一些 HTML 和 CSS 中的坑

发表于 2016-03-06

<!DOCTYPE> 声明

<!DOCTYPE> 会声明一个文档类型给浏览器识别，例如我们经常使用的 HTML5 的文档类型声明是这样的：

```
<!DOCTYPE html>
```

如果你没有为页面声明文档类型，浏览器将会使用 Quirks 模式（兼容模式）去渲染页面上非标准的表格，input 控件和其它元素，这样可能会引发一些问题。

盒模型计算

当元素设置了 padding 或者 border-width 的时候，实际上元素的实际宽度是要比设置的 width 要 宽 的。为了避免这个尴尬的问题，我们可以统一使用 box-sizing: border-box; 来重新设置元素盒模型宽度的计算范围。

rem 单位和 Safari Mobile 的爱恨情仇

虽然 Safari Mobible 已经支持识别属性值的 rem 单位了，但是当你在媒体查询条件中使用rem的时候，会导致页面的文字在不同尺寸下来回跳闪，亮瞎你的双眼。

解决办法是在媒体查询条件中使用em代替rem。

```
html {
font-size: 16px;
}

/* Causes flashing bug in Mobile Safari */
@media (min-width: 40rem) {
html {
font-size: 20px;
}
}

/* Works great in Mobile Safari */
@media (min-width: 40em) {
html {
font-size: 20px;
}
}
```

紧急求助！ 如果你有向 Apple 或者 Webkit 报告这个 BUG，请告诉我，我会把链接附上。因为这个 BUG 只出现在手机端的 Safari 上，桌面版并没有，所以我也不是很确定应该向哪里提 BUG。

高贵的浮动元素

浮动元素最好放置在文档流的开头。因为浮动元素需要内容环绕，否则除了浮动显示内容，它还容易出现一些意想不到的负面效果。

```
<div class="parent">
<div class="float">Float</div>
<div class="content">
<!-- ... -->
</div>
</div>
```

清除浮动

当你使用完float之后，你可能需要清除它。当某元素使用了float之后，附近元素的内容都会环绕着它，只有清除了浮动这个影响才会去除。你可以使用下面的方法清除浮动。

根据 `clearfix` 大法 定义一个样式类，使用它来清除浮动：

```
.clearfix:before,
.clearfix:after {
display: table;
content: "";
}
.clearfix:after {
```

```
7  clear: both;
8  }
```

或者你也可以使用overflow，在父元素上设置overflow:auto;或者overflow:hidden;也可以达到目的。

```
1  .parent {
2  overflow: auto; /* clearfix */
3  }
4  .other-parent {
5  overflow: hidden; /* clearfix */
6  }
```

当然你要小心使用overflow，因为它会给其他元素带来一些不可预料的影响。

重大提示！ 由于样式类可以被覆盖或者增加其他内容，所以你最好在样式类上加上 / clearfix / 注释用以提示你的同伴。

浮动元素计算高度

当父元素仅包含一个浮动元素的时候，此时父元素最终高度是 0。通过在父元素上使用 clearfix 清除浮动之后可以强制浏览器计算父元素的高度为浮动元素高度。

浮动元素就是块级元素

当一个元素被设置了 float 属性，那么它会自动设置 display: block。所以不要同时设置这两个属性，因为 float 会覆盖你的 display 设置。

```
1  .element {
2  float: left;
3  display: block; /* Not necessary */
4  }
```

趣闻： 许多年前，我们不得不给浮动元素设置display: inline;属性避免元素在 IE6 上触发 margin 加倍的 bug。好在现在我们已经不用管这些了。

垂直方向相邻 margin 合并

相邻元素的 margin-bottom 和 margin-top （两个margin是挨着的）一般情况会发生合并，但是对 float 元素和 position: absolute; 的元素是无效的。阅读这篇 MDN 文档，或者这篇 CSS2 规范中的 margin 合并 了解更多。

注意！水平方向相邻的 margin 是永远不会合并的！

给表格的定义样式

除非你给父元素<table> 设置了 border-cllapse: collapse; 属性，否则你的是无法直接设置 border 属性的。当然你还需要知道的是，如果你的<tr>中的子元素 <td>或者<th> 设置了一个和父元素一样大小的 border-width，那么父元素 中的设置将不会生效。无例子无真相，点击这里查看例子。

Firefox 和 <input>按钮的悲欢离合

不知道为什么，Firefox 会自动给<input>类按钮增加 line-height 属性，并且你不能自定义样式覆盖它！针对这个问题你有两个选择：

坚持使用 <button> 元素

在你的按钮上不要使用 line-height

如果你选择了第一个（我也推荐这个因为怎么看使用 <button>也是更好的选择），下面这个是你需要知道的：

```
<!-- Not so good -->
<input type="submit" value="Save changes">
<input type="button" value="Cancel">

<!-- Super good everywhere -->
<button type="submit">Save changes</button>
<button type="button">Cancel</button>
```

如果你选择了第二个，其实你只要不给元素设置 line-height 转而只使用 padding 去达到垂直居中按钮文字也还好啦。大家可以在 Firefox 中浏览示例查看问题表现和解决方法。

捷报！ 这个问题似乎在 Firefox 30 中已经修复了。但是老版本肯定还是会存在的，所以使用上依旧要小心。

Firefox 默认给按钮增加外边框

当按钮获得焦点即 :focus 的时候，Firefox 默认给按钮（包括<input>和<button>）增加了外边框。这个设置如果你觉得很奇怪的话可以使用下面的 CSS 覆盖它：

```
input::-moz-focus-inner,
```

```
2  button::-moz-focus-inner {
3  padding: 0;
4  border: 0;
5  }
```

你可以在之前的例子中看到修复后的实际效果。

重大提示！ 确保为你的按钮，链接和 input 空间增加了获得焦点的状态。为这个状态提供可视化对于一些使用Tab键切换焦点和示例不好的人来说是极为重要的用户体验。

总是给<button>元素设置 type 属性

<button>默认的 type 属性值是 submit，这个意味着如果按钮在表单中点击可以提交表单。使用 type=”button” 可以让按钮在表单中回归本来，而不会触发提交事件。当然如果你想要设置提交事件，设置 type=”submit” 即可。

```
1  <button type="submit">Save changes</button>
2  <button type="button">Cancel</button>
```

当你不在表单中需要使用 <button>元素时，最好也设置一下 type="button"。

```
1  <button class="dismiss" type="button">x</button>
```

趣闻： IE7 是不支持读取 <button> 元素的 value 属性值的。当它读取 value 属性的值的时候，它会去查找元素的 innerHTML 的值并复制给它。然而我为什么没有把这个坑给列出来呢？主要是因为 IE7 的份额在日渐下滑，已经很少人在用它了，另外一个原因也是因为几乎很少人会同时设置 <button>元素的 value 属性和它的 innerHTML。

IE 选择器限制

IE9- 允许一个样式表中最多只有 4096 个选择器。对于页面内样式表<style></style> 也有每页最大 31 个选择器的限制。当然其他浏览器是没有这个问题的。你要么选择分割你的 CSS 样式表，要么对你的代码进行重构。我个人建议选择后者。

下面给一段代码告诉下大家浏览器是如何计算选择器个数的：

```
1  /* One selector */
```

```
/* One selector */

.element { }

/* Two more selectors */
.element,
.other-element { }

/* Three more selectors */
input[type="text"],
.form-control,
.form-group > input { }
```

Position 属性值讲解

元素设置position: fixed; 属性之后，位置是相对于浏览器的视口来设置的。元素设置 position: absolute;属性之后，位置是相对于最近的一个 position 值为非 static （例如 relative, absolute 或 fixed）的父元素而设置的。

Position 和 Width 不得不说的故事

不要对一个设置了 position: [absolute|fixed];, left 和 right 的元素设置 width: 100%;。这里 width:100% 和设置了 left: 0;及right: 0; 的效果是一样的。两者择其一，没必要都用。 Don’t set width: 100%; on an element that has position: [absolute|fixed];, left, and right. The use of width: 100%; is the same as the combined use of left: 0; and right: 0;. Use one or the other, but not both.

position: fixed;和transform 的恩怨是非

当一个 position: fixed; 元素的父元素设置了 transform 属性之后，浏览器会破坏 position: fixed 的解析。使用 transform 属性元素脱离文档流，相当于创建了一个新的内容块，强制父元素的表现和 position: relative; 一样，强制 position: fixed; 的元素的行为表现和 position: absolute; 一样。