**[Web开发人员常犯的10个错误](http://www.codeceo.com/article/10-mistakes-web-developer-make.html" \o "Web开发人员常犯的10个错误 - 码农网)**

2015-09-21    分类：[程序员人生](http://www.codeceo.com/article/category/programmer" \o "查看 程序员人生 中的全部文章)、[首页精华](http://www.codeceo.com/article/category/pick)[1人评论](http://www.codeceo.com/article/10-mistakes-web-developer-make.html#comments)

分享到：更多4

**本文由**[**码农网**](http://www.codeceo.com/)**– 小峰原创翻译，转载请看清文末的转载要求，欢迎加入技术翻译小组！**

说到开发一个运行在现代网络中的网站：Web开发人员需要选择虚拟主机平台和底层数据存储，准备编写HTML、CSS和JavaScript用的工具，要有设计执行方式，以及一些可用的JavaScript库/框架。在将任务分解为这几步之后，接下来要做的就简单多了，可以去网上找文章，浏览论坛，看看那些能提供更好的Web体验提示的示例。

然而不管是走哪条路，犯错却是每一个开发人员都不可避免的。虽然有些错误与某一个具体的行为相关，但有些错误却是所有Web开发人员都需要面对的挑战。因此，通过研究，体验和观察，我总结了Web开发人员常犯的10个错误——以及如何避免这些错误。



以下要点没有特定的顺序。

**1）写一些过时的HTML**

错误：早期的互联网比起我们现在，标记的选择要少得多。然而，旧习难改，现在很多开发人员写的HTML就好像还身处20世纪一样。举例来说，我们使用<table>元素用于布局，当其他特定语义标签更适合的时候使用<span>或<div>元素，在当前HTML标准不受支持的时候使用<center>或<font>标签，在页面上间隔项目，如果有大量&nbsp;字符实体的话。

影响：遵从这种过时的HTML规则可能会导致标记过于复杂，结果是在不同的浏览器中发生不同的行为。而且没有了改进浏览器的动力，因为没有必要更新到最新的浏览器，如Microsoft Edge，哪怕是Internet Explorer版本（11、10、9）也变得没有必要。

如何避免：停止使用<table>元素用于内容布局，限制使用<table>元素来显示表格数据。例如可以去whatwg.org了解当前可用的标记选项。使用HTML去描述内容是什么，而不是说明内容如何展现。对于如何显示内容，请使用CSS（<http://www.w3.org/Style/CSS/>）。

**2）“明明在我的浏览器中是可行的……”**

错误：开发人员往往会偏爱某一个特定的浏览器，或者特别讨厌某一个，可能主要是因为对测试网页视图有所偏见。也有可能是因为从网上找到的代码示例不能保证会如何呈现在其他浏览器中。此外，一些浏览器对风格有不同的默认值。

影响：以某一个浏览器为中心写的网站，在其他浏览器中显示时，其质量将会很差。

如何避免：在开发过程中，在所有浏览器和版本中测试网页是不切实际的。不过，每隔一段时间，在多个浏览器中检查网站的样子不失为一个好方法。现在，不管你偏好的是什么平台，总有免费的工具可用：免费的虚拟机、网站扫描仪。如<http://browsershots.org/>和[https://www](https://www.browserstack.com/show" \t "_blank)[.browserstack.com/show](https://www.browserstack.com/show)网站可以给出页面如何呈现在多个浏览器/版本/平台中的快照。[vi](http://www.codeceo.com/article/vi-editor-guide.html" \o "vi" \t "_blank)sual Studio等工具还可以调用多个浏览器，来显示你正在工作的单一页面。当涉及到CSS设计时，可以参考在meyerweb.com中所示的那样“重置”所有的默认值。

如果你的网站正在使用的CSS特性是专为某一浏览器特制的，那么注意它的引擎前缀，如-webkit-，-moz-和-ms-。对于行业在这方面的发展趋势指导，那么可以阅读以下参考：

* Microsoft Edge开发博客：[A break from the past, part 2: Saying goodbye to ActiveX, VBScript, attachEvent](http://blogs.windows.com/msedgedev/2015/05/06/a-break-from-the-past-part-2-saying-goodbye-to-activex-vbscript-attachevent/?WT.mc_id=13417-DEV-sitepoint-article41" \t "_blank)
* QuirksMode.org：[CSS vendor prefixes considered harmful](http://www.quirksmode.org/blog/archives/2010/03/css_vendor_pref.html" \t "_blank)
* Bruce Lawson： [On Internet Explorer supporting -webkit- vendor prefixes](http://www.brucelawson.co.uk/2014/on-internet-explorer-supporting-webkit-vendor-prefixes/)

上面这些参考资料解说了引擎前缀的革新，以及你还可以点击这里——这个网站提供了一些如何摒弃引擎前缀的实用建议。

**3）不好的格式**

错误：提示用户提供信息（特别是在输入文本字段的时候），并假设数据会如预期接收。

影响：很多事情会（或者很有可能将会）出错，当我们信任用户输入的时候。如果无法提供所需的数据，或接收到的数据不能与下面的数据模式兼容，页面可能会失败。更为严重的是，有的用户可能会故意违背网站的数据库，例如可以通过注入式攻击（见OWASP：Top 10 2013-A1-Injections）。

如何避免：你首先要做的事是确保用户清楚你需要什么类型的数据。比如说，如果你只说要地址，那用户不知道指的是单位，家庭还是电子邮件的地址！除了要具体，还要充分利用现在的HTML提供的数据验证技术。不管数据在浏览器端是如何验证的，确保它始终也在服务器端验证。不要让一个串接的T-SQL语句使用来自于用户输入的，各个字段的类型没有经过确认的数据。

**4）臃肿的响应结果**

错误：页面充满了许多高品质的图形和/或图片，这些图形和/或图片借助img元素的高度和宽度属性按比例缩小。来自于页面链接的文件，如CSS和JavaScript，很大。源HTML标记也可能是不必要的复杂和全面。

影响：完全渲染页面的时间是如此之久，以致于一些用户放弃了，或者甚至于直接不耐烦地重新请求整个页面。在某些情况下，如果页面处理等待太久，会出现错误。

如何避免：不要抱有现在互联网接入越来越快的侥幸心态——从而允许臃肿的场景。相反，要将从浏览器到你的网站的来回当为一种成本。图像是网页臃肿的主要罪犯。为了最大限度地减少图像成本，减轻页面加载的压力，可以试试以下三个技巧：

1. 问问你自己：“这些图形真的有必要吗？”删除不需要的图片。
2. 使用例如Shrink O’Matic或RIOT的工具来减少图像文件大小。
3. 预加载图像。这不会提高初始下载的成本，但可以让网站其他页面图像加载速度更快。

另一种减少成本的方式是压缩CSS和JavaScript链接文件。有很多的工具，如[Minify CSS](http://www.minifycss.com/" \t "_blank)和[Minify JS](http://www.minifyjs.com/" \t "_blank)都能帮你做到。

**5）创建所谓“应该能行”的代码**

错误：无论是JavaScript，还是在服务器上运行的代码，开发人员都需要测试并确认它是否可以正常工作，而不是在部署了之后，就认为它应该就能从一而终地运行。

影响：不经过适当错误检查的网站就是对最终用户耍流氓。不仅会极大地影响用户体验，而且其错误消息内容的类型可能会给黑客线索来渗透这个站点。

如何避免：是人都会犯错，这个哲理同样适用于编码。使用JavaScript，一定要实施好的技术来防止并抓住错误。虽然这篇文章描绘了用JavaScript编码Windows应用程序，但是大部分的内容也适用于web开发，许多提示都很不错！另一种能让代码变得可靠又能在未来变化中存活下来的方法是单元测试。

如果我们够仔细，那么就能捕捉到服务器端的代码失败，而不被用户发现。只显示必要的信息，并且一定要确保设置友好的错误页面，如HTTP 404s。

**6）写分叉代码**

错误：本着支持所有浏览器和版本的崇高理念，开发人员立志创建可对任意可能情况作出回应的代码。代码中if语句成堆，所有方向都有分叉。

影响：随着浏览器新版本的更新，代码文件会变得越来越笨拙和难以管理。

如何避免：实现代码的功能检测和浏览器/版本检测。功能检测技术不仅可以显著减少代码量，还更易于阅读和管理。不妨考虑使用如Modernizr这样的库，不仅有助于功能检测，还能自动帮助提供不能跟上HTML5和CSS3速度的旧版浏览器的反馈支持。

**7）非响应式设计**

错误：假设开发/设计人员在相同尺寸的显示器上开发网站。

影响：当在移动设备或在非常大的屏幕查看网站时，用户体验要么很难看到页面的重要方面，要么甚至要时刻注意着不导航至其他网页。

如何避免：响应式的思维方式。在网站中使用响应式设计。这里有一些关于这方面的实用教程，包括响应式图片，还有一个非常受欢迎的库，那就是[Bootstrap](http://www.codeceo.com/article/bootstrap-web-css-framework.html" \o "Bootstrap" \t "_blank)。

**8）制作无意义的页面**

错误：制作面向公众内容的网页才是有用的，绝不能不提供关于搜索引擎的任何线索。没有实现可访问性功能。

影响：如果不能让搜索引擎发现网页，那么，可能会只有少量或根本没有访问。

如何避免：使用SEO（搜索引擎优化）和HTML的支持可访问性。关于SEO，一定要添加标签以提供有意义的网页关键字和描述。 About Tech就写得很好，可以借鉴。为了能有更好的可访问性功能体验，请对每一个img和area标签提供一个alt="your image description" 属性。更多建议请见[About Tech](http://webdesign.about.com/od/seo/a/keywords-html.htm" \t "_blank)。你也可以在[Cynthia Says](http://www.cynthiasays.com/" \t "_blank)测试公共网页，看它是否兼容Section 508。

**9）网站过多刷新**

错误：创建的网站需要为每一个互动而全面刷新页面。

影响：类似于页面臃肿（参见＃4），页面加载时间的性能会受到影响。用户体验缺乏流畅性，并且每次互动都可能导致网页短暂（或长时间）的复位。

如何避免：快速避免这种情况的一个方法就是，通过测定回发到服务器的内容是否是真正需要的。例如，当不依赖服务器端资源的时候，客户端脚本可用于提供直接结果。你也可以应用AJAX技术或进一步使用单页的应用程序“SPA”方法。流行的JavaScript库/框架，如JQuery、KnockoutJS和AngularJS，能让这些方法的采用变得容易得多。

**10）做了太多的无用功**

错误：开发人员花了很长的时间来创建web内容。大量的时间花在了重复的任务上，或者自己敲代码写了很多。

影响：初始网站的发布和后续的更新时间过于冗长。如果其他开发人员也在在做同样的工作，却用了更少的时间和精力，那么你的开发价值显然就低了。手动劳动很容易出现错误，而排除故障错误需要更多的时间。

如何避免：探索你的选择。在开发的每一个阶段考虑使用新的工具和新的流程技术。例如，你目前使用的代码编辑器相比[Sublime Text](http://www.sublimetext.com/)和[Visual Studio](https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx/?WT.mc_id=13417-DEV-sitepoint-article41)，如何？不管你使用的是什么样的代码编辑器，你最近有好好钻研它的功能吗？也许只投入稍稍一点时间去仔细阅读文档，就可以发现做事的新方法，为今后节省一个又一个小时的时间。例如，在这篇文章中，扩展Visual Studio可以为web开发人员提高生产效率。

不要错过网上可用的帮助工具！例如，检查在dev.modern.ie上的工具以简化测试（跨多个平台和设备）以及排除故障。

你也可以通过实现流程自动化以减少时间和错误。这方面的例子是使用[Grunt](http://gruntjs.com/)工具，例如它的自动化功能可以减小文件（见第4点）。另一个例子是[Bower](http://bower.io/)，可以协助管理库/框架（参见第9点）。

至于web服务器本身？在例如[Microsoft Azure Web Apps](http://azure.microsoft.com/en-us/services/app-service/web/?WT.mc_id=13417-DEV-sitepoint-article41)的帮助下，你可以快速创建一个网站，几乎所有的开发场景都可以很轻轻松松地规模化到你的业务中！

**总结**

通过识别这些常见的错误，web开发人员可以避免很多让其他人饱受煎熬的挫折。我们不仅需要承认错误，还应该清楚错误的影响，并采取措施避免错误，这样才能有更好的开发表现——并有信心完成任务！