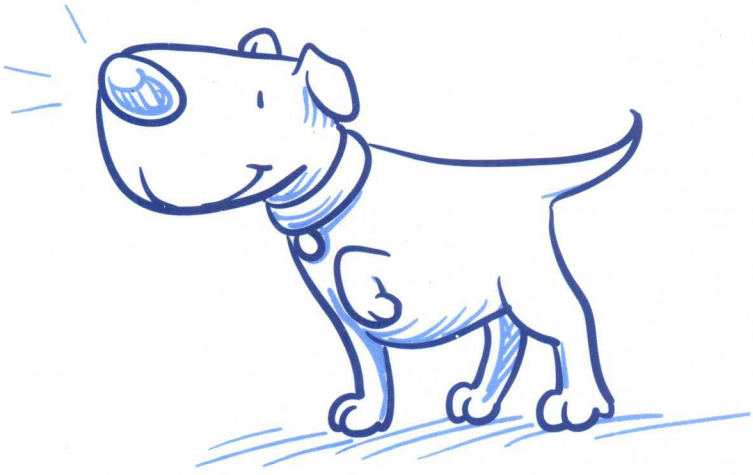
HZ BOOKS
华章IT

资深前端工程师、知乎著名前端专栏作者兼前端类话题优秀答主经验总结

从思想提升和内容修炼两个维度，围绕前端工程师必备的前端技术和编程基础，
总结出高效编程和应用优化的34个最佳实践

Web开发技术丛书



高效前端

Web高效编程与优化实践

EFFECTIVE FRONT END

李银城 著

机械工业出版社
China Machine Press

Web开发技术丛书

高效前端

Web高效编程与优化实践

EFFECTIVE FRONT END

李银城 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

高效前端：Web 高效编程与优化实践 / 李银城著 . —北京：机械工业出版社，2018.1
（2018.5 重印）
（Web 开发技术丛书）

ISBN 978-7-111-59021-7

I. 高… II. 李… III. 网页制作工具 - 程序设计 IV. TP393.092

中国版本图书馆 CIP 数据核字（2018）第 016715 号

高效前端：Web 高效编程与优化实践

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：李 艺 责任校对：李秋荣
印 刷：北京市兆成印刷有限责任公司 版 次：2018 年 5 月第 1 版第 2 次印刷
开 本：186mm × 240mm 1/16 印 张：25.75
书 号：ISBN 978-7-111-59021-7 定 价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Preface 前　言

为何写作本书?

本书通过介绍前端的优化实践以达到高效编程之功效，这里并不是教你怎么用 CSS 的某个属性，如“display：grid”，或者怎么用 JS 的 ES6，而是重点教你一些前端的思想，如怎么提高用户体验，怎么写出简洁优美的代码等。注重思想而不注重形式，注重功底而不注重框架是本书的特色。本书有一大部分篇幅在介绍怎么提升编程的功底，怎么修炼内功，从而达到高效编程的目的。

全书以问题为导向，例如有些页面为什么打开会比较卡顿，从怎么解决这种问题，有哪些方法，这些方法的优缺点是什么，一步步由浅入深地分析和解决问题。学会解决问题，比学会知识更为重要。

本书主要内容

本书分为七章，第 1 ～ 4 章和第 7 章的实践性比较强，第 5 章和第 6 章注重基础。

第 1 章介绍如何使用浏览器提供的便利进行开发，能使用 HTML/CSS 解决的问题就不要使用 JS，因为用 HTML/CSS 解决一般会更加简单，用户体验也会更好。

第 2 章介绍怎么样写出简洁高效的 JS 代码，怎么组织代码逻辑，让代码更加优美，具有更好的扩展性。

第 3 章介绍页面整体的优化，包括怎么加快页面的打开速度，怎么避免页面的卡顿，怎么从一些细节之处提升用户的体验，怎么更好地使用调试工具。

第 4 章结合实际经验，介绍 HTML5 的一些实用技术，如使用 history 改善 AJAX 体验、图标字体和 SVG、裁剪压缩图片、如何做一个 PWA 应用等。

第 5 章回归技术基础，以 WebSocket、wasm、Web Workers 等 HTML5 的新技术为出发点回归到计算机基础，如网络协议、程序编译、多线程等。特别介绍了它们和前端的联系，只有掌握这些基础，才能更好地解决问题，做一个优秀的前端开发人员。

第 6 章讨论了诸如跨域、上传文件、CSS 布局等前端技术支柱，特别是有些很常用但却是前端知识盲点的部分。

第 7 章介绍前端的单元测试与自动化测试，以及怎么使用可视化工具制作网页动画，还介绍了其他一些前端开发常用的工具，作为本书的一个补充内容。

在写作的过程中，我都是结合实际的经验进行阐述，并不像很多大学课本那样只讲理论。所以相对来说，本书看起来应该会比较生动，并且很多章节都是图文并茂的。

本书读者对象

本书适用于以下读者对象：

- 具有一定的前端基础，想要找一本高阶的、能提升水平的书；
- 刚毕业，没有什么实践经验，需要一本有实践指导作用的书；
- 已经工作了，想要学习一下其他人的前端开发经验；
- 不是做前端开发，但是有编程基础，想要深入理解前端是怎么运作的，或者是想加深理解 HTTP 之类的计算机基础知识。

如何阅读本书

如果你一点编程经验都没有，可能不太适合阅读本书，你要是不知道什么是变量，什么是 HTML，应该先读一些编程入门书籍。

读者可以从头看到结尾，我相信每一篇看完都会有收获的。或者有针对性地看，例如，你觉得自己在计算机基础里的网络协议、数据结构算法等方面比较薄弱，可以直接看第 5 章；如果你对 HTML5 比较感兴趣可以直接看第 4 章。在阅读的过程中，建议读者都实际操作一遍，而不仅仅是当作睡前读物，因为只有自己动手实践才能识别书中的真伪并且加深理解。所以本书不提供相关源码等资源，读者可自行根据书中描述动手练习。

致谢

在本书的写作和出版过程中得到了很多人的帮助，感谢我的家人对我写作的支持和鼓励，

感谢人人网同事在写作过程中提出的建议和对错误的修正，感谢机械工业出版社华章分社对本书出版付出的努力，特别是杨福川编辑对本书的策划以及李雷鸣老师的认真审阅、还要感谢阮一峰、大漠老师在百忙之中审阅本书、认可本书，并为本书写推荐语。

由于水平有限，书里难免会有一些不足和错误的地方，虽经过几番修改，可能还会有些许问题，欢迎读者朋友对本书的内容积极讨论，提出意见。我的邮箱是 liyincheng@m.scnu.edu.cn。

李银城

2017 年 12 月 17 日

目　录 *Contents*

第 1 章 Chapter 1

HTML/CSS 优化

切图是作为前端的一项基本技能，切图切得好，能够简化后续写 JS 的逻辑，有些交互甚至不用写 JS 就能完成。一方面 HTML/CSS 越来越强大了，另一方面 HTML/CSS 是浏览器提供的特性，只要写几个标签、写几行样式，一个好看的排版就出来了。善于使用浏览器提供的便利进行开发，能够简化代码，提高编程效率。

一般人都认为切图就是静态的，是死的，其实不然，一个好的切图除了好看之外，应该还要具备良好的交互性，是活的。而这不需要借助 JS 也能实现，而且比写 JS 更加方便。

不过也有人认为切图是比较低端的活儿——传说中程序员的鄙视链，写 C 的鄙视写 C++ 的，写 C++ 的鄙视写 Java 的，写 Java 的鄙视那些认为 HTML/CSS 是一门编程语言的人，如图 1-1 漫画所示。

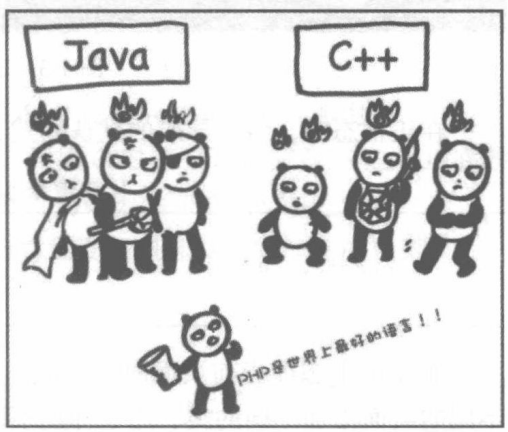


图 1-1　程序员的鄙视链（图片来自网络）

所以切图真得是很低端的工作吗？其实不然。

有人向大师提问，如何成为一名优秀的小提琴家，大师回答，先成为一名优秀的人，再成为一名优秀的音乐家，最后再成为一名优秀的小提琴家。而怎么成为一名优秀的前端？我认为要先成为一名优秀的人，然后再成为一名切图优秀的前端，最后再成为一名优秀的前端。这个类比虽然有点牵强，但是切图确实是一门技术活。

切图有三境界：第一境界——长得好看，长得好看方能让人有兴趣去了解你的思想；

第二境界——灵活，可根据数据长短扩展，维护方便；第三境界——友好的交互和用户体验，例如能否自动监听回车提交。

Effective 前端 1：能用 HTML/CSS 解决的问题就不要用 JS

为什么说能使用 HTML/CSS 解决的问题就不要使用 JS 呢？两个字，因为简单。简单就意味着更快的开发速度，更小的维护成本，同时往往具有更好的体验，下面介绍几个实例。

导航高亮

导航高亮是一种很常见的需求，包括当前页面的导航在菜单里面高亮和 hover 时高亮。你可以用 JS 控制，但其实用一点 CSS 技巧就可以达到这个目的，而不需要使用 JS。如图 1-2 和 1-3 所示。

图 1-2 HOME 菜单高亮

图 1-3 EVALUATION 菜单高亮

在正常态时，每个导航的默认样式为代码清单 1-1 所示：

代码清单 1-1 未选中态菜单是暗的

```
nav li{
    opacity: 0.5;
}
```

而在选中态即当前页面时，导航不透明度为 1。为了实现这个目的，首先通过 body 给不同的页面添加不同的类，用于标识不同的页面，如代码清单 1-2 所示：

代码清单 1-2 不同页面 body 的 class 不一样

```
<!-- home.html -->
<body class="home"></body>
<!-- buy.html -->
<body class="buy"></body>
```

所有的 li 也用 class 标识，为了有一个一一对应的关系，如代码清单 1-3 所示：

代码清单 1-3 导航 li 的 class

```
<li class="home">home</li>
<li class="buy">buy</li>
```

然后就可以设置当前页面的样式，覆盖掉默认的样式，如代码清单 1-4 所示：

代码清单 1-4 通过 body 和 li 的类建立起一一对应的关系

```
body.home nav li.home,
body.buy nav li.buy{
    opacity: 1;
}
```

这样，如果当前页面是 home，则 body.home nav li.home 这条规则将生效，home 的导航高亮。

这个技巧在《精通 CSS》这本书里面有提及。如果你用 JS 控制，那么在脚本加载好之前，当前页面是不会高亮的，而当脚本加载好之后会突然高亮。所以这种情况下用 JS 吃力不讨好。

同时，hover 时的高亮可以用 CSS 的 :hover 选择器实现，如代码清单 1-5 所示：

代码清单 1-5 hover 高亮

```
nav li:hover{
    opaciy: 1;
}
```

加上 :hover 选择器后的优先级将会高于原本的优先级，鼠标 hover 的时候将会覆盖默认样式，即高亮生效。

你也可以用 JS 的 mouse 事件实现此功能，但 JS 会在 mouseover 的时候添加一个类，mouseleave 的时候移除掉这个类，这样就变复杂了，而用 CSS 甚至可以兼容不支持 JS 的浏览器，所以，推荐使用 CSS。一个纯展示的静态页面，为啥要写 JS 呢，是吧。

注意这个 hover 选择器特别好用，几乎适用于所有需要用鼠标悬浮时显示的场景。

鼠标悬浮时显示

鼠标悬浮的场景十分常见，例如导航菜单，如图 1-4 所示，当鼠标 hover 到某个菜单时，它的子菜单就显示出来：

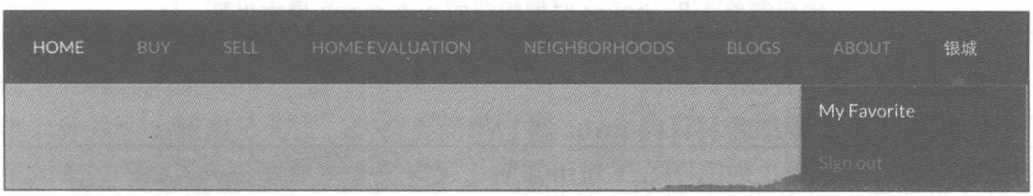


图 1-4 hover 菜单时显示下拉选项

还有像在地图里面，鼠标悬浮到某个房子图标时，显示这个房子的具体信息，如图 1-5 所示。

图 1-5 hover 图标时显示它的具体信息

这类场景的实现，一般要把隐藏的对象如子菜单、信息框作为 hover 目标的子元素或者相邻元素，才方便用 CSS 控制，例如，上面的菜单是把 menu 当作导航的一个相邻元素，HTML 结构如代码清单 1-6 所示：

代码清单 1-6 菜单 menu 紧挨着 user

```
<li class="user">用户</li>
<li class="menu">
    <ul>
        <li>账户设置</li>
        <li>登出</li>
    </ul>
</li>
```

menu 在正常态下是隐藏的，如代码清单 1-7 所示：

代码清单 1-7 菜单默认隐藏

```
.menu{
    display: none;
}
```

当导航 hover 时结合相邻选择器，把它显示出来，如代码清单 1-8 所示：

代码清单 1-8 hover 时把相邻的 sub-menu 显示出来

```
.user:hover + .menu{
    display: list-item;
}
```

注意这里使用了一个相邻选择器，这也是上面说的为什么要写成相邻的元素。而 menu 的位置可以用 absolute 定位。

同时，menu 本身 hover 的时候也要显示，否则鼠标一离开导航的时候，菜单就消失了，如代码清单 1-9 所示：

代码清单 1-9 menu hover 时也要显示

```
.menu:hover{
    display: list-item;
}
```

这里会有一个小问题，即 menu 和导航需要挨在一起，如果中间有空隙，上面添加的菜单 hover 就不能发挥作用了，但是实际情况下，从美观的角度，两者是要有点距离的。这个其实也好解决，只要在 menu 上面再画一个透明的区域就好了，如图 1-6 中选中的方块。

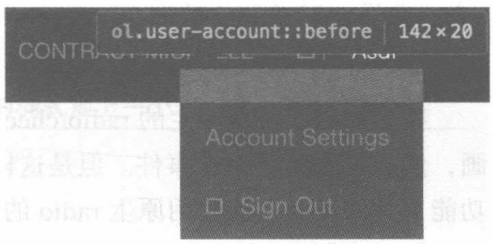


图 1-6 填充空白区域

可以用 before/after 伪类用 absoute 定位实现，如代码清单 1-10 所示：

代码清单 1-10 使用 before 画小蓝块

```
ul.menu:before{
    content: "";
    position: absolute;
    left: 0;
    top: -20px;
    width: 100%;
    height: 20px;
}
```

这样鼠标往下移的时候就会马上 hover 到 menu 身上，而不会因为中间的空白导致 menu 出不来了。

如果既写了 CSS 的 hover，又监听了 mouse 事件，用 mouse 控制显示隐藏，双重效果会有什么情况发生？如果按正常思路，在 mouse 事件里面 hover 的时候，添加了一个 display: block 的 style，会覆盖掉 CSS 的设置。也就是说，只要 hover 一次，CSS 的代码就不管用了，因为内联样式的优先级会高于外联的。但是实际情况下会有意外发生，那就是在移动端 Safari 上面，触摸会触发 CSS 的 hover，并且这个触发会很高概率地先于 touchstart 事件，此时会先判断当前是显示还是隐藏的状态，由于 CSS 的 hover 发挥了作用，所以判断为显示，然后又把它隐藏了。也就是说，点一次不出来，要点两次。所以最好别两个同时写。

第二种方法，使用子元素，这个更简单。把 hover 的目标和隐藏的对象当作同一个父容器的子元素，然后 hover 写在这个父容器上面就可以了，不用像上面那样，隐藏的元素本身也要写个 hover。如代码清单 1-11 所示：

代码清单 1-11　marker hover 时把它的子元素 detail-info 放出来

```
.marker-container .detail-info{
    display: none
}

.marker-container:hover .detail-info{
    display: block
}
```

自定义 radio/checkbox 的样式

我们知道，使用原生的 radio/checkbox 是不可以改变它的样式的，得自己用 div/span 去画，然后再去监听单击事件。但是这样需要自己去写逻辑控制，例如实现 radio 按钮单选的功能，另外没有办法使用原生 radio 的 change 事件，没有用原生的来得方便。

但是实际上可以用一点 CSS3 的技巧实现自定义的目的，如图 1-7 所示，就是用原生 radio 实现的。

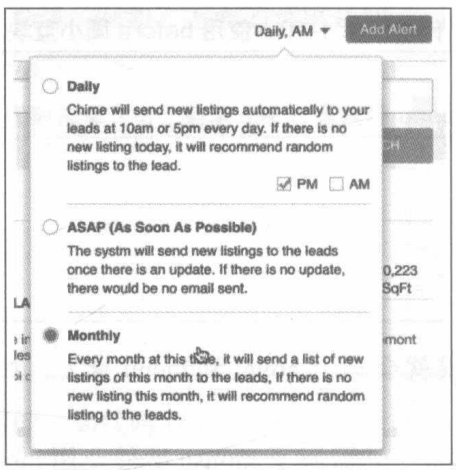


图 1-7　左边的圆框和上面的方框都是用原生实现的

这主要是借助了 CSS3 提供的一个伪类 :checked，只要 radio/checkbox 是选中状态，这个伪类就会生效，因此可以利用选中和非选中这两种状态，去切换不同的样式。代码清单 1-12 是把一个 checkbox 和一个用来自定义样式的 span 写在一个 label 里面，同时 checkbox 始终隐藏。

代码清单 1-12　实现自定义单选、多选按钮样式

```
<style>
input[type=checkbox]{
    display: none;
}
/* 未选中的 checkbox 的样式 */
```

```
.checkbox{
    /* 实现略 */
}
</style>
<label>
    <input type="checkbox">
    <span class="checkbox"></span>
</label>
```

写在 label 里面是为了能够在单击 span 的时候改变 checkbox 的状态。最后，再改一下选中态的样式即可，如代码清单 1-13 所示：

代码清单 1-13 选中时，把单选框的样式加上一个勾

```
input[type=checkbox]:checked + .checkbox{
    /* 实现略 */
}
```

注意，这一步很关键，添加一个打勾的背景图也可以，使用图标字体也可以（我们将在第 4 章介绍图片字体）。:checked 兼容性还是比较好的，只要你不需要兼容 IE8 就可以使用，换句话说只要你可以用 nth-of-type，就可以用 :checked。

多列等高

多列等高的问题是这样的，排成一行的几列由于内容长短不一致，导致容器的高度不一致，如图 1-8 所示。

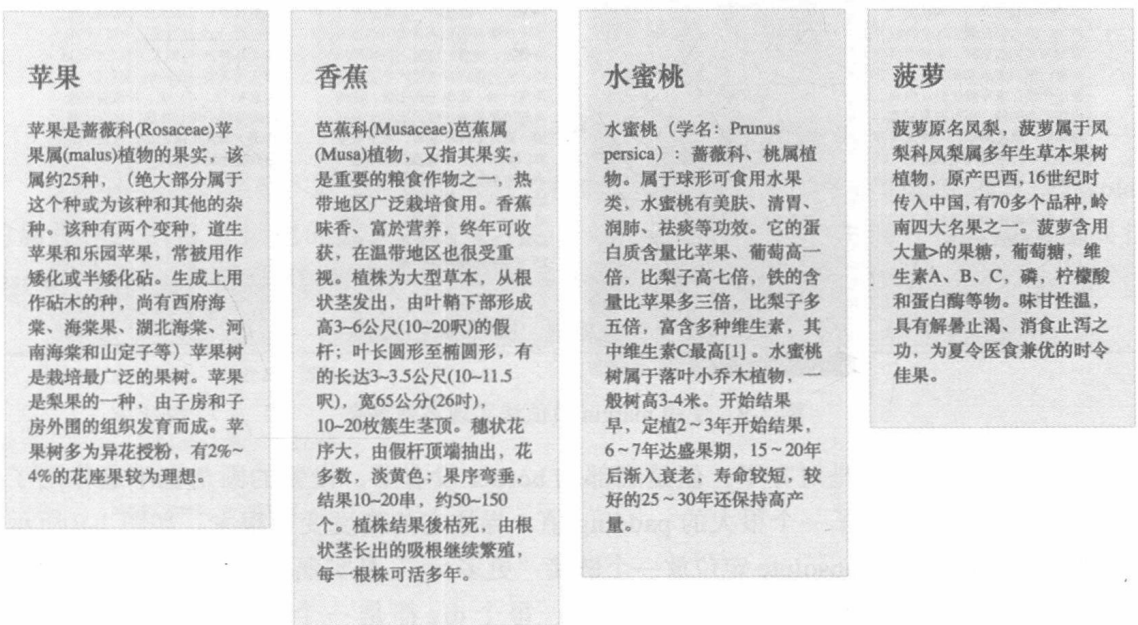


图 1-8 多列排列时由于内容长短不一，导致各列对不齐

你可以用 JS 计算一下，以最高的一列的高度去设置所有列的高度，然而这会造成页面闪动，刚开始打开页面的时候高度不一致，然后突然又对齐了。解决办法主要有两种：

第一种是每列来一个很大的 padding，再来一个很大的负的 margin 值矫正回去，就对齐了，这种方法在《精通 CSS》里面提到过，如代码清单 1-14 所示：

代码清单 1-14　借助 margin/padding 实现等高对齐

```
<style>
    .wrapper > div{
        float: left;
        padding-bottom: 900px;
        margin-bottom: -880px;
        background-color: #ececec;
        border: 1px solid #ccc;
    }
</style>
<div class="wrapper">
    <div>column 1</div>
    <div>column 2</div>
    <div>column 3</div>
    <div>column 4</div>
</div>
```

效果如图 1-9 所示。

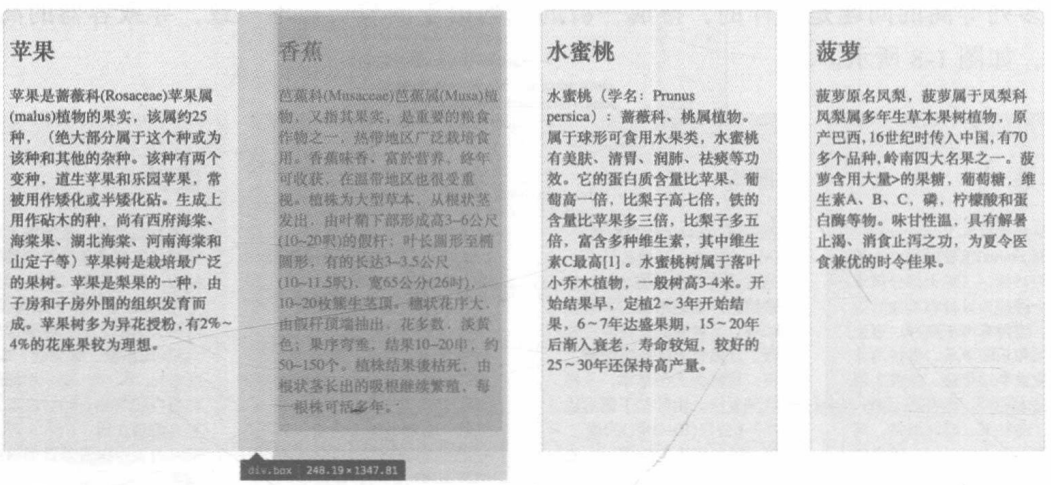


图 1-9　使用 margin 负值法实现多列等高

你会发现，这样做是对齐了，但是底部的 border 没有了，设置的圆角也不起作用了，究其原因，是因为设置了一个很大的 padding 值，导致它的高度变得很大，如图 1-9 所示。所以如果你想在底部用 absolute 定位放一个链接“更多>>”是实现不了的。

第二种办法是借助 table 的自适应特性，每个 div 都是一个 td，td 肯定是等高的，HTML 结构不变，CSS 改一下，如代码清单 1-15 所示：

代码清单 1-15 借助 td 实现多列等高

```
.wrapper{
    display: table;
    border-spacing: 20px;   /* td间的间距 */
}

.wrapper > div {
    display: table-cell;
    width: 1000px;          /*设置很大的宽度，table自动平分宽度 */
    border-radius: 5px;     /*这里设置圆角就正常了 */
}
```

对齐效果如图 1-10 所示。

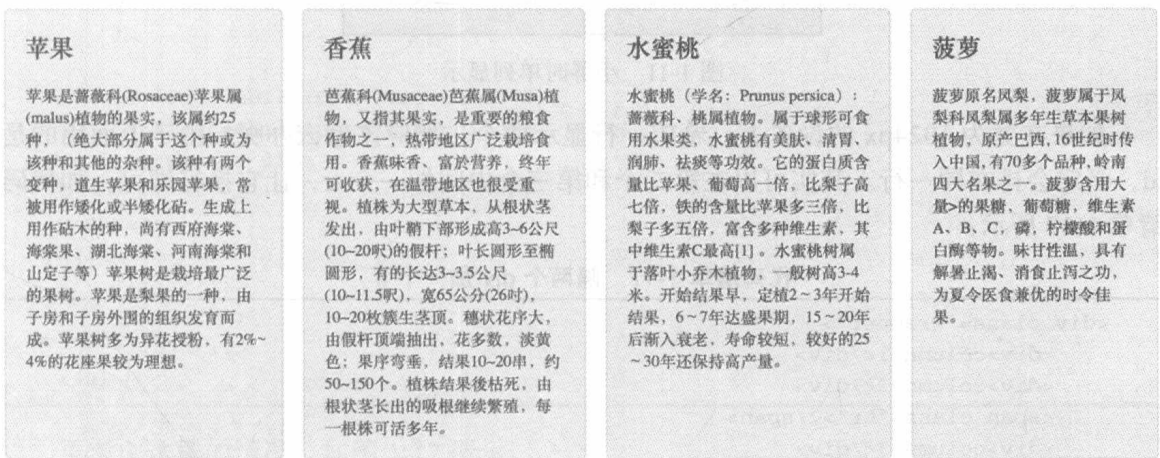


图 1-10 借助表格元素特性实现多列等高

这样还有一个好处，就是在响应式开发的时候，可以借助媒体查询动态地改变 display 的属性，从而改变排列的方式。例如在小于 500px 时，每一列占满一行，那么只要把 display: table-cell 覆盖掉就好了，如代码清单 1-16 所示：

代码清单 1-16 小屏时改成单列显示

```
@media (max-width: 500px){
    .wrapper{
        display: block;
    }
    .wrapper > div{
        display: block;
        width: 100%;
    }
}
```