或者你对 CSS 有一定的理解和实践经验的话，解决 CSS 问题就有很多简单的方法，否则的话，方法就少一些。

有经验的 CSS 开发者可能都知道的一些 CSS 陷阱：

* 图片周边存在有趣的空白？设置 display: block（图片默认是内联的，因此会有空白）。
* 元素排列不正确？你可能有浮动的元素。
* 绝对定位元素不显示、位置错误或者被遮挡？你可能没有设置父元素的 position 属性或者用 transform 及 opacity 创建一个 z-index 上下文。
* 伪元素不显示？你可能忘记了设置 ‘content’的值。

**这样的 “bug” 有一大堆。实际上根本没有 bug，更多的是开发者缺少对浏览器行为的理解。更准确地说，是 CSS 代码让浏览器怎么做。**

对这些 CSS 特性熟悉的开发者能够快速定位到问题并且修复。他们对 bug 的认识与那些对 CSS 不了解的人会产生分歧。这样在解决 CSS bug 中对‘工作流’需求的重要性的认识就会因人而异。

对于‘快速修复’中没有覆盖的陌生问题，在开发者工具中靠猜来解决问题的方式已经没什么价值。即使运气好问题被解决了，也很难判断出问题到底是怎样被解决的。

**如果出现的问题不能被轻易解决，先确定问题区域的范围，抓取 HTML 标签（也就是拷贝 DOM），进入下一个调试阶段：还原和重现。**

专业提示：大多数浏览器的开发者工具会让你选择包裹元素并拷贝 HTML 区块。在 Chrome 的开发者工具中，要连同包裹元素一起拷贝，需要点击 ‘Copy > Copy OuterHTML’。

**二、还原和重现**

本阶段的 CSS bug 修复在类似 [Codepen](http://codepen.io/) 的帮助下异常简单。我们目的主要是复现出此问题 - 也就是引起 bug 的代码。这能让我们快速定位 bug，直捣黄龙。

**为清晰起见，只把相关的 HTML 和 CSS 提取出来复现问题。你既可以手打 HTML 对应的 CSS，也可以复制真实的代码。如果可能的话，不用把所有 CSS 代码一股脑拷贝过去重现问题，保证最精简的要素即可。保持逐步增加 CSS 的习惯，问题就会自己找到你。**

在快要接近真相时，往往只需要一个特殊的 CSS 属性的改变就能让 bug 暴露出来。

相对应的做法是，把所有 CSS 都扔进入复现问题，然后每次移除一点，直到问题出现。在实践中，我发现这略笨，不过也因人而异，你可能有不同的见解。

**逐步地增加或删除 CSS 代码已经是重现问题和定位故障的固定套路了。（先从代码块排查，然后逐行排查）**

**那么 HTML 标签呢？**

假设使用最少 CSS 代码复现问题时，效果有如原始代码一样。这也是有用的，我们现在看 HTML 标签。

第一件事要做的，也是不能跳过的，就是检查标签的有效性。即使报告出我们不关心的问题（比如 meta），至少能保证它不会以某种方式破坏美感。我们希望能发现未闭合的标签、没有引号的属性，以及其它任何可能影响浏览器解析的问题。建议你使用 W3C validator。

一旦标签检查通过，将有助于消除浏览器引入意外样式的可能性。这样做：

首先，把所有元素改成 div（块级元素）和 span（行内元素），保证它们只被 CSS 的类选择器选中。也许有必要把额外的选择器移除，如把 a.link 改为 .link。

通过使用固定的标签我们消除了浏览器针对特定元素引入默认样式的可能性。表单元素是个特例（马上会在例子中见到）。

如果把所有元素改成 div 和 span，问题消失了，那么浏览器引入默认样式的嫌疑就被确定了。现在在 computed styles 面板中寻找浏览器增加了什么样式，想办法覆盖它。总之就是要看计算后的样式。

**三、定位根源并修复**

如果简化 HTML 标签也没有找到问题，并且是可稳定复现的，那么就该换个浏览器试一试。是否同样的问题出现在 Chrome，IE，Safari 和 Firefox 上？如果不是，哪个的表现是正确的？如果只有一个浏览器是错的，那么就值得去搜索一下对应的 bug 跟踪系统了：

* [Chrome bugs](https://bugs.chromium.org/p/chromium/issues/list)
* [Firefox bugs](https://bugzilla.mozilla.org/)
* [Safari bugs](https://bugs.webkit.org/)
* [Microsoft bugs](https://developer.microsoft.com/en-us/microsoft-edge/platform/issues/)

是某浏览器的问题吗？或者是某浏览器的特定版本的问题？问题是否在修复中？有没有不影响其它浏览器的解决方案？实在不行你可以为特定的浏览器编写修复代码吗？

[过去](https://benfrain.com/reporting-browser-bugs-chrome-firefox-safari-teams-gets-fixed/)我曾详细描述过如何向浏览器提 bug，在 2011 年 [Lea Verou 也写过一份描述提 bug 流程的文章](https://www.smashingmagazine.com/2011/09/help-the-community-report-browser-bugs)。

另一种情况是可能需要‘无害的’hack。例如，我最近遇到的一个场景是在一个块级元素后面的元素必须是绝对定位的才能显示出来。 left: 100% 只有在 IE 浏览器（移动端是Windows Phone 8，8.1 和 10）中不生效。IE 中在两个元素之间总有一个空隙。看起来像是一个亚像素渲染问题，因此left: 99.99% 就能解决问题而不会影响其它浏览器。这是个 hack 手段，但我们知道原理（有的浏览器会舍入，其它则不会），标注在 CSS 的注释中，没有任何危害。

微软的 [Greg Whitworth](https://twitter.com/gregwhitworth) 告诉我了关于亚像素舍入的更多细节。WebKit 和 Blink 内核舍入 1/64，Gecko 内核舍入 1/60，Edge 舍入 1/100（感谢 Webkit 开发者 [‘smfr’](https://twitter.com/smfr)）。

**计算后样式**

开发者工具中比较容易被忽视的是 computed styles 面板。如果你对 computed styles 不熟悉的话，顾名思义，就是真正应用到元素上的样式。这很重要，因为你写的样式不一定会生效。同样，你写的样式也不是**所有**生效的样式。下面的例子将解释我的意思：

<fieldset class="outer">

<div class="inner">

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

<label for="" class="item"><span>hello</span></label>

</div>

</fieldset>

对应的 CSS 是：

.outer {

max-width: 400px;

}

.inner {

width: 100%;

overflow-x: auto;

overflow-y: hidden;

-ms-overflow-style: -ms-autohiding-scrollbar;

-webkit-overflow-scrolling: touch;

white-space: nowrap;

}

.item {

display: inline-block;

width: 100px;

}

outer 的宽度会是多少？如果你认为是 max-width 的 400px，我会原谅你的。但是不是我们看到的宽度，看 Ben Frain 编写的 [codepen](http://codepen.io/benfrain/pen/mPYqYd/)。

什么情况？为什么不是 max-width 的值？给你个思路，打开 Computed Styles 面板。

**找到问题的根源了吗？**

我来给你解释下。默认地，fieldset 元素的宽度会等于其内容的宽度。在 Chrome 的 Computed Styles 面板中，min-width 的值是一个新的 min-content。

给 min-width 设置一个新值来“修复”它。这个例子中，min-width: 0 就会让 max-width 按照我们期望的那样进行工作。

这正是开发者工具的 Computed Styles 面板中看到的值。**记住你写的代码不一定是浏览器计算后的。**

**讨论**

页面出现异常的原因可能很多并且不尽相同。不同浏览器对规范的实现存在差异是普遍存在的现象。相比于编写一个疯狂的浏览器 bug 目录，解决问题的最有效流程还是始终保持条理性。总结来看有效的措施包括：

* 评估 bug，执行快速修复
* 使用最少的代码重现问题
* 利用工具和 bug 追踪描述原因
* 使用更灵活的代码修复问题，或者使用注释过的hack手段，亦或拷贝副本修复

**关注”千锋互联”微信公众号( ID: qianfengjiaoyu ),更多学习资源(电子书/面试题/简历模板/视频/公开课等等)与你分享**

