怪异模式（Quirks Mode）对 HTML 页面的影响

Quirks Mode 概述

定义

什么是 Quirks Mode? 简单来说，Quirks Mode 就是浏览器为了兼容很早之前针对旧版本浏览器设计、并未严格遵循 W3C 标准的网页而产生的一种页面渲染模式。

历史

由渲染引擎产生的两种文档模式

谈到 Quirks modes 首先就要从浏览器渲染引擎说起。我们知道所有的浏览器都有自己的页面渲染引擎，渲染引擎主要包含两部分，一部分负责 HTML、CSS 代码的解析，另一部分负责脚本代码解析，这两部分合起来就可以绘制出完整的页面。

表 1 各浏览器渲染引擎构成

	渲染引擎	HTML,CSS 解析模块	JS 解析模块
IE	Trident	--	Chakra
Firefox	Gecko	--	SpiderMonkey
Chrome	Webkit	Webcore	V8
Safari	Webkit	Webcore	Javascriptcore
Opera	Presto	--	Carakan

从表 1 可以看出，现在市面上的主流浏览器中除了 Chrome 和 Safari 都采用了 Webkit 渲染引擎，其余三种浏览器采用了各自不同的渲染方式（不同的 HTML 解析，不同的 js 解析）。我们这里暂且先不讨论不同的渲染引擎绘制页面时的差异，单以

每一种渲染引擎而言，随着版本的发展其渲染页面的方式也有很大的不同。

IE 是最早提出 Quirks Mode 与 Standards Mode（与 Quirks 相对应的一种模式）的，后来 Firefox、Chrome、Safari、Opera 等浏览器也都支持了这两种渲染方式。但是只有在 IE 中用户才可以自由地在两种方式之间切换，其他浏览器都是自动匹配其中一种。下文将主要以 IE 为例来说明 Quirks Mode 对页面绘制的影响，表 2 展示了 IE 随着其渲染引擎的发展，它对 HTML 页面的渲染改变如下。

表 2 IE 渲染引擎发展历史

Trident 版本	MSHTML.dll 版本	IE 版本	更新
unversioned	4.0.x	4	首发
unversioned	5.0.x	5	增加对 CSS1 的支持及改变对 CSS 的渲染
unversioned	5.5.x	5.5	修正部分 CSS 的排版控制
unversioned	6.0.x	6	修正 box model 的错误及新增 quirks Mode 的切换功能
unversioned	7.0.x	7	修正部分 CSS 排版错误以及增加对 PNGalpha 通道(半透明)
4.0	8.0.x	8	第一个通过 Acid2 测试的版本
5.0	9.0.x	9	首次支持 HTML5、SVG、CSS3 及采用新的 JavaScript 引擎
6.0	10.0.x	10	支持 CSS3 多栏式排版、格子对齐、浮动式区块排版、渐变

从表 2 可以清晰的看出，随着 IE 的发展，其渲染引擎（早期为 MSHTML.dll，后来命名为 Trident）也在不断加入新的特性以及修正一些早先版本的错误。在 2001 年 IE6 正式发布之前，当时的市面主要就是 IE 和 Netscape 的 Navigator 两款浏览器，而 IE 拥有很大的用户群，所以大多数的页面都是针对 IE 而设计的，但是 IE 的渲染引擎却没有遵循 W3C 的规范，当时微软已经认识到 W3C 规范的重要性。所以当 IE

发展到 IE6 的时候，渲染引擎（MSHTML.dll）做出了一个重要的改变，将自己原先不符合 W3C 规范中的盒模型 box mode 绘制方式改为与 W3C 标准一致（后面会详细讨论），由于这个重大的改动，原先针对 IE 旧版本所设计的 HTML 页面都不能正确显示了。所以在 IE6 发布的时候附带了一个切换回 IE5 页面渲染方式的功能，这个功能中就首次提出了 Quirks Mode。

当用户需要显示旧版本的页面时切换到 Quirks Mode，这时浏览器的渲染引擎就切换到 IE5.5 所对应的版本(MSHTML.dll 5.5.x)，box mode 还是按照之前的方式绘制，这样页面就可以正确显示。当用户需要显示一些新的、满足 W3C 规范的页面时，渲染引擎切换到一个与 Quirks Mode 对应的 Standards Mode（标准模式），在此模式下渲染引擎就是当前的最新版本，这样也就满足了更多的 W3C 规范。这两种 Mode 之间的差别就是因为工作在不同版本的渲染引擎环境下。

最后，Quirks Mode 和 Standards Mode 合起来就称为浏览器的文档模式 Document Mode。

Quirks 和 Standards 之外的第三种模式

实际上，在上文提到的具有 Quirks 和 Standards 两种文档模式的浏览器中还存在第三种模式—Almost Standards Mode。这种模式和 Standards Mode 几乎一致，唯一的区别就在于某些情况下 Almost Standards Mode 会采用与 Quirks 相同的方式来绘制页面。比如当我们需要把图片分割后显示在一个表格单元中时，Almost Standards Mode 与 Quirks Mode 采用同样的绘制方式从而让图片显示不像在 Standards Mode 中那么的四分五裂。

但是这只是极少数的情况，在大部分情况下 Almost Standards 和 Standards 两种模式是一致的，所以我们一般不专门区分二者，后面我们会提到如何查看浏览器渲染引擎信息，在这个信息中同样对 Almost Standards Mode 和 Standards Mode 是不做区分的。

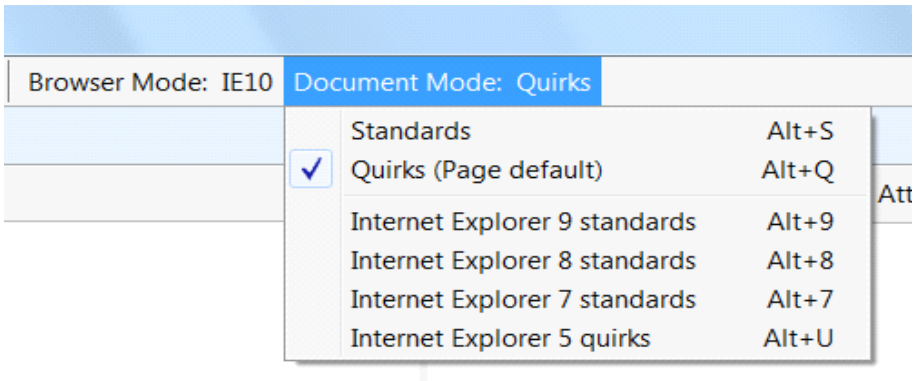
Quirks Mode 有两种

如果我们将 IE 升级到最新的 IE10 就会发现 IE10 除了拥有 IE7/8/9/10 Standards Mode 四种 Standards Mode，同样还有拥有了两种 Quirks Mode：IE5 Quirks 和 IE10Quirks。随着 IE10 发布而产生的这个新的怪异模式 IE10Quirks 被认为是一种更好的支持了 HTML5 规范的 Quirks Mode。我们可以发现针对 HTML5 规范而设计的页面（如含有<audio>、<video>、<canvas>等标签）在 IE5Quirks 下是不能正确显示的，但是在 IE10Quirks 下完全可以正确显示。也就是说，IE10Quirks 是为了在那些针对 HTML5 设计，但是又没有添加 doctype(可以决定浏览器工作在哪种模式下，后面会详细讨论)的页面而存在的。

如何查看 Document Mode

IE 中，用户可以在 developer tools 中切换模式，如图 1 所示，IE10 的六种文档模式都被显示在 Document Mode 菜单下，用户可以直接选择，下一章的 Demo 我们都采用 IE10(version: 10.0.9200.16576)作为测试浏览器。

图 1 IE Document Mode



除了从 developer tools 中查看，还有可以从 document 对象的属性 compatMode 中获知文档模式，这个属性只有两个值 BackCompat 和 CSS1Compat，前者对应的是 Quirks Mode 后者对应 Standards Mode。在 developer tools 中切换文档模式时，页面

会自动刷新，compatMode 的值也会随之改变。

浏览器如何判断文档类型？

上一节中我们知道 IE 用户可以在 developer tools 中改变文档模式。那么，如果用户没有自己选择，浏览器在准备解析、绘制一个页面的时候，它是如何决定文档模式的呢？实际上浏览器在渲染页面之前会检查两个内容，一个是页面是否有 doctype 信息，另外一个是页面是否有 x-ua-compatible 信息。

Doctype 检测

对于一个 HTML 页面，<!DOCTYP >声明位于其中最前面的位置，处于<html>标签之前，这个<!DOCTYP >可以告知浏览器使用哪种 HTML 规范，针对每种规范浏览器同样也会选择对应的文档模式。平时最常见的三种 doctype 信息对应的文档模式如下。

- 当 doctype 信息如下时，表明该页面是遵守了 HTML5 规范的，浏览器会选择 Standards Mode，这种 doctype 是最推荐的一种，我们平时设计页面都应该加上这一个 doctype。

  <!DOCTYPE html>
- 当 doctype 如下时，浏览器同样会选择 Standards Mode，虽然和第一种 doctype 有一些区别，但是几乎可以认为是一样的。

  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  “http://www.w3.org/TR/html4/strict.dtd">
- 当 doctype 如下时，浏览器会选择 Almost Standards Mode，需要注意的是如果今后需要把这个页面改为 HTML5 规范，那么上文讨论的<table>中的分割图片问题可能会错乱。

  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
- 当 doctype 缺失的时候，浏览器会选择 Quirks Mode，这是非常不推荐的方式，我们应该尽量避免 Quirks Mode，这对一个 web 应用是非常不利的地方。

x-ua-compatible 信息

除了上一节提到的 doctype 检测，HTML 页面的开发者可以在页面的<head>标签中加入 x-ua-compatible 信息来影响文档类型的判定，具体如下表所示。

表 3 x-ua-compatible 影响文档类型

x-ua-compatible	doctype	Document Mode
<meta http-equiv="X-UA-Compatible" content="IE=5" >	无影响	IE5 quirks
<meta http-equiv="X-UA-Compatible" content="IE=7/8/9/10" >	无影响	IE7/8/9/10 Standards
<meta http-equiv="X-UA-Compatible" content="IE=Edge" >	无影响	IE 最新版本的 Standards
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7/8/9" >	<!DOCTYPE html>	IE7/8/9 Standards
	不存在	IE5 quirks
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE10" >	<!DOCTYPE html>	IE10 Standards
	不存在	IE10 quirks

从表 3 可以看出，一般情况下 x-ua-compatible 是优先于 doctype 的设定的，但是当 x-ua-compatible 设定了如“EmulateIExx”的情况时，就会同样考虑到 doctype 的影响。

另外，在<head>中加入 x-ua-compatible 信息与在请求消息的 header 中加入是等同的，如下代码效果是等同的。

response.setHeader("X-UA-Compatible","IE=Edge");

<meta http-equiv="X-UA-Compatible" content="IE=Edge" >

到现在为止我们分析了 Quirks Mode 产生的历史、对浏览器的影响以及浏览器如何选择文档模式。下一章我们主要讨论两种不同的文档模式下渲染页面的差别。

标准模式下的页面与怪异模式下的页面区别

这一章我们主要选择一些典型的例子来说明 Quirks Mode 和 Standards Mode 对页面渲染的影响。

盒模型

前面提到，盒模型（box mode）是浏览器 Quirks Mode 和 Standards Mode 的主要区别。

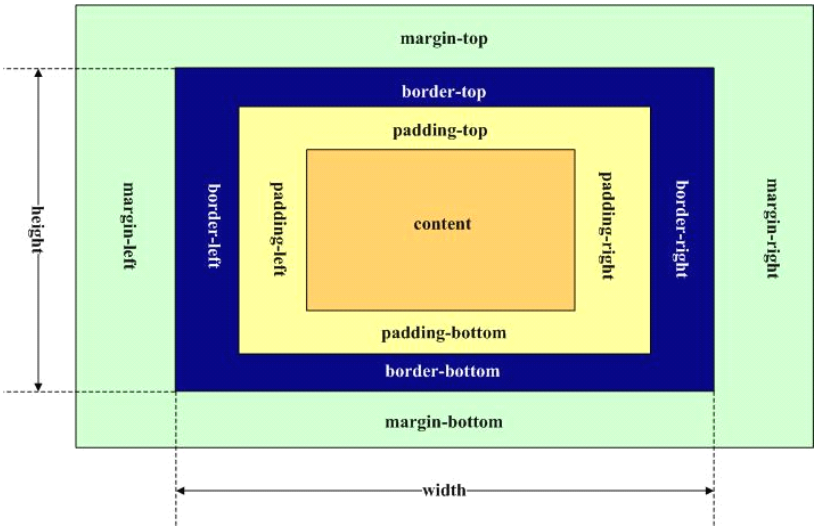
描述

对于“盒模型”一词并没有明确的文档定义，它是开发人员描述 CSS 中块级元素的一种约定俗称。

具体而言，针对一个块级元素，如<p>、 、等，CSS 的规范定义了一个宽度和高度，以及 3 个级别的环绕它的框 padding，border 和 margin 。这些属性我们可以把它转移到我们日常生活中的盒子上来理解，所以将这种模型称为盒模式。对于盒模型，针对高度和宽度的定义，不同浏览器的解释不同。

出于历史原因，早期的 IE 浏览器（IE 6 以前）将盒子的 padding 和 border 算到了盒子的尺寸中，这一模型被称为 IE 盒模型。该模型如图 2.1 所示，

图 2 IE 盒模型



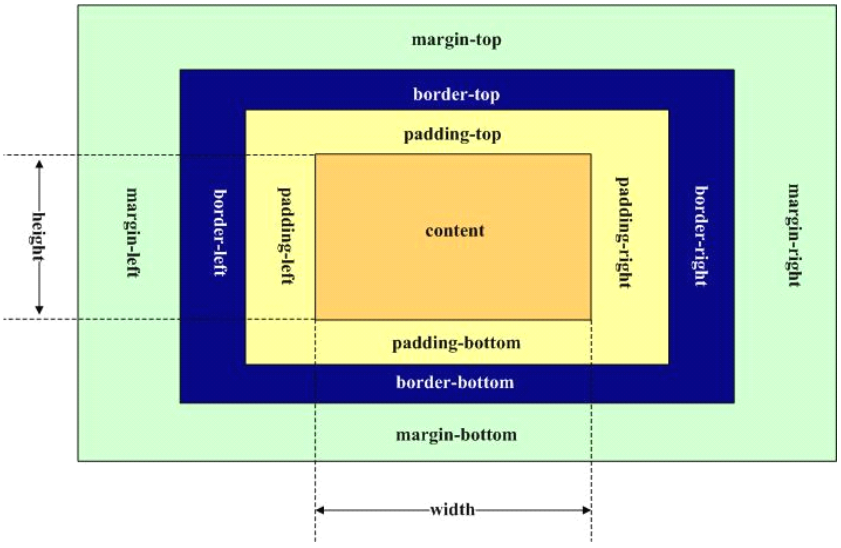
在 IE 盒模型中，

box width = content width + padding left + padding right + border left + border right，

box height = content height + padding top + padding bottom + border top + border bottom，

而在 W3C 标准的盒模型中，box 的大小就是 content 的大小，如图 3 所示，

图 3 W3C 标准盒模型



box width = content width，

box height = content height，

这一区别将导致页面绘制时所有的块级元素都出现很大的差别，所以两种不同的文档模式下的页面也区别很大。

示例展示

如下代码段所示，我们定义一个简单的 DIV 元素，设定其宽度和高度分别为 500px，定义 border 为 50px，红色。

清单 1

```
div.a{
    width:500px;
    height:500px;
    border:50px;
    border-style:solid;
    border-color:red;
}
```

分别在 IE 5 Quirks Mode 和 IE 8 Standards Mode 下运行，结果如图 4 和 5 所示。明显可以看到，在 Standards Mode 下的 div 要大于 Quirks Mode，其实际渲染大小为 600px*600px。

图 4 IE 5 Quirks Mode

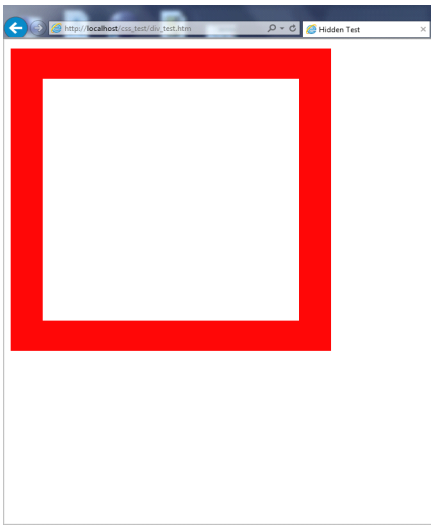
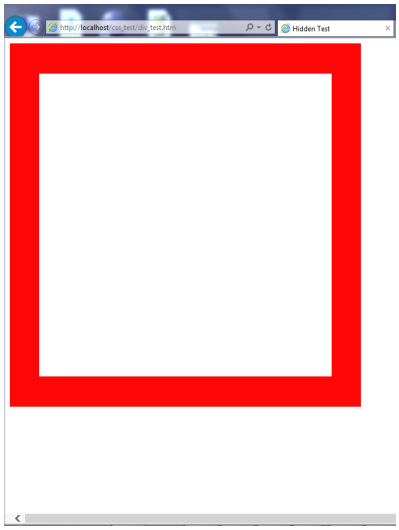


图 5 IE 8 Standards Mode



图片元素的垂直对齐方式

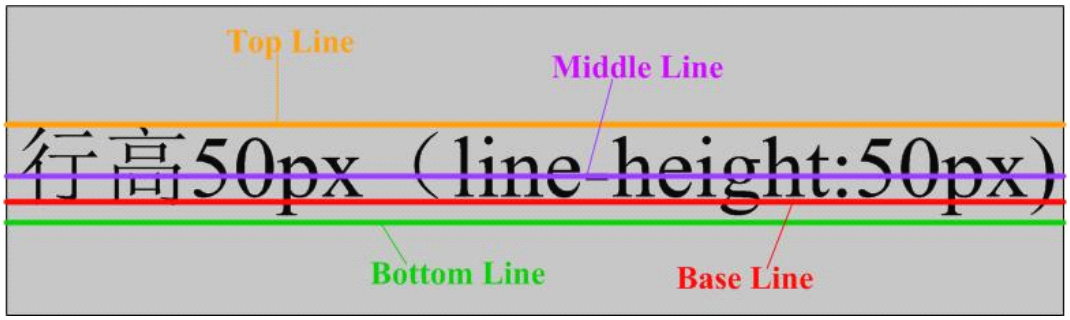
CSS 中 vertical-align 属性用于设置或检索对象内容垂直对齐方式，该属性定义行内元素的 base line 相对于该元素所在行的 base line 的垂直对齐。在表单元格中，这个属性会设置单元格框中的单元格内容的对齐方式。其取值可以为 baseline，sub，

supper，top，text-top，bottom，text-bottom，middle 等。什么是 baseline 和 bottom，他们有何区别？下面我们通过一副图来进行解释。

描述

CSS 为了确定文字行的位置，定义如下概念描述，base line，bottom line，top line，middle line。其中，base line 指的是一行字横排时下沿的基础线，baseline 并不是汉字的下端沿，而是英文字母 e 的下端沿，bottom line，指的是汉字，或者英文字母 p，g 的下端沿。如下图 6 所示。

图 6 base line 概念



对于 inline 元素和 table-cell 元素，在 IE Standards Mode 下 vertical-align 属性默认取值为 baseline。而当 inline 元素的内容只有图片时，如 table 的单元格 table-cell。在 IE Quirks Mode 下，table 单元格中的图片的 vertical align 属性默认为 bottom，因此，在图片底部会有几像素的空间。

示例展示

如下代码段所示，我们定义一行两列的 table，table 单元格设定为宽度和高度均为 200px 的 img 图片，为了突出显示区别，分别定义单元格与图片的边框颜色为蓝色和橘色。

清单 2

```
td.a
{
    border-style:solid;
    border-color:blue;
}

img.c
{
    width:200px;
    height:200px;
    border-style:solid;
    border-color:orange;
}
```

分别在 IE 10 Quirks Mode 和 IE 8 Standards Mode 下运行，结果如图 7 和 8 所示。在 Quirks Mode 下，table 单元格中的图片与单元格底部对齐，而在 Standards Mode 下，图片与单元格之间多了 3 个像素的空间。

图 7 IE 10 Quirks Mode

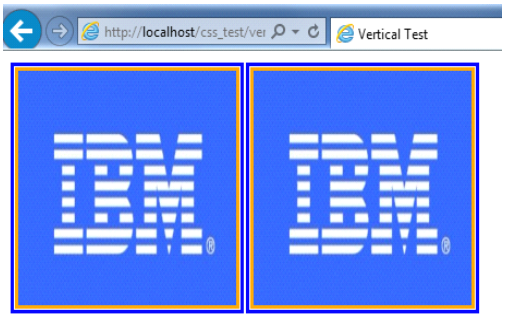
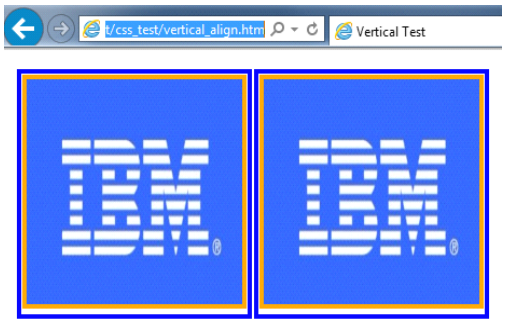


图 8 IE 8 Standards Mode



<table>元素中的字体

CSS 中，描述 font 的属性有 font-family，font-size，font-style，font-weight，分别表示了 font 的族系，大小，风格以及粗细。

描述

在 CSS 标准中，上述属性都是可以继承的。而在 IE Quirks Mode 下，对于 table 元素，字体的某些属性将不会从 body 或其他封闭元素继承到 table 中，特别是 font-size 属性。

示例展示

如下代码段所示，我们定义 body 的 font 属性为斜体、红色、加粗、fantasy 字体，对于 table 元素，未定义其 font 属性。

清单 3

```
body
{
    font-style:italic;
    color:red;
    font-size:200%;
   font-weight:bold;
```

```
    font-family: fantasy;
}
```

分别 IE 5 Quirks Mode 和 IE 8 Standards Mode 下运行，结果如图 9 和 10 所示。在 Quirks Mode 下，table 单元格中字体的 font-size，font-style，font-weight 属性不会继承 body，只有 family 属性会被继承。而在 Standards Mode 下，所有属性都被继承。

图 9 IE 5 Quirks Mode

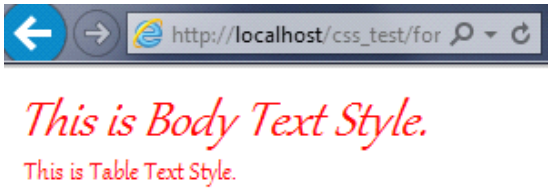


图 10 IE 8 Standards Mode

内联元素的尺寸

CSS 中常见的元素有两类，分别是 block（块级）元素及 inline（内联）元素。这两类元素的主要区别在于 block 元素通常作为独立的一块继续显示，前后都有换行，而 inline 元素则不会产生换行。根据 CSS 标准，对于 inline 元素，又可以分为 replaced 和 non-replaced 两类。

描述

什么是 non-replaced inline 元素？首先，我们解释下 non-replaced 元素，对于 HTML 中定义的元素，默认具有 CSS 格式化外表范围的元素，比如 img 元素，有自己的宽和高，我们称其为 replaced 元素，其他例 input，textarea，select，object，等都是 replaced 元素。除了这些元素之外的元素就是 non-replaced 元素。因此，具有 non-replaced 属性的 inline 元素即为 non-replaced inline 元素，如 span 元素。

在 IE Standards Mode 下，non-replaced inline 元素无法自定义大小，而在 IE Quirks Mode 下，定义这些元素的 width 和 height 属性，能够影响该元素显示的大小尺寸。

示例展示

如下代码段所示，为了突出显示，我们定义一个背景色为橘色的 div 中嵌套一个 span 元素，该 span 元素的高和宽均为 200px，背景色为蓝色。

清单 4

```
div.a
{
    width:300px;
    height:300px;
    background-color:orange;
}

span.b
{
    height:200px;
    width:200px;
    background-color:blue;
}
```

分别在 IE 5 Quirks Mode 和 IE 8 Standards Mode 下运行，结果如图 11 和 12 所示。在 Quirks Mode 下，span 元素能够正常显示，左图中 200*200 的蓝色的区块。而在

Standards Mode 下，span 尺寸为零。

图 11 IE 5 Quirks Mode

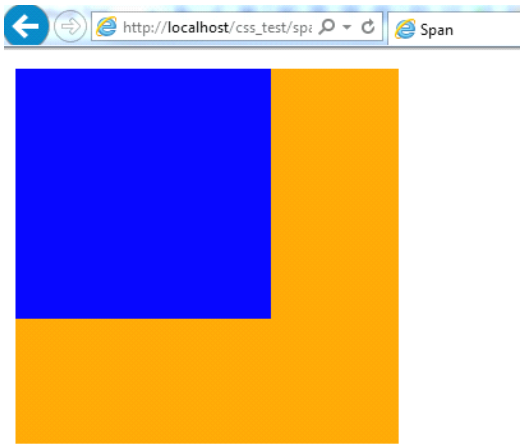
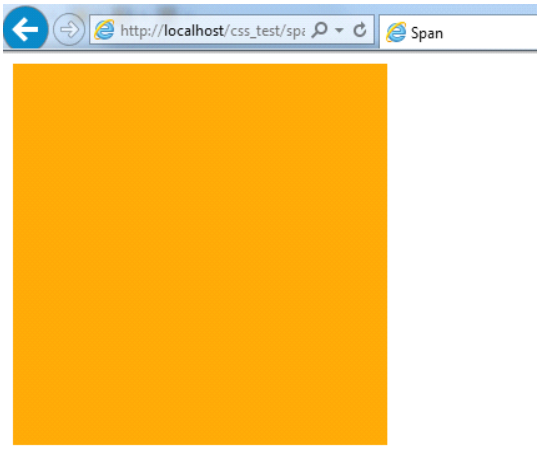


图 12 IE 8 Standards Mode



元素的百分比高度

CSS 中对于元素的百分比高度规定如下，百分比为元素包含块的高度，不可为负值。如果包含块的高度没有显式给出，该值等同于“auto”（即取决于内容的高度）。所以百分比的高度必须在父元素有声明高度时使用。

描述

当一个元素使用百分比高度时，在 IE Standards Mode 下，高度取决于内容的变化，而在 Quirks Mode 下，百分比高度则被正确应用。

示例展示

如下代码所示，为了突出显示，我们定义一个背景为粉色的 table，在 table 的单元格中嵌入一个背景为橘色的 div b，将该 div 的高度设置为 90%。定义 b 的子节点 c 为高度和宽度均为 200px 的 div 元素，背景为蓝色。

清单 5

```
table.a
{
    height:500px;
    background-color:pink;
}
div.b
{
    background-color:orange;
    width:300px;
    height:90%;
    display:block;
}
div.c
{
    width:200px;
    height:200px;
    background-color:blue;
}
```

分别在 IE 5 Quirks Mode 和 IE 8 Standards Mode 下运行，结果如图 13 和 14 所示。在 Quirks Mode 下，div b 的高度为 talle 单元格的 90%，而在 Standards Mode 下，div b 的高度由其子节点 c 决定，为 200px。

图 13 IE 5 Quirks Mode

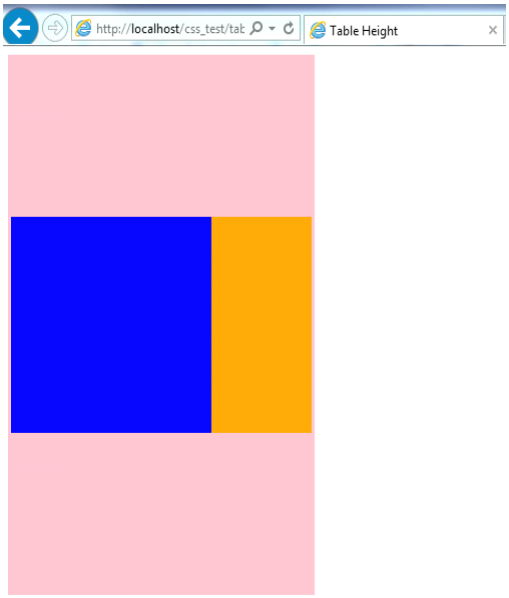
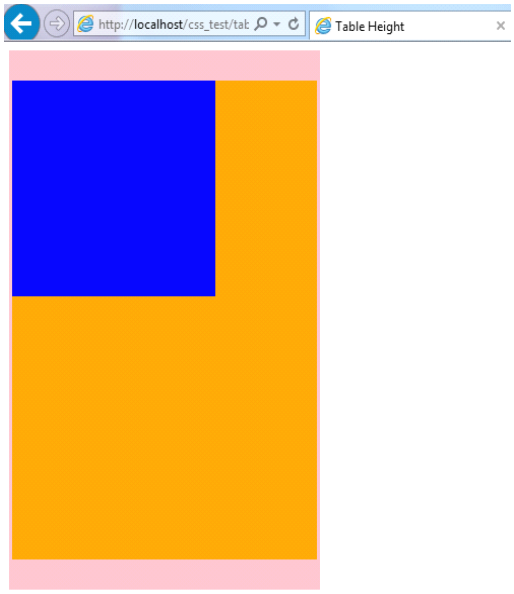


图 14 IE 8 Standards Mode



元素溢出的处理

CSS 中 overflow 属性定义了一个元素的内容不适合指定的尺寸时，溢出元素内

容的处理方式。默认值为 visible，即显示溢出。

描述

在 IE Standard Mode 下，overflow 取默认值 visible，即溢出可见，这种情况下，溢出内容不会被裁剪，呈现在元素框外。而在 Quirks Mode 下，该溢出被当做扩展 box 来对待，即元素的大小由其内容决定，溢出不会被裁剪，元素框自动调整，包含溢出内容。

示例展示

如下代码段所示，我们定义一个背景为蓝色的 div a，在 a 中嵌入一个背景为橘色的 div b，设置 b 的高度 400px 大于 a 的高度 300px，使 a 发生溢出。

清单 6

```
div.a
{
    width:300px;
    height:300px;
    background-color:blue;
}

div.b
{
    width:200px;
    height:400px;
    background-color:orange;
}
```

分别在 IE 5 Quirks Mode 和 IE 8 Standards Mode 下运行，结果如图 15 和 16 所示。在 Quirks Mode 下，div a 的高度为又 300px 变为 400px，以适应 b 的大小，而在 Standards Mode 下，div a 的大小保持不变，溢出部分保留。

图 15 IE 5 Quirks Mode

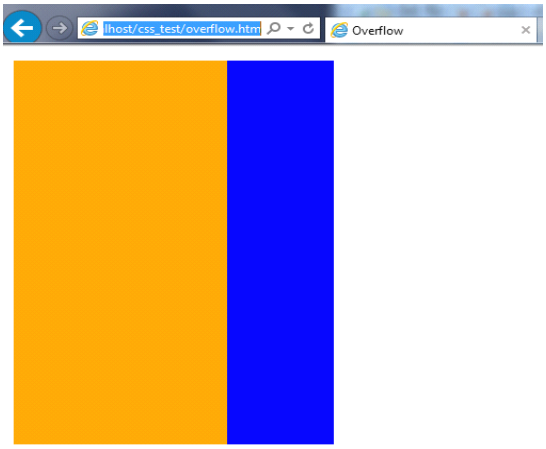
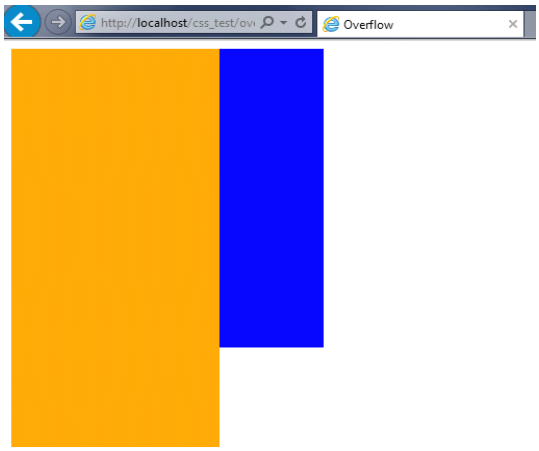


图 16 IE 8 Standards Mode



结束语

通过前文的描述我们已经知道 Quirks 和 Standards 这两种文档模式渲染页面的时候会有很大的差别，这些差别主要是由于渲染引擎在历史的发展过程中与 W3C 标准的差异性而导致的。

知道这些差异的存在和由来将对 web 工程师的工作有很大的帮助。在新产品的开发中 web 工程师应该让页面满足 HTML5 规范，避免浏览器工作在 Quirks 模式下。另外一点是如果在维护只能工作在 Quirks 模式下的 web 产品时，可以从 Quirks 模式的根源来出发，考虑如何改进使得产品回到 Standards 模式，这样就可以添加进更

多更好的功能。

参考资源（resources)

- 参考 http://en.wikipedia.org 中的以下条目介绍：Quirks_mode, Trident_(layout_engine)，WebKit, V8_(JavaScript_engine), SpiderMonkey_(JavaScript_engine)。
- 查看文章“Activating Browser Modes with Doctype”，系统的了解了Quirks Mode的产生原因，分类以及检测。
- 查看文章 “IE10 quirks mode介绍”，了解了IE10 中两种quirks mode之间的区别。
- 查看文章“Specifying legacy document modes”，了解了DOCTYPE如何影响HTML页面的文档模型。
- 查看文章“Defining document compatibility”，了解了浏览器兼容视图，明白了浏览器模型（Browser Mode）与文档模型（Document Mode）的区别。
- 查看文章“Just The Facts: Recap of Compatibility View”，了解了浏览器模型与文档模型如何互相影响。
- 查看文章“About conditional comments”，了解了条件注释在不同的浏览器模型和文档模型下的表现。