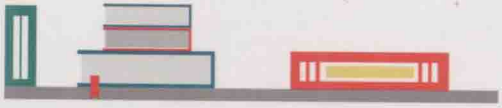
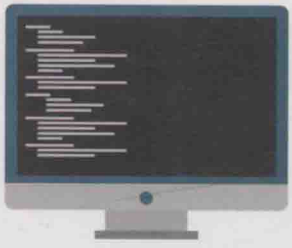


```
<style>
    body,div{
        margin:0;
    }
    html{
        margin-top:50px;
    }
    div{
        width:100px;
        height:100px;
        margin-top:50px;
        background-color:red;
    }
</style>
<div></div>
```



CSS 核心技术详解

肖志华 著

深入浅出地讲解CSS核心技术，
幽默风趣，实用性强。

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

CSS 核心技术详解

肖志华 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书共分 13 章，第 1 章主要解答 CSS 中常见的问题，以及常用的技巧。第 2～6 章讲解了 CSS 的核心技术，其中第 2 章是最为核心的内容，相对于其他章节理解起来会比较难一点。第 3～6 章主要介绍案例，配合第 2 章阅读会轻松很多。本书每个章节都是独立的，因此如果某些章节看不懂，可以暂且跳过，先阅读其他章节。第 7～13 章讲解关于 CSS 3 的内容。

本书内容精练、实例丰富、通俗易懂，可作为广大 CSS 设计人员和前端开发人员的参考书，同时也非常适合大中专院校师生学习阅读。

图书在版编目（CIP）数据

CSS 核心技术详解 / 肖志华著. —北京：电子工业出版社，2017.6

ISBN 978-7-121-31330-1

Ⅰ. ①C… Ⅱ. ①肖… Ⅲ. ①网页制作工具 Ⅳ.①TP393.092.2

中国版本图书馆 CIP 数据核字（2017）第 076255 号

策划编辑：黄爱萍
责任编辑：徐津平
印　　刷：北京中新伟业印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：20.5　字数：409 千字
版　　次：2017 年 6 月第 1 版
印　　次：2017 年 6 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

前　言

看似简单的 CSS，却暗藏玄机，那是我们摸爬滚打好长时间后悟出的真理。

在很长的一段时间里，我并没有重视 CSS，觉得 CSS 很简单，无非就是一些属性；后来才发现自己小看了 CSS，对 CSS 的了解实在是太少，尤其是对其核心概念的理解太模糊，实际上它有很多神奇的地方并不为大家所知。对于一个新手来说，只知道一些理论但在实际开发中不会使用是不行的，于是笔者萌生了写作本书的最初想法。

市面上介绍 CSS 基础的书已经有很多了，已经没有必要再去重复，但是一些核心的内容还是很有必要写出来的，因为我发现很多前端朋友对 CSS 都不太重视。我认为做前端的技术人员不仅要掌握好 CSS 的核心内容，还要懂得怎样把这些内容灵活运用到实际开发中。如果对一门技术只停留在了解的层面而不会使用，那和不会有什么区别？所以本书将实用放在第一位，大量的例子都来自于我在实际开发中所遇到的问题，将这些实际的例子拿来讲解才更有说服力，同时也更易于读者的理解。

本书共分 13 章，第 1 章主要解答 CSS 中常见的问题，以及常用的技巧。第 2～6 章讲解了 CSS 的核心技术，其中第 2 章是最为核心的内容，相对于其他章节理解起来会比较难一点。第 3～6 章主要介绍案例，每个代码片段都有一些案例，配合第 2 章阅读会轻松很多。本书每个章节都是独立的，因此如果某些章节看不懂，可以暂且跳过，先阅读其他章节。第 7~13 章讲解关于 CSS 3 的内容。

本书举例时用到了很多关于 CSS 3 的属性，所以读者在测试时需要使用高级浏览器，这里推荐使用 Chrome 浏览器，书中的例子主要也是在 Chrome 浏览器中测试的。另外本书并不会过多地讲解兼容性问题，因为花太多时间讨论兼容性是不太值得的。书中有一些个人看法，由于才疏学浅，不免会有疏漏。如果发现错误，还请指出，不吝赐教，在此深表

谢意，可发邮件至 c776@foxmail.com 邮箱，一定一一回复并乐此不疲，因为这是我的工作，和你们交流也是我的快乐。

本书的出版要特别感谢电子工业出版社的黄爱萍和张童编辑，感谢他们在选题策划和书稿编辑方面做出的大量工作，同时对伯乐在线黄利民大哥的大力支持深表谢意。

肖志华

2017 年 3 月 28 日

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与作者交流**：在页面下方【读者评论】处留下您的疑问或观点，与作者和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/31330

目　录

第 1 章　遇见未知的 CSS　/　1

1.1　在 CSS 中会遇到的问题　/　1

1.1.1　CSS 层叠规则　/　3

1.1.2　CSS 的命名　/　5

1.2　CSS 的一些技巧　/　6

1.2.1　使用 pointer-events 控制鼠标事件　/　6

1.2.2　玩转 CSS 选择器　/　8

1.2.3　利用 padding 实现元素等比例缩放　/　11

1.2.4　calc 函数　/　14

1.3　隐藏元素　/　18

第 2 章　CSS 核心概念　/　23

2.1　CSS 解析规则　/　23

2.2　替换元素与非替换元素　/　28

2.3　属性值的计算规则　/　28

2.4　可视化格式模型　/　30

2.5　包含块　/　31

2.6　控制框　/　38

2.7　格式化上下文 BFC、IFC　/　40

2.7.1　从 overflow 清除浮动看 BFC（块格式化上下文）　/　40

2.7.2 块级格式化上下文 BFC / 45
2.7.3 折叠外边距 / 54
2.7.4 行内格式化上下文 IFC / 58
2.7.5 行高的计算 / 61

第 3 章 CSS 单位究竟来自何方 / 67

3.1 百分比究竟为谁 / 67
3.2 探索 auto 密码 / 82
3.3 设计响应式网页 rem / 93
3.4 vw、vh、vmin、vmax 基于视口单位 / 97
3.5 什么是 ch / 102
3.6 min、max 的巧妙运用 / 104
3.7 一个 none 引出的大学问 / 106

第 4 章 那些年我们一起定位过的元素 / 108

4.1 定位的特点 / 108
4.1.1 定位之 absolute 篇 / 109
4.1.2 定位之 relative 篇 / 113
4.1.3 当定位遇到定位 / 117
4.1.4 定位之 fixed 篇 / 121
4.1.5 偶遇定位 bug，才知定位的真理 / 122
4.1.6 定位之 static 篇 / 129
4.2 透彻研究定位隐藏的秘密 / 130
4.3 总结 / 140

第 5 章 元素的七十二变——元素转换 / 142

5.1 display 介绍 / 142
5.2 大块头——block / 142
5.3 我们一起站一排——inline / 143

5.4 inline 和 block 的结合体——inline-block / 149
5.5 行内和块的烦恼 / 152
5.6 dispaly 的一些其他属性 / 155
5.7 总结 / 159

第 6 章 浮动趣事 / 160

6.1 浮动简介 / 160
6.2 浮动的特点 / 161
6.3 浮动的秘密 / 167
6.4 实现任意形状的文字环绕 / 173
6.5 总结 / 188

第 7 章 再不学这些选择器就老了 / 189

7.1 那些被遗忘的选择器 / 189
7.1.1 相邻兄弟选择器 / 189
7.1.2 利用 hover 实现一个下拉菜单 / 192
7.1.3 利用 active 做一个集能量 / 194
7.1.4 用 first-letter 选中第一个字 / 195
7.1.5 用 first-line 选择首行文字 / 197
7.2 模拟父级选择器 / 199
7.3 强大的新选择器 / 200

第 8 章 CSS 图标制作 / 210

8.1 隐藏在边框中的秘密 / 210
8.2 边框的烦恼 / 212
8.3 边框的孪生兄弟——outline / 215
8.4 纯 CSS 图标制作 / 220

第 9 章　你今天换背景了吗　/　232

9.1　对背景属性的深入探索　/　232

9.2　新增的背景功能　/　237

9.2.1　改变背景原点——background-origin　/　237

9.2.2　背景裁剪——background-clip　/　239

9.2.3　设置背景图片大小——background-size　/　243

9.3　总结　/　245

第 10 章　让文字更美一些　/　246

10.1　制作非主流文字　/　247

10.2　新增的文字对齐属性　/　250

10.2.1　文字两端对齐　/　250

10.2.2　末尾文本对齐　/　252

10.2.3　文本书写模式　/　257

10.3　关于文字的一些其他属性　/　259

10.3.1　将超出宽度的文字隐藏　/　259

10.3.2　字母转换大小写　/　262

10.4　总结　/　263

第 11 章　内容生成技术——用 CSS 来计数　/　264

11.1　伪元素　/　264

11.2　CSS 计数器　/　265

11.3　content 的其他用途　/　272

11.4　总结　/　273

第 12 章　解决让人头疼的布局　/　274

12.1　制作可自适应的布局　/　274

12.1.1　左侧固定、右侧自适应的布局　/　274

12.1.2　右侧固定、左侧自适应的布局　/　276

12.1.3 多列文字垂直居中 / 278
12.2 利用伸缩盒模型进行布局 / 283
12.2.1 伸缩盒模型基础 / 285
12.2.2 伸缩盒模型进阶 / 296
12.2.3 伸缩盒模型实战 / 299

第 13 章 飞越 CSS / 303

13.1 CSS 最佳实践 / 303
13.2 纯 CSS 的世界 / 307
13.2.1 利用 checked 选择器实现 tab 切换 / 308
13.2.2 利用:target 选择器实现遮罩层效果 / 310
13.2.3 scaleY 配合 animation 制作 loading / 311
13.2.4 利用 hover 实现手风琴效果 / 313
13.2.5 利用 checked 选择器制作星星评分 / 314
13.2.6 使用 flex 伸缩盒模型实现瀑布流布局 / 316
13.3 结束语 / 318

第 1 章　遇见未知的 CSS

1.1　在 CSS 中会遇到的问题

1. 在 CSS 中为什么要有层叠

在 CSS 中可能会有多个样式表同时影响同一个元素的某个属性，设计这个功能的主要原因有两个，解决模块化和作者、用户、用户代理样式冲突。

（1）模块化

一个页面中的样式可以拆分成多个样式表，代码如下。

```
@import url(style/base.css);
@import url(style/layer.css);
```

但这种方式也会随之产生一个问题，即如果对某个元素的同一个属性设置样式，到底应用谁的呢？

（2）作者/用户/用户代理

当作者（写代码的人）和用户（浏览页面的人），以及用户代理（一般指浏览器）都能更改样式表时，也会产生同样的问题：究竟用谁设置的样式，因此 CSS 层叠机制就显得格外重要。

2. 为什么“@import”指令需要写在样式表的开头

代码如下。

```
@import url(style/layer.css);
body{
  background-color:red;
}
```

“@import”指令之所以需要写在样式表的开头，是因为可以使后面的样式能更好地层叠导入进来的样式。

3. 当 CSS 值为 0 时为什么可以省略单位

因为当 CSS 值为 0 时，任何单位的结果都是一样的，就像数学中的 0 乘以任何数都得 0。

4. margin 垂直外边距折叠的意义是什么

margin 垂直外边距折叠的特性主要来自传统排版，举个例子，代码如下。

```
<style>
  body,ul,li{
    margin:0;
    padding:0;
  }
  ul{
    list-style:none;
  }
  ul>li{
    margin:20px 0;
  }
</style>
<ul>
  <li>1111111111</li>
  <li>2222222222</li>
  <li>3333333333</li>
</ul>
```

代码的效果如图 1.1 所示。

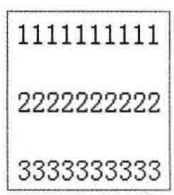


图 1.1　margin 外边距折叠效果

从图 1.1 中可以看到 3 行数字的垂直外边距都是一样的。如果没有这个特性，第一行数字与下面两行数字的外边距就不一样了，因为我们给每个 li 都设置了一个上下外边距，假如没有外边距折叠，那么第 1 个 li 的下边距加上第 2 个 li 的上边距，就是两倍的间距了，但是第一个 li 上面没有其他元素，所以它只有一个上边距，最终导致的结果就是，第 1 个 li 和后面的几个 li 的外边距不一样，这显然不是我们所希望的。而 margin 外边距折叠功能正是要在这种情况下，让格式可以好看一点。

1.1.1 CSS 层叠规则

在介绍 CSS 层叠规则之前首先举个例子，代码如下。

```
<style>
  .box{
    color:red;
    font-size:18px;
  }
</style>
<div class="box">
  <a href="#">层叠</a>
</div>
```

代码的效果如图 1.2 所示。

层叠

图 1.2 继承的颜色被层叠

按理说颜色是可以继承的，那么为什么 a 标签的颜色没有变成红色呢？审查一下元素，如图 1.3 所示。

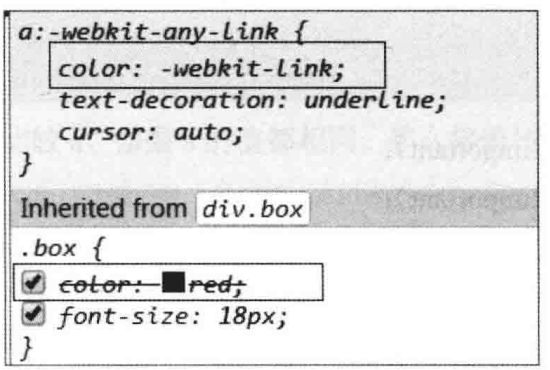


图 1.3 Chrome 浏览器中的效果

从图 1.3 中可以看到继承的颜色被划掉了，出现这个问题的原因是浏览器对 a 标签设置了默认样式，将继承的样式层叠了，因为继承的样式权重最小。下面介绍 CSS 关于层叠规则是怎么计算的。

在 CSS 中一个样式可能会来自不同的地方，分别是作者、用户及用户代理。如果在这几个样式中，它们对同一个元素的同一个属性做了不同的操作，那么用户代理应该如何处理这段 CSS 呢？举个例子，代码如下。

```
/* 作者 */
.box{
  color:red;
}
/* 用户代理 */
.box{
  color:green;
}
/* 用户 */
.box{
  color:pink;
}
```

可以看到用户代理、用户和作者的代码起冲突了，而 CSS 的层叠规则就是为了解决这些冲突，以下是一些 CSS 层叠规则。

在层叠中每个样式规则都有一个权重值，当其中几条规则同时生效时，权重值最大的规则优先。一般来说作者指定的样式权重值高于用户样式权重值，用户样式权重值高于客户端（用户代理）权重值。

在层叠顺序中，以下权重值从小到大。

（1）用户代理样式；

（2）用户一般样式；

（3）作者一般样式；

（4）作者重要样式（!important）；

（5）用户重要样式（!important）；

（6）如果是两个样式来自相同的代码，如都来自作者（代码），并且它们的样式声明同样重要，则根据特异度来计算，特异度高的会覆盖特异度低的；

（7）如果特异度也相同，则越往后的样式优先级越高。

1．!important 声明规则

!important 声明的样式比一般声明优先级高，并且用户设置的!important 比作者设置的!important 优先级高。这样做的原因是为了方便用户实现一些特殊的要求，例如页面字体大小的调整等。

下面举一个!important 规则的例子，代码如下。

```
<style>
  .box{
    color:red !important;
```

```
  }
  .box{
    color:green;
  }
</style>
<div class="box">!important</div>
```

在正常情况下，后一个“color:green”会层叠前一个“color:red”，但这里我们给“color:red”设置了!important 规则，所以前一个优先级高。

2. 选择器特异度的计算

（1）如果一个声明出现在元素的 style 属性中，则将 a 计为 1；

（2）b 等于选择器中所有 id 选择器加起来的数量和；

（3）c 等于选择器中所有 class 选择器和属性选择器，以及伪类选择器加起来的数量和；

（4）d 等于选择器中所有标签选择器和伪元素选择器加起来的数量和。

将 a、b、c、d 这 4 个数字连接起来（a-b-c-d）就组成了选择器的特异度。一段特异度的计算，如下所示。

```
.box{}          /* a=0 b=0 c=1 d=0 特异度 = 0,0,1,0 */
.box div{}      /* a=0 b=0 c=1 d=1 特异度 = 0,0,1,1 */
#nav li{}       /* a=0 b=1 c=0 d=1 特异度 = 0,1,0,1 */
p:first-line{}  /* a=0 b=0 c=0 d=2 特异度 = 0,0,0,2 */
style=""        /* a=1 b=0 c=0 d=0 特异度 = 1,0,0,0 */
```

它们的比较顺序是先比较 a，如果 a 的值都相同，那么接着比较 b、c、d 的值，谁的数大则优先级就越高。

在使用 CSS 选择器时，你需要注意以下两点。

- 继承的优先级最低，没有特异度；
- 结合符（如+、>等）及通用选择符（*）特异度为 0。

因此，可以知道之前 a 标签 color 属性为什么没有被应用了，因为继承的优先级最低。

1.1.2 CSS 的命名

在代码复用时，也许你写过类似以下这样的代码，代码如下。

```
size-10{
 font-size:10px;
}
```

虽然这段代码看起来没什么问题，但如果考虑到可维护性，那就有很大问题了。假如有一天你不想用 10px，想改成 12px，那么直接再加一个 class 就行了，修改后的代码如下。

```
size-10{
font-size:10px;
}
size-12{
 font-size:12px;
}
```

但那些页面中原本用的 size-10 的 class 都得修改成 size-12，所以不建议这么修改代码。笔者建议用语义的方式命名，代码如下。

```
.size-small{
  font-size:12px;
}
```

这样写代码的好处是当需要调整字体大小时，只需修改一处，而不必修改添加到元素上的 class。也就是说不要按照显示的效果命名，而是根据这个 class 的用意命名。

1.2 CSS 的一些技巧

1.2.1 使用 pointer-events 控制鼠标事件

可以用 CSS 中的 pointer-events 来控制元素什么时候响应鼠标事件，比较常用的一个场景是获取验证码，如图 1.4 所示。

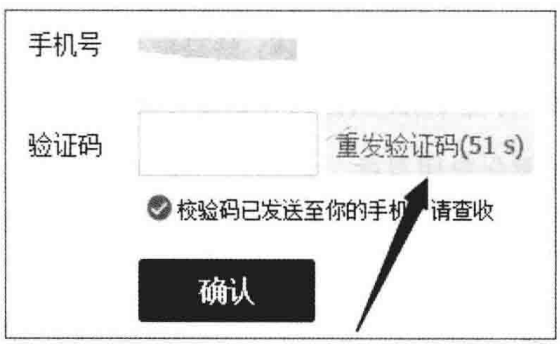


图 1.4 获取验证码

当用户单击“获取验证码”按钮后，需要等待 60 秒才能再次单击“重发验证码”按钮，

在这种情况下，就可以尝试用 pointer-events 实现禁用鼠标单击事件。在 pointer-events 属性中有一个 none 值，将 pointer-events 的值设置成 none 就不会响应鼠标事件了。举一个获取验证码的例子，代码如下。

```
<style>
    a{
        color:red;
    }
    .disable{
        pointer-events:none;
        color:#666;
    }
</style>
<a href="javascript:;" id="btn">发送验证码</a>
<script>
    var oBtn = document.getElementById('btn');
    oBtn.onclick = function(){
        oBtn.classList.add('disable');
        setTimeout(function(){
            oBtn.classList.remove('disable');
        },3000)
    };
</script>
```

如果看不懂这段代码也没关系，将这段代码复制下来即可。这段代码的意义就是定义了一个鼠标事件禁用的 class，单击“发送验证码”按钮后加上刚刚定义的.disable，3 秒以后再将这个 class 去掉。默认情况下的按钮，如图 1.5 所示。

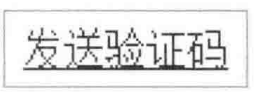


图 1.5 默认情况下的按钮

单击此按钮后，在 3 秒内不会再次响应单击事件。

pointer-events 除了可以实现此功能之外，还有很多用处，比如实现 a 标签禁止页面跳转，提高网页性能，用户在滚动页面时可能会不小心碰到一些元素上绑定的事件，这些事件就会被触发，从而浪费资源，但如果在页面滚动时给 body 加上 pointer-events:none;属性，那么就避免了这个问题。