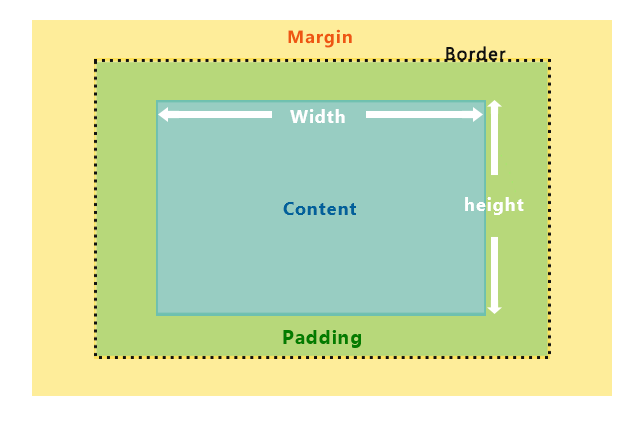
# CSS框模型（Box Model）【盒模型】



解释：

1. 图中，margin、padding、border各有上下左右，分别对应视觉上的上下左右。
2. Padding 、Margin 、Border默认是透明的。

注意：

* Background的区域包括= content + padding + border，也就是说，若如图中border是虚线的话，会显示出橙色的background。
* Padding默认值为0。
* Marin 、 Width 、Height默认值为auto。

# 盒模型在各类元素中的表现：

## 非替换块元素

特点：块级元素会自动充满父元素。

块级元素会按照标准盒模型正常渲染。例如（div，p，ul…….）

按照块元素的特点、CSS默认值和渲染方式，可以得到以下情况：

1. 当margin-left 、margin-right 、width都为固定值时，如果margin-left + margin-right + width 小于父元素的width，会自动重置margin-right以充满父元素。
2. margin为负值时（padding、width、height、border不能为负值），可以造成子元素的溢出现象。例如margin-left（-30） + margin-right + width（+30） = Width（父元素）。

Tips:可以使用负边距实现特殊布局，如bootstrap中.row的实现、新闻版块的整体偏移。

1. 除边框外其他属性可以为百分比值，其值根据父元素Width来计算。
2. 在垂直方向上Margin会有合并现象。

## 替换块元素

当width设置为auto时，元素的宽度是内容的固有宽度。【非替换块元素的width为auto时会优先自动填满父元素】其他表现与非替换块元素相同。例如 （img）。

## 非替换行内元素

例如（匿名字体即文本节点 span 、em 、strong 、 highlight……）。

非替换行内元素遵循着另外一套高度原则。但是CSS的盒模型在此仍然有所渲染。在此会牵扯到另外2个CSS属性：line-height 、vertical-align。

那么，行内元素到底怎么表现呢？

1. 在行内元素中，width 、 height 将失效，其具体值只有具体内容的宽、高决定。
2. margin-top、margin-bottom无论设置为何值，浏览器一律忽略为0。
3. margin-left、margin-right、border-left、border-right表现同块级盒模型。
4. 竖直方向上比较复杂，详见下：

想要了解竖直方向上的情况，首先应该知道以下几个概念：

1. em 框，即字体渲染所生成的一个框，其内容正好能包括文字。（具体情况请查看css权威指南）.

注意：em框的高度由font-size来决定。

1. line-height行高，就是属性line-height的值。(默认值：1.2em左右)

tip：使用行高时，设置为具体数值（px）会直接使用该值，使用百分比（%）值时，会使用父元素的line-height的100%来计算，使用相对值（em）时，是相对父元素的font-size计算（而不是该元素的font-size）。

1. 行间距，即line-height – font-size的值（可以为负值）。渲染时会将行间距除2并分别应用于元素的顶部和底部。
2. 行内框（行内高），单个行内元素最终被渲染出来的高度（由line-height和font-size决定）。
3. 行框。对于一行来说，就是这一行的高度。即在同一行，可能有许多不同的行内元素，例如一行中可能同时存在匿名文本 、span标签 、strong等，而其line-height和font-size可能不同，导致其不同的高度，所以行框指该行中出现的行内框的最高点和最低点的最小框。
4. 内容区：
   1. 对于行内非替换元素，内容区即em框。
   2. 对于替换元素，内容区指元素固有的高度 + 内边距 + 边框 + 外边距。

那么，我们来理一下网页中一行内容渲染的过程：

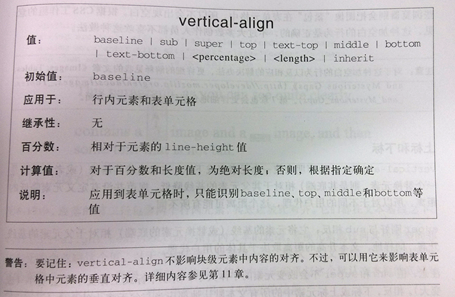
1. 确定该行中各个元素行内框的高度。
2. 对于各个内容区，根据相关的字体、元素等确定它在整行基线的上方和下方分别超出多少，然后将它们按照基线对齐。
3. 处理vertical-align。对于指定了vertical-align值的元素，确定其垂直偏移量。

注意：此时可能会改变元素的内容区和行内框大小。

1. 知道了所有行内框高度和位置。计算得到行框。

注意：CSS在渲染时，不管怎样一定会把html中的文本信息渲染出来的。所以当 line-height < font-size 时，行间距为复数，会导致行内框比em框小，而为保证字体的显示，行框会被强行撑开，此时会覆盖周围的元素，且覆盖行为具有不规律性。

那么，此时就必须要解释一下vertical-aligin属性(默认值baseline)了（估计很多人栽倒在它上或者觉得它很鸡肋？）



当vertical-align为不同值时，有不同的计算方法：

1. 数值（10px……），在baseline对齐的基础上，将元素上移（正数）或下移（负数）。
2. 百分比（30%），以改元素的line-height为100%，计算得到像素值，接下里与1相同。
3. 关键字：

* top / bottom：将元素行内框的顶端（底端）与包含该元素（元素所在行）的行框的顶部（底部）对齐。
* text-top / text-bottom：将元素行内框的顶部与父元素内容区的顶端（底端）对齐。
* middle：将元素行内框的垂直重点与父元素基线上0.5ex处的一点对齐。

注意：该值在不同浏览器和不同字体中差异较大。

* super / sub：将元素的内容区和行内框上移（下移）。

注意：上移（下移）距离并没有规范，差异性大，慎用！！！

OK，最后我们还剩下padding-top / padding-bottom / border-left / border-right没有说明，然而你是不是以为他们没有用？哼哼，并不是！

对于行内非替换元素而言，padding-top / padding-bottom / border-left / border-right是存在的，然而并不占实际空间（与outline相似的表现形式）。也就是说，padding-top / padding-bottom / border-left / border-right会被渲染出来，而且按照在文档出现的位置，后面的元素会覆盖前面的元素！

## 替换行内元素

注意：line-height对替换元素是没有用的。但是这个属性仍然可以在行内替换属性上继承。因为，vertical-align对替换元素是有用的，而当vertical-align值为百分比时要根据该元素的line-height来计算。此外，对于其他元素也需要通过line-height的继承来获得line-height值。

另外，相对于字体来说，替换行内元素是没有基线的，所以以其底端（如果有margin-bottom也要算上），即内容区的底端当做基线来对齐。

对于替换行内元素来说，负边距是使其与其他元素重叠的唯一方式。

## 表格元素(以行为主)

在html表格中，表元素table与非替换块元素相同。而内部表元素如td、tr、th、colgroup、col等会生成矩形框，但是它们没有外边距margin（也就是说对内部表元素设置margin是无效的）。注意：总标题例外！！！

表格的编排规则（略）——请自行查阅。

CSS并不建议对表元素使用定位。最好不要这么做。

在display对应值中，有如下值与表格元素对应：

display：table —— <table>;

display：inline-table —— 接近于inline-block;

display：table-row —— <tr>;

display：inline-header-group 、table-footer-group

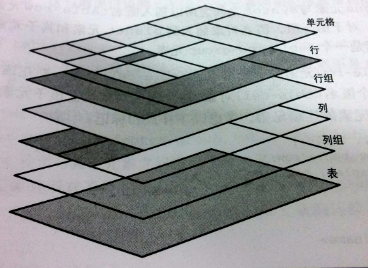
display：table-column —— <col>;

display：table-column-group —— <colgroup>;

display：table-cell —— <th> 、<td>;

display：table-caption —— <caption>;

注意：css中列（table-column）和列组（table-column-group）只能接受4种样式：border 、background 、width 、 visibility



如上图，在渲染表格元素时，会按照上图绘制7层。覆盖关系为上边覆盖下面。

表大小：

固定布局：（table-layout：fixed）【渲染速度很快】

1. Width不是auto的所有列元素会根据width值设置该列宽度。
2. 如果列为auto但是该列首行单元格的width不是auto，则根据改单元格宽度设置该列宽度。如果这个单元格跨多列，则宽度在这些列上平均分配。
3. 如果列和首行的宽度都为auto，会自动确定其大小，并且使其宽度尽可能相等。

注意：

* 此时，表的宽度为 （设置的width值 / 列的宽度综合）中的最大值。
* 若一个单元格的内容无法放下，该单元格的显示有overflow控制【英文会直接溢出，而中文会自动换行】

自动布局：（table-layout：auto）【默认值】【渲染速度慢，是一般div的3倍左右】

好吧，这是一个非常恶心和复杂的过程。

1. 对于一列中的各个单元格，计算最小和最大单元格宽度：
   1. 确定显示内容所需的最小宽度（就是说把单元格里的内容全部显示在一行所需要的宽度）。因为单元格的内容可以流入多行，不过不能超出单元格框。
      1. 如果单元格的width值大于最小可能宽度，则把最小单元格宽度设置为该width值。（如果单元格有设置width，而且width比把全部内容显示在一行还大，则，最小单元格宽度设置为width）。
      2. 如果单元格的width值为auto。则最小单元格宽度设置为最小内容宽度。（如果单元格没有设置width或值为auto，把能把所有内容显示在一行的宽度设置为最小单元格宽度。）
   2. 最大宽度，要确定完全显示内容而且不包括换行符所需的宽度（<br>等强制换行除外）。（就是说把所有内容放成一行的宽度值）
2. 对于每一列，计算最小和最大列宽
   1. 列的最小宽度由该列中所有单元格的最小单元格宽度中的最大值决定。如果该列制定的width值大于列中所有单元格的最小单元格宽度，则width就是了列宽。
   2. 最大列宽的计算原理同最小列宽。

注意：这两种行为改写了html传统的表行为，对于HMTL表，会强制列扩张为与其最宽的单元格同款。

1. 如果一个单元格跨多列，最小列宽之和必须等于这个跨列单元格的最小单元格宽度。类似的，最大列宽之和必须等于跨列单元格的最大宽度。如果列宽之和与单元格宽度由差距，用户代理会把这个差距在所跨的列上平均分配。
2. 完了吗？！还没完！！如果一个列的宽度为百分比值（%），这个百分数要相对于表的宽度来计算，但是一开始浏览器并不知道这个表多宽，所以必须把这个百分数储存起来，在算法的下一步使用。

到了这一步，浏览器终于确定各列可能是多宽或者多窄。

1. 确定表的宽度
   1. 如果表的计算宽度值不是auto，将这个计算表宽度值与（所有列宽 + border + padding等）相比较（设置为百分数的值此时还要计算出来）。较大者胜出作为表的宽度。如果表的计算宽度大于（所有列宽 + border + padding等），则减去该值后再平均分配到各单元格中，让整个表格是被单元格填充满的。
   2. 如果表的计算宽度值时auto，最终宽度便为 （所有列宽 + border + padding等）。这说明表只会刚好显示其内容，而不会有多余。

通过以上麻烦的工程，终于可以确定表的布局开始渲染了。当然你以为这已经很麻烦了吗？并不是，如果表中再出现表单元素、定位元素、浮动元素，再有js去改变元素，表的计算会更加的复杂！！！

然而到此位置，还只是宽度的确定，类似的高度的确定也必须经过这样一个过程。才能最终确定表的模型。（而且，并不建议为表设置高度）。

关于vertical-align在表模型中的渲染：（都是通过补充内边距来实现）

top / bottom ：单元格内容的顶端（底端）与行顶端（底端）对齐。

middle ： 单元格内容的中间与其行中间对齐。

Baseline：单元格的基线与其行的基线对齐。

说明：行的基线由该行所有单元格中最低初始单元格基线定义。（其实就是第一行文字的基线，说白了就是一行中所有单元格的第一行文字的baseline对齐）。