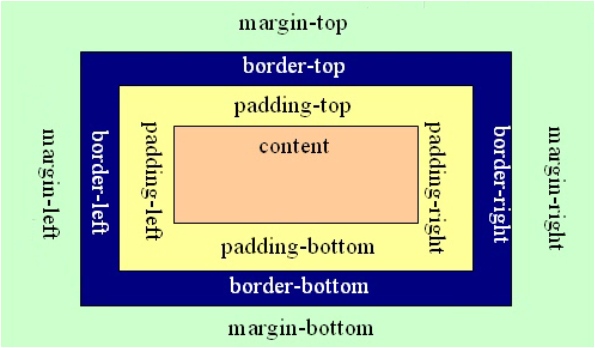
# CSS盒子模型

　　CSS中， Box Model叫盒子模型（或框模型），Box Model规定了元素处理元素内容（element content）、内边距（padding）、边框（border） 和 外边距（margin） 的方式。在HTML文档中，每个元素（element）都有盒子模型，所以说在Web世界里（特别是页面布局），Box Model无处不在。下面是Box Model的图示：

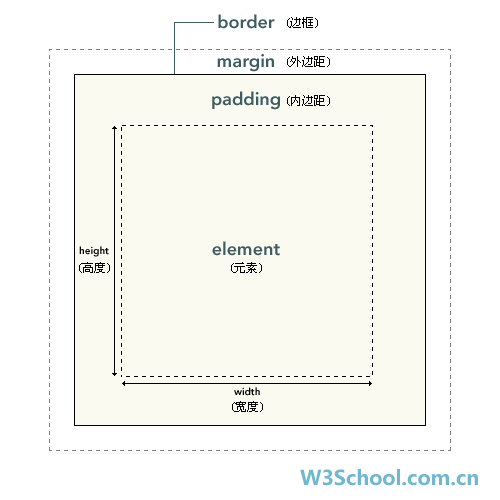


　　　　　　　　　　　　　　　　　　Box-Model 1

　　说明：上图中，由内而外依次是元素内容（content）、内边矩（padding-top、padding-right、padding- bottom、padding-left）、边框（border-top、border-right、border-bottom、border- left）和外边距（marging-top、margin-right、margin-bottom、margin-left）。

　　内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。而且，外边距可以是负值，而且在很多情况下都要使用负值的外边距。

　　下图是W3School的Box Model 图解：



　　　　　　　　　　　　Box-Model 2（W3School）

　　说明：

　　1.和第一幅图一样，在上图中，元素框的最内部分是实际的内容（element）；直接包围内容的是内边距（padding），内边距呈现了元 素的背景（background）；内边距的边缘是边框（border）；边框以外是外边距（margin），外边距默认是透明的，因此不会遮挡其后的任 何元素（其实元素的margin就是其所在父元素的padding）。元素的背景应用于由内容和内边距、边框组成的区域。

　　2.内边距、边框和外边距都是可选的，默认值是零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 margin 和 padding 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器（\*）对所有元素进行设置：

/\*设置所有元素的外边距和内边矩为0\*/\* {

  margin: 0;

  padding: 0;

}

　　3. 在 CSS 中，width 和 height 指的是内容区域（element）的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，以下是CSS代码：

#box {

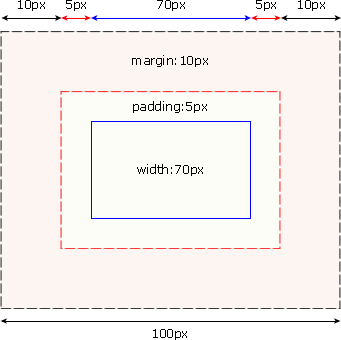
  width: 70px;

  margin: 10px;

  padding: 5px;

}

下图是对上面CSS代码的解释：



 　　以下是一个完整的CSS Box Model布局的示例。

　　以下是上面示例的效果截图：



# [block，inline和inline-block概念和区别](http://www.cnblogs.com/KeithWang/p/3139517.html)

## 总体概念

1. block和inline这两个概念是简略的说法，完整确切的说应该是 block-level elements (块级元素) 和 inline elements (内联元素)。block元素通常被显示为独立的一块，会单独换一行；inline元素则前后不会产生换行，一系列inline元素都在一行内显示，直到该行排满。
2. 大体来说HTML元素各有其自身的布局级别（block元素还是inline元素）：
   * 常见的块级元素有 DIV, FORM, TABLE, P, PRE, H1~H6, DL, OL, UL 等。
   * 常见的内联元素有 SPAN, A, STRONG, EM, LABEL, INPUT, SELECT, TEXTAREA, IMG, BR 等。
3. block元素可以包含block元素和inline元素；但inline元素只能包含inline元素。要注意的是这个是个大概的说法，每个特定的元素能包含的元素也是特定的，所以具体到个别元素上，这条规律是不适用的。比如 P 元素，只能包含inline元素，而不能包含block元素。
4. 一般来说，可以通过display:inline和display:block的设置，改变元素的布局级别。

## block，inline和inline-block细节对比

* display:block
  1. block元素会独占一行，多个block元素会各自新起一行。默认情况下，block元素宽度自动填满其父元素宽度。
  2. block元素可以设置width,height属性。块级元素即使设置了宽度,仍然是独占一行。
  3. block元素可以设置margin和padding属性。
* display:inline
  1. inline元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化。
  2. inline元素设置width,height属性无效。
  3. inline元素的margin和padding属性，水平方向的padding-left, padding-right, margin-left, margin-right都产生边距效果；但竖直方向的padding-top, padding-bottom, margin-top, margin-bottom不会产生边距效果。
* display:inline-block
  1. 简单来说就是将对象呈现为inline对象，但是对象的内容作为block对象呈现。之后的内联对象会被排列在同一行内。比如我们可以给一个link（a元素）inline-block属性值，使其既具有block的宽度高度特性又具有inline的同行特性。

# 替换和不可替换元素

从元素本身的特点来讲，可以分为替换和不可替换元素。

## a) 替换元素

替换元素就是浏览器根据元素的标签和属性，来决定元素的具体显示内容。

例如浏览器会根据<img>标签的src属性的值来读取图片信息并显示出来，而如果查看(X)HTML代码，则看不到图片的实际内容；又例如根据<input>标签的type属性来决定是显示输入框，还是单选按钮等。

(X)HTML中的<img>、<input>、<textarea>、<select>、<object>都是替换元素。这些元素往往没有实际的内容，即是一个空元素，例如：

**[html]** [view plain](http://blog.csdn.net/jlds123/article/details/8647448) [copy](http://blog.csdn.net/jlds123/article/details/8647448)

1. **<img** src=”cat.jpg” **/>**
2. **<input** type="submit" name="Submit" value="提交" **/>**

浏览器会根据元素的标签类型和属性来显示这些元素。可替换元素也在其显示中生成了框。

## b) 不可替换元素

(X)HTML 的大多数元素是不可替换元素，即其内容直接表现给用户端（例如浏览器）。例如：

**[html]** [view plain](http://blog.csdn.net/jlds123/article/details/8647448) [copy](http://blog.csdn.net/jlds123/article/details/8647448)

1. **<p>**段落的内容**</p>**

段落<p>是一个不可替换元素，文字“段落的内容”全被显示。

## 2. 显示元素

除了可替换元素和不可替换元素的分类方式外，CSS 2.1中元素还有另外的分类方式：块级元素（block-level）和行内元素（inline-level，也译作“内联”元素）。

### a) 块级元素

在视觉上被格式化为块的元素，最明显的特征就是它默认在横向充满其父元素的内容区域，而且在其左右两边没有其他元素，即块级元素默认是独占一行的。

典型的块级元素有：<div>、<p>、<h1>到<h6>，等等。

通过CSS设定了浮动（float属性，可向左浮动或向右浮动）以及设定显示（display）属性为“block”或“list-item”的元素都是块级元素。

但是浮动元素比较特殊，由于浮动，其旁边可能会有其他元素的存在。而“list-item”（列表项<li>），会在其前面生成圆点符号，或者数字序号。

### b) 行内元素

行内元素不形成新内容块，即在其左右可以有其他元素，例如<a>、<span>、<strong>等，都是典型的行内级元素。

display属性等于“inline”的元素都是行内元素。几乎所有的可替换元素都是行内元素，例如<img>、<input>等等。

不过元素的类型也不是固定的，通过设定CSS 的display属性，可以使行内元素变为块级元素，也可以让块级元素变为行内元素。

替换元素一般有内在尺寸，所以具有width和height，可以设定。例如你不指定img的width和height时，就按其内在尺寸显示，也就是图片被保存的时候的宽度和高度。

对于表单元素，浏览器也有默认的样式，包括宽度和高度。