05 页面结构基础

更新时间：2019-06-21 14:49:22

> 天才免不了有障碍，因为障碍会创造天才。
>
> ——罗曼·罗兰

这一节我们来介绍下页面结构的基础，了解一下盒模型、文档流和页面层级这几个概念。在页面里，盒模型是最基础的结构，每一个HTML节点都对应一个盒模型；多个盒模型的排布，就构成了文档流；而在盒模型堆叠排列的时候会出现相互遮挡的问题，盒模型有层级的关系，这就是页面的层级。了解了页面的结构基础，后面在做各种布局的时候，就能把一个个的元素按着页面的基础结构来划分，有利于理解各种布局的原理。下面我们来详细的介绍下这几个概念。

盒模型

在浏览器中，每一个 DOM 节点渲染后，都会在屏幕上占用一个方形的区域，这个方形的区域就被称为盒子，我们把这种渲染方式叫盒模型。在盒模型中，我们主要介绍盒模型的主要属性、两种盒模型和边距折叠这三个内容：

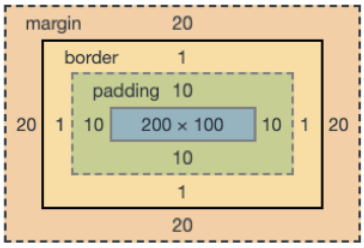
一、盒模型主要属性

在说盒模型的属性时，为了让同学们好理解，我们以下面这三个画框做类比来说明。

上面的三幅画，其实就是 HTML 中的三个盒子。对盒子大小和位置能产生影响的一共有四类属性，他们分别是：

- **宽度（width）和高度（height）**，这两个属性分别决定了一个盒子的宽度和高度，在标准盒模型中，这个宽高指定的就是最里层内容区的宽度和高度，对应画框中最里面带色彩的部分。
- **内边距（padding）**，当内容区和边框需要离开一定的间距，避免布局太过拥挤时，就可以用内边距来指定他们隔开的距离。在画框中就对应着内容区和边框中间夹着的那部分白色的区域。
- **边框（border）**，边框比较好理解，就是每幅画的外框。
- **外边距（margin）**，外边距是用来限制盒子与盒子中间的距离的，在上图中，三幅画边框与边框中间空出来的距离，就由外边距来指定。

把上面的画框抽象一下，每个盒子的结构就是下图所示：



二、两种盒模型

我们在盒模型的属性中提到过标准盒模型，这是因为除了标准盒模型外，还有在IE浏览器中使用的怪异盒模型。我们现在用一段代码分别演示这两种盒模型。

```
<html>
  <head>
    <style>
      div{
        width: 200px;
        height: 100px;
        padding: 10px;
        border: 1px solid #000;
        margin: 20px;
      }
      .content-box{
        box-sizing: content-box;
      }
      .border-box{
        box-sizing: border-box;
      }
    </style>
  </head>
  <body>
    <!-- 标准盒模型 -->
    <div class="content-box">
      我是content-box
    </div>
    <!-- 怪异盒模型 -->
    <div class="border-box">
      我是border-box
    </div>
  </body>
</html>
```

这两个盒子中，使用了 box-sizing:content-box;属性的就是标准模式，用了 box-sizing:border-box;属性的就是怪异模式。

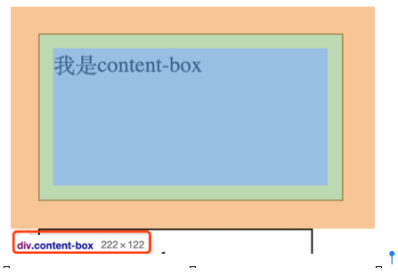
我们给盒子都设置成 200px 的宽度，100px 的高度，10px 的 padding 和 1px 的 border，代码里的 margin 是处理盒子间距离的这里先不考虑。

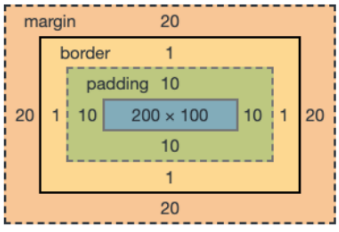
- **标准盒模型**，标准盒模型这个标准是由 W3C 组织制定的，现在除了低版本IE以外，基本所有浏览器都遵循这个标准。标准盒模型中，width 和 height 属性所指定的宽高就是实际内容区的大小，而盒子实际大小是：

```
横向空间：width  + padding宽度 + border宽度
纵向空间：height + padding宽度 + border宽度
```

根据上面代码的运行结果，可以看出来这个盒子的总宽度是 222px，也就是 200px 的 width，加上两边各 10px 的 padding，再加上两边各 1px 的边框，实际占用屏幕的宽度就是

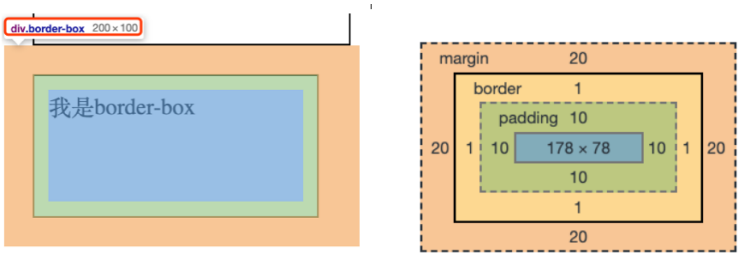
```
200px + 10px*2 + 1px*2 = 222px。
```





- **怪异盒模型**，怪异盒模型是微软在 IE6、7、8 中使用的一种盒模型，所以也把怪异盒模型叫做 IE盒模型。在怪异模式下，width 和 height 做指定的宽高就是盒子的实际宽高，而它内容区部分的大小是在 width 或 height 指定尺寸的基础上，再减去 border 和 padding 所占的宽度。

在怪异盒模型中，我们设置的宽度也是 200px，它的盒子占用的横向空间就是 200px，但是内容区的宽度变成了178px，这就是用它的宽度减去了两边 padding 和两边 border 的结果。



这两种模型的出现还是有一段故事的。微软开始设计的 IE盒模型是有它的考虑的，这种模式下只要设置了 width 和 height，那么这个盒子的尺寸就是可控的了，不会因为设置 border 或者 panding 造成页面错乱，最差的情况也就是内容区的空间不对，而不会影响到其他盒子。所以在 W3C 组织提出来标准盒模型的时候，微软坚持自己的设计，并没有遵从 W3C 的标准，只是提供了通过 DOCTYPE 指定解析方式来兼容标准盒模型。但到后面微软的市场份额越来越少，开发人员也都倾向遵从 W3C 的标准，微软之后只能认怂了，从 IE9 开始都默认使用了标准盒模型。

但刚才说了，标准盒模型是有缺陷的，在标准模式下，我们设置盒子的宽高并不是它实际所占空间，所以在布局的时候，就要再把盒子的 border 和 padding 计算一下来给这个盒子分配空间。但是在用 px 这种绝对单位指定盒子宽高的时候还能算一下应该把盒子设置成多大，但如果是用百分比指定盒子宽高的时候，就没办法算了。另外 border 只支持绝对宽度，而 px 和百分比混在一起做运算是比较困难的。比如一行四个盒子，每个盒子有 1px 的边框，这时候按着标准盒模型给每个盒子25%的宽度，这四个盒子就会折行，不能放在一行。

这个时候就体现出怪异模型的好处了，假如一行四个盒子平均分配，在怪异模式中只需要每个盒子宽度 25%，然后 border 和 padding 随便设置，浏览器会自己去算内容区应该有多大的。W3C 应该也是意识到了这个问题，但标准模式用了这么多年，肯定是回不去了，也不能自己打脸说这些年我错了。所以在 CSS3 中做了补救，就是加上了上面使用过的 box-sizing 属性。这样可以对外声称我们这是升级，开发者能自己指定用哪种模型了，两种模式还能混着用，但就是没提过这些年标准盒模型给前端同志们带来的困扰。

三、外边距折叠

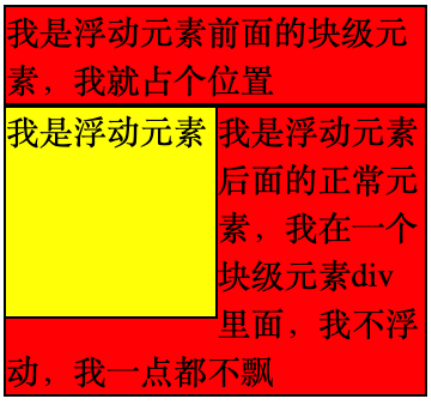
最后说一个特殊的概念，就是外边距折叠。我们在前面讨论的都是内容区、padding 和 border 间的关系，在这我们来说一下 margin。margin 的意思就是距别的框之间的距离，这很好理解。但是如果两个盒子都设置了 margin，在排列这两个盒子的时候并不会把两个 margin 值的和作为两个盒子的间距。比如上下排列的盒子A和盒子B，盒子A的下边距是10px，盒子 B 的上边距是 20px，那么在渲染时候，它俩之间的距离将会是 20px。就像A说我 10 米之内寸草不生；B说我20米内片甲不留。那把这俩盒子中间隔上个20 米，它俩就都是安全的了，而用不着隔开 30 米。盒子间距的实际值其实就是取两个盒子间margin值较大的那个。在这要敲下黑板，外边距折叠是面试中常问的内容，一定要了解！

文档流

说完盒子模型，就可以说文档流了。所谓文档流，就是DOM节点排版布局过程中，元素会自动从左往右，从上往下的流式排列。在一行里，节点是按着从左到右排列，当这一行内容排满的时候，就再单起一行排余下的内容，再排满就再新开一行。这里面有几个地方要注意：

1. 在正常布局下，块级元素会自己占用一行，有几个块级元素就会有多少行。而行内元素才会从左到右的排列。
2. 如果某元素用到了绝对定位或固定定位（absolute和fixed），那这个元素就会脱离文档流，它前后的元素会忽略它的位置。而这个脱离文档流的元素会按着指定的位置重新定位，如果没指定位置，则会按着脱离文档流之前的位置进行定位。

3. 如果某元素使用了浮动属性（float），那这个元素也会脱离文档流，浮动元素后面如果是块级元素的话，就会忽略它的存在。但它和绝对定位不同的是，浮动元素虽然也会脱离文档流，但会在文本的排布中占有位置，浮动元素后面的文本会环绕着它进行排布。从下面的图可以看出来，两个红色的块是标准的文档流，已经没黄色的浮动的盒子什么事了，但是第二个红色块中的文本却受到了黄色块的影响。



页面层级

刚说了文档流在排列的时候，有元素脱离文档流的情况，这时候不同的元素就会出现层叠的情况。或者在标准文档流中，通过设置负值的 margin，也可能会导致元素互相遮盖。从这种遮盖关系来看，就知道 HTML 其实是一个三维的空间，有平面的 x,y 位置，也有 z 轴上的层叠关系。
在默认的情况下，即不指定层级的时候，所有元素是按下面的层级排列的：

1. HTML 在渲染的时候最先渲染的是标准文档流，所以标准文档流中的内容会被排在最下一层，标准文档流中如果还有层级，那就是后出现的会挡住先出现的。
2. float 元素在标准文档流之后渲染，所以 float 会在标准文档流的上一层。
3. 绝对定位的元素最后渲染，所以默认情况下绝对定位元素会排在最上层。

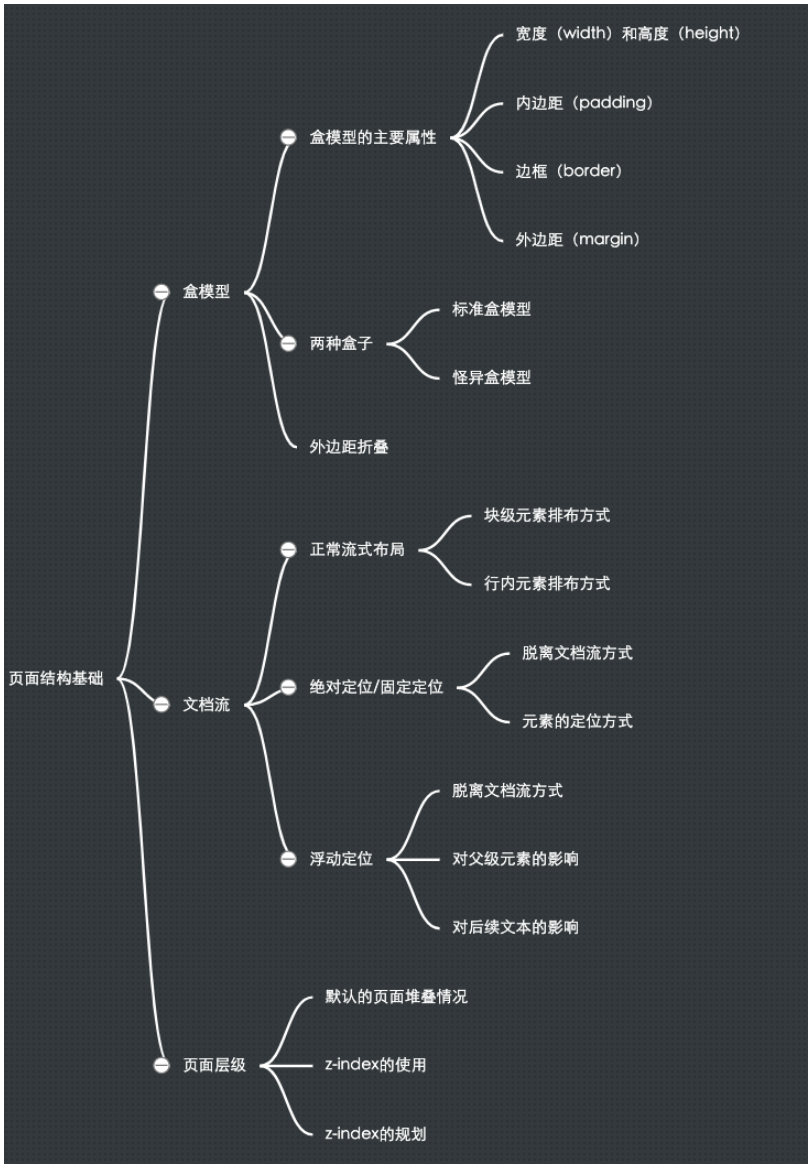
如果对默认的层叠关系不满意，我们就可以用 z-index 数值手动指定各个元素的层级关系。但这里要注意，z-index 只对已经有定位的元素生效（position:relative/absolute/fixed），所以当写样式的时候遇到 z-index 没有达到你预期效果的时候，考虑一下这种情况。

在使用 z-index 的时候，我们最好对整个项目的层级关系有个提前的设计，不要随便用。当我们的页面需要分成多个层级的时候，可以对每个层级设置一个 z-index 使用的范围。比如一个页面主要分为内容层、导航层和蒙版提示层这三层，那么就可以限制内容层的 z-index 值从 100 到 200 之间；导航层要覆盖住内容层，z-index 可以用 200 到 300 这个范围的值；蒙版提示层要盖住一切，z-index 可以用 300 到400 这个范围的值。这样每一层的层级关系就是明确的，不会互相影响。

小结

这一节的内容里，要了解 DOM 树中的每个节点在渲染的时候的存在形式–盒模型，知道盒模型的由哪些属性构成的，两种盒模型的关系以及边距折叠的概念。然后是这些盒子在页面中的排布方式–文档流，这一部分要理解一下脱离文档流的特殊情况。最后讲了文档流排布时会遇到的层叠问题–页面层级，这部分要理解页面层级的概念，学会用合理的方式规划页面层级。

这一节的关键知识点如下：



这一节就到这里，下一节我们将进入CSS选择器的学习。

04 课前准备

06 CSS选择器（一）之基础选择器