# **你不知道的CSS -- 内联元素**

我们知道，在一个页面中，我们往往会用块级元素负责结构的部分，而内联元素会负责一些内容，比如图片和文字等

但是由于前端界有重 JS 轻 CSS 的风气，也由于面试很少问 CSS ，导致我们有时候对 CSS 掌握得不是很扎实

那么，基于这样的情况，我们在用css处理一些图片和文字，还原设计稿的时候，就可能会出现问题，比如产生一些 “莫名其妙” 的间隙，图片和文字 “莫名其妙” 不一起居中，以及明明设置了vertical-align却没有作用等等

为此，我写了下面这篇文章，总结了一下内联元素的相关知识，相信你看了之后会对你每天用的内联元素有新的认识

## **一.什么是内联元素**

首先我们来谈谈什么是内联元素：

### **1.1从定义看**

内联元素是display的值为inline，inline-block，inline-table的元素

### **1.2从表现看**

就表现来看，“内联元素”的典型特征就是可以和文字在一行显示。因此，文字是内联元素，图片是内联元素，按钮是内联元素，输入框，下拉框等原生表单控件也是内联元素

### **1.3常见的的内联元素有哪些**

<span>,<a>,<br>，<i>,<strong>,<em>等元素

可替换元素（<img>，<input>,<button>等）

## **二.内联盒模型简介**

下面，我想讲一下内联盒模型，这个知识点是入门css开发和熟练css开发之间的分水岭，是需要反复拿来体味的。这里介绍的”内联盒模型“是简易版，但是已经足够，如果大家对完整的概念和名词感兴趣，可以阅读规范文档。

所有的内联元素的样式表现都与行内盒子模型有关。

下面是一段和很普通的HTML：

IMG_256

看似普通，实际上包含了很多属性和概念，或者说包含了很多种盒子。我们归结为下面3个盒子：

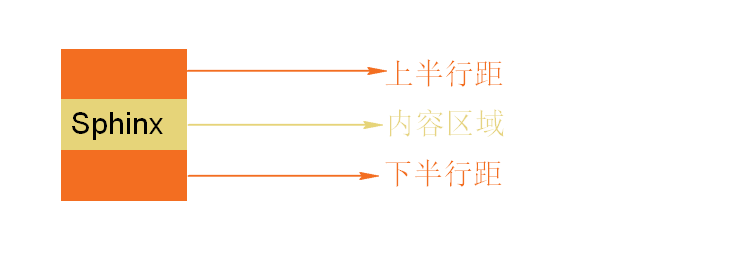
1.内联盒子（inline box），不会让内容成块显示，而是排成一排。

内联盒子类型：

如果内容被行内标签所包含（span、a、em），属于内联盒子。（上图中的实线部分）

如果只是光秃秃的文字，属于匿名内联盒子（虚线部分）

内联盒子的组成：



* 内容区域（content area）：是一种围绕文字看不见的盒子。其大小和font-size有关。

对于替换元素，内联块状元素（图片），内容区域可以看成元素本身。

* 行间距

多行文本之间相隔的距离

* 半行距

在css当中行间距被分散在文字的上方和下方，行间距的一半

2.行框盒子（line box）

每一行都是一个行框盒子，每个行框盒子都由一个一个的内联盒子组成。

IMG_258

3.包含盒子（containing box）

此盒子由一行一行的行框盒子组成

每个行框边框刚好与各行的顶边和底边刚好吻合

IMG_259

补充：

需要补充说明一点，在CSS规范中，并没有“包含盒子”的说法，更准确的称呼应该是“包含块”（containing block）。这里之所以把它称为盒子，一是为了与其他盒子名称统一，二是称为盒子更形象、更容易理解。

## **三.幽灵空白节点简介**

下面介绍内联盒模型中非常重要的一个概念 ------ ”幽灵空白节点“。这个概念很重要，会在后面的文章中多次提及。

### **3.1定义**

“幽灵空白节点”实际上也是一个盒子，不过是个假想盒，名叫“strut”，中文直译为“支柱”，是一个存在于每个“行框盒子”前面，同时具有该元素的字体和行高属性的0宽度的内联盒。

注意，这里有一个前提，文档声明必须是HTML5文档声明（HTML代码如下），如果还是很多年前的老声明，则不存在“幽灵空白节点”。

<!doctype html>

w3c规范里有这么一段：

Line boxes are created as needed to hold inline-level content within an inline formatting context. Line boxes that contain no text, no preserved white space, no inline elements with non-zero margins, padding, or borders, and no other in-flow content (such as images, inline blocks or inline tables), and do not end with a preserved newline must be treated as zero-height line boxes for the purposes of determining the positions of any elements inside of them, and must be treated as not existing for any other purpose.

意思就是如果一个line box里没有文字、保留的空格、非0的margin或padding或border的inline元素、或其他in-flow内容（比如图片、inline-block或inline-table元素），且不以保留的换行符结束的话，就会被视作高度为0的line box。

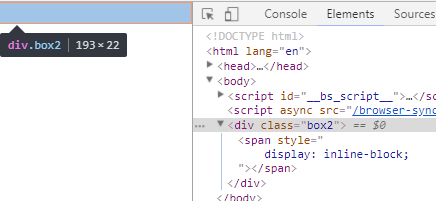
### **3.2一个简单例子**

div元素里面有一个设置了display:inline-block的span元素，没有给div设置高度，但是div却有高度。这个高度就是因为 strut 。

CSS和HTML代码如下：

<style>  
        /\* 没有设置高，却撑开了高 \*/  
        div{  
            background: #000;  
       }  
        span{  
            display: inline-block;  
       }  
</style>  
<div class="box2">  
        <span></span>  
</div>

表现：

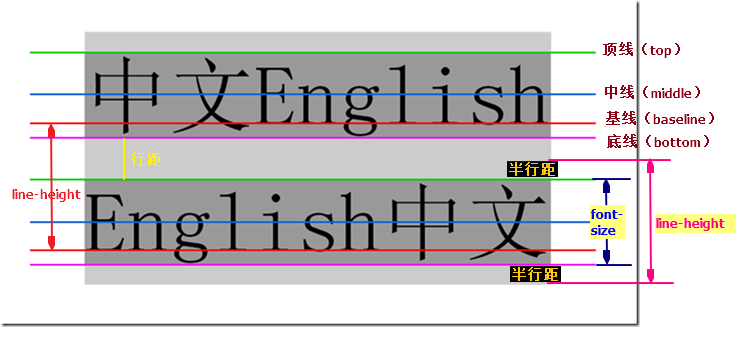


## **四.line-height**

下面讲一讲内联元素的非常主要的一个概念---line-height。

在介绍什么是Line-height之前，我想先给大家看一个图，并且图解一下行高，行距和半行距以及4条线。

### **4.1图解行高，行距和半行距和4条线**



如上图所示：如果我们仔细查看并理解，4条线:（顶线，中线，基线，底线）以及行高，行距和半行距都不用多介绍了，已然一目了然。

然后，我还想针对上面的内容补充一些东西：

**顶中基低四条线**：

vertical-align属性中有top、middle、baseline、bottom，就是和这四条线相关

基线不是最下面的线，最下面的是底线

**关于基线需要知道：**

基线其实位置是字母x的下边缘

line-height行高的定义就是两基线间的距离

vertical-align的默认值就是基线（baseline）

内联元素默认基线对齐

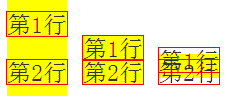
**行高，行距和半行距：**

图中两条红线之间的距离就是行高（line-height），上一行的底线和下一行的顶线之间的距离就是行距，而同一行顶线和底线之间的距离是font-size的大小，行距的一半是半行距，半行距、font-size、line-height之间的关系看图片的右下角就一目了然了

总结一下就是

半行距 = （line-height - font-size）/2

当line-height 设为2 的时候，半行距是一半的文字大小，两行文字中间的间隙差不多一个文字尺寸大小；如果line-height 大小是1倍文字大小，则根据计算，半行距是0 ，也就是两行文字会紧密依偎在一起；如果line-height 值是0.5 ，则此时的行距就是负值，虽然line-height 不支持负值，但是行距可以为负值，此时，两行文字就是重叠纠缠在一起。



看完上面的图，我们现在看看line-height究竟是什么

### **4.2line-height究竟是什么**

1.line-height是2行间基线与基线之间的距离

2.如果只有一行，那么 line-height = font-size + 行距（2个半行距）

3.line-height也是css的一个属性，用于设置行框盒子的高度

### **4.3line-height属性的作用**

既然知道了什么是line-height，我们其实更想知道的是，line-height究竟有什么作用呢？

#### **4.3.1line-height如何影响高度**

我们知道line-height这个属性是用来设置行高的，而行高也会影响高度。那么他在不同的场景下，对高度都产生什么样的一些作用呢？下面列举了行高在几种情况下对高度的影响。

几种场景下line-height的情况

1. 对于非替换的纯内联元素，其可视高度完全由line-height决定。注意，这里的完全，指的是什么padding，border属性对于可视高度是没有任何影响的。
2. 对于替换元素，比如img，line-height不能影响替换元素的高度  
   但是，在这里需要注意一点，比如下面的代码：

.box{  
 line-height:256px;  
}  
<div class="box">  
 <img src="1.jpg" height="128">  
</div>

运行之后发现：元素中，就一张图片，其他什么都没有，但此时.box 元素高度却是256px ，难道不是line-height 把图片占据的高度变高了吗？

事实上，不是的，不是line-height 把图片占据高度变高了，而是把“幽灵空白节点”的高度变高了。图片为内联元素，会构成一个“行框盒子”，而在HTML5文档模式下，每一个“行框盒子”的前面都有一个宽度为0 的“幽灵空白节点”，其内联特性表现和普通字符一模一样，所以，这里的容器高度会等于line-height 设置的属性值256px 。

1. 图片和文字混用的情况：

由于同属内联元素，因此，会共同形成一个“行框盒子”，line-height 在这个混合元素的“行框盒子”中扮演的角色是决定这个行盒的最小高度，听上去似乎有点儿尴尬，对于纯文本元素，line-height 非常威风，直接决定了最终的高度。但是，如果同时有替换元素，则line-height 的表现一下子弱了很多，只能决定最小高度，对最终的高度表现有望尘莫及之感。为什么会这样呢？一是替换元素的高度不受line-height 影响，二是vertical-align 属性在背后作祟。

1. 块级元素：

对于块级元素，line-height 对其本身是没有任何作用的，我们平时改变line-height ，块级元素的高度跟着变化实际上是通过改变块级元素里面内联级别元素占据的高度实现的。

例如：

如果我有一个div元素，他里面设置一些文字，他就有了高度，那么这个div的高度并不是由里面文字撑开的，高度和文字大小有关，实际上并不如此，实际上，这个div的高度是由里面内联元素line-height决定的。

我们可以试一下：

<!-- 设置font-size为16px，line-height为0 -->  
<div class="test1">我的高度是？</div>  
<style>  
.test1 {  
　 font-size: 16px;  
　 line-height: 0;  
　 border: 1px solid #ccc;  
　 background: #eee;  
}  
</style>

<!-- 设置line为0，font-size为16px -->  
<div class="test2">我的高度是？</div>  
<style>  
.test1 {  
　 font-size: 0;  
　 line-height: 16px;  
　 border: 1px solid #ccc;  
　 background: #eee;  
}  
</style>

结果：

IMG_263

#### **4.3.2让内联元素垂直居中**

这边列举几种情况：

1. 单行文本：

* 没有设置height，则直接设置line-height

<style>  
div{  
 line-height:24px   
}  
</style>  
<div>  
 我们是共产主义接班人！  
</div>

* 如果有height，且height大于line-height，则设置line-height 和 height一样大

<style>  
div{  
    height:30px  
 line-height:30px   // 和height大小一致   
}  
</style>  
<div>  
 我们是共产主义接班人！  
</div>

* 多行文本或者替换元素：

多行文本和替换元素（比如图片）利用line-height做垂直居中其实是一个做法

在这里，我们需要借助 line-height 的好朋友 vertical-align 去实现

比如：

<style>

.box{

line-height:120px;

background-color:#f0f3f9

}

.content{

display:inline-block;

line-height:20px;

margin:0 20px;

vertical-align: middle;

}

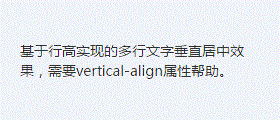
</style>

<div>

<div class="content">基于行高实现的...</div>

</div>

效果：



上面给了一些代码，也做到了多行文本的垂直居中，下面说明一下，这个效果究竟是怎么实现的：

* 多行文字使用一个标签包裹，然后设置display 为inline-block 。好处在于既能重置外部的line-height 为正常的大小，又能保持内联元素特性，从而可以设置vertical-align 属性，以及产生一个非常关键的“行框盒子”。我们需要的其实并不是这个“行框盒子”，而是每个“行框盒子”都会附带的一个产物——“幽灵空白节点”，即一个宽度为0、 表现如同普通字符的看不见的“节点”。有了这个“幽灵空白节点”，我们的line- height:120px 就有了作用的对象，从而相当于在.content 元素前面撑起了一个高度为120px 的宽度为0 的内联元素 (和上面列举的简单例子一致) 。
* 因为内联元素默认都是基线对齐的，所以我们通过对.content 元素设置 vertical- align:middle 来调整多行文本的垂直位置，从而实现我们想要的“垂直居中”效果。如果是要借助line-height 实现图片垂直居中效果，也是类似的原理和做法。

最后，我们来探究一下line-height属性的各个取值：

#### **4.3.3line-height的属性值**

line-height 可以取下面的值：

* normal 默认值
* 数值
* 百分比
* 长度值

具体解释：

normal的话，他是默认值，大概是字体大小一倍多的样子，具体大小和font-family( 字体 )有关

下面是几个桌面浏览器，在不同字体下的line-height:normal 解析值的测试数据：

| **字体** | **Chrome** | **Firefox** | **IE** |
| --- | --- | --- | --- |
| 微软雅黑 | 1.32 | 1.321 | 1.32 |
| 宋体 | 1.141 | 1.142 | 1.141 |

数值 ，如line-height:1.5 ，其最终的计算值是和当前font-size 相乘后的值。例如，假设我们此时的font-size 大小为14px ，则line-height 计算值是1.5\*14px=21px

百分比值 ，如line-height:150% ，其最终的计算值是和当前font-size 相乘后的值。例如，假设我们此时的font-size 大小为14px ，则line-height 计算值是150%\*14px=21px

长度值 ，也就是带单位的值，如line-height:21px 或者line-height:1.5em 等，此处em 是一个相对于font-size 的相对单位，因此，line-height:1.5em 最终的计算值也是和当前font-size 相乘后的值。例如，假设我们此时的font-size 大小为14px ，则line-height 计算值是1.5\*14px=21px

介绍完几个line-height的取值之后，下面有一些点需要说明一下：

1. 由于line-height的默认值是normal，也由于不同浏览器默认的中英文字体并不是一样的，操作系统的默认字体也不一样。而normal的具体数值和字体是有关的。所以，使用默认值的话，在不同系统，不同浏览器中就会有差异。因此，我们要对line-height进行重置。使用数值，百分比或者长度值。
2. 第二这里，我想说明上述使用数值，百分比和长度值的时候，他们有什么区别。

当我们设置

line-height:1.5 、line-height:150% 和line-height:1.5em

乍一看，似乎这3种用法是一模一样的，最终的行高大小都是和font-size 计算值，但是，实际上，line-height:1.5 和另外两个有一点儿不同，那就是继承细节有所差别。如果使用数值作为line-height 的属性值，那么所有的子元素继承的都是这个值；但是，如果使用百分比值或者长度值作为属性值，那么所有的子元素继承的是最终的计算值。

具体我们来看一个例子

body {

　 font-size: 14px;

　 line-height: 1.5;

}

body {

　 font-size: 14px;

　 line-height: 150%;

}

body {

　 font-size: 14px;

　 line-height: 1.5em;

}

对于<body> 元素而言，上面3段CSS最终的行高计算值是21px 是没有任何区别的，但是，如果同时还有子元素，例如：

h3, p { margin: 0; }

h3 { font-size: 32px; }

p { font-size: 20px; }

<h3>标题</h3>

<p>内容</p>

结果line-height:150% 和line-height:1.5em 的最终表现是“标题”文字和“内容”文字重叠在了一起，如图所示。

IMG_265

俗话讲“没有对比就没有伤害”，我们来看看line-height:1.5 的最终表现，排版令人舒畅，如图所示

IMG_266

我们在使用line-height:数值，并利用继承去给元素设置一致的属性和值的时候，我们需要注意一点：因为继承的优先级是最低的，又因为HTML中很多的替换元素，尤其表单类的替换元素，如输入框，按钮之类的，很多具有继承特性的CSS属性其自己也有一套，如font-family，font-size以及这里的line-height。这样，这些自身有默认样式line-height的元素，我们去设置line-height的时候，就会继承不过来。在这里我们可以这样解决：

\*{

line-height:1.5

}

也就是在通配符选择器里面去设置，这样，利用通配符的优先级比默认样式高的规则，成功给这些替换元素设置我们的line-height

第四点来谈谈具体我们在写项目中要怎么去设置我们的line-height比较合适

如果我们做的是一个重图文内容展示的网页或者网站，如博客、论坛、公众号之类的，一般使用数值，我们可以在html使用

/\* 值一般设置为1.6～1.8 \*/

html{

line-height:1.6

}

如果是一个偏重布局结构精致的网站，使用长度值或者数值都是可以的。因为网站内容的样式不是动态不可控的，有时候，固定的长度值反而更利于精确布局

/\* 通过继承设置数值line-height \*/

html{

line-height:1.5

}

/\* 在某些部分使用长度值 \*/

.spe{

line-height:20px

}

一个参考：

基本上各大站点都是使用数值作为全局的line-height值，一般为1.3, 1.4, 1.5

使用line-height:20px和line-height:1.5，排版的时候会比较好计算

## **五.vertical-align**

接下来讲一个可以说css里面最复杂的属性 -- vertical-align

首先我们来看看MDN对vertical-align的介绍：

CSS的属性 vertical-align 用来指定行内元素（inline）或表格单元格（table-cell）元素的垂直对齐方式

这句话其实包含了2个信息点：

1. vertical-align作用于行内（inline）元素或表格单元格元素（table-cell）
2. vertical-align是用来指定垂直对齐方式的

下面对第一个信息点展开说明一下：

### **5.1vertical-align作用的前提**

有的同学，不了解vertical-align作用的前提，对块级元素设置了vertical-align:middle，想要让他垂直居中，但是发现没有效果

原因在于：

vertical-align作用的前提是：只能应用于内联元素以及display值为table-cell的元素

换句话说，vertical-align 属性只能作用在display 计算值为inline 、inline- block ，inline-table 或table-cell 的元素上。因此，默认情况下，<span> 、< strong> 、<em> 等内联元素，<img> 、<button> 、<input> 等替换元素，非HTML规范的自定义标签元素，以及<td> 单元格，都是支持vertical-align 属性的，其他块级元素则不支持。

但是下面几点是需要注意的：

* 浮动和绝对定位元素会让元素块状化，而vertical-align对于块级元素无效

因此，下面的代码组合是无效的：

.example1{

float:left;

vertical-align:middle

}

.example2{

position:absolute;

vertical-align:middle

}

* 使用vertical-align:middle的时候，是相对于行框盒子去居中。因此下面的代码没有实现想要的居中：

/\* 在这里，我们想要img相对于外面的box去居中，但是我们这里使用vertical-align:middle并没有实现我们想要的效果\*/

.box{

height:128px;

}

.box > img{

height:96px;

vertical-align:middle

}

失效的原因是，他外面包含盒子的高是128px，而里面行框盒子的高是由行框黑里面的内联盒子img决定的，是96px。因此，vertical-align:middle实际上是生效了的，但是他居中是相对于行框盒子去居中了，也就是96px的图片相对于96px的行框盒子去居中，所以看起来没什么效果。

基于上面失效的原因，我们如果想要实现效果，我们可以：

/\* 在这里，我们给外面的box元素设置line-height=height，这样行高够大，而行框盒子前面的幽灵空白节点高度也随之变大，把行框盒子变高，这样vertical-align:middle就达到我们期望的效果了\*/

.box{

height:128px;

line-height:128px; /\*关键的css属性\*/

}

.box > img{

height:96px;

vertical-align:middle

}

* 设置display:table-cell的元素：

在这里，我们要明确2点：

* 对table-cell元素而言，vertical-align起作用的是table-cell元素自身
* 就效果而言，table-cell 元素设置vertical-align 垂直对齐的是子元素，但是其作用的并不是子元素，而是table-cell 元素自身。就算table-cell 元素的子元素是一个块级元素，也一样可以让其有各种垂直对齐表现。

### **5.2vertical-align，line-height与幽灵空白节点**

这3者经常会一起出现，弄出一些奇奇怪怪的间隙或者有着奇怪的表现。下面列举了几个奇怪场景：

奇怪场景一

行框盒子里面，我的匿名内联盒子（文字）和我的内联盒子的font-size设置得不一致，出现了超出我设定行高的现象：

具体来看一个例子：

<style>

.box { line-height: 32px; }

.box > span { font-size: 24px; }

</style>

<div class="box">

　 <span>文字</span>

</div>



这里我们可以看到，div的高度是36px，比给他的行高32px还要多4px，那么这是为什么呢？

这是由于行框盒子里面的内联盒子字体大小不一致，外加vertical-align默认baseline导致的。具体点说就是：

我们行框盒子前面都有幽灵空白节点，由于这个幽灵空白节点看不见不好解释，我们换成一个x字符，效果一样，所以这里我们在行框盒子前面加一个x字符，同时我们在”文字"后面也加上一个x字符（因为所有字符里面，x的下端就是基线所在地方,方便看后面文字所在基线）。

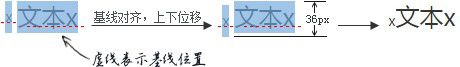
于是我们便有：

<div class="box">

　 x<span>文字x</span>

</div>

此时，我们可以明显看到两处大小完全不同的文字。一处是字母x构成了一个“匿名内联盒子”，另一处是“文字 x”所在的<span> 元素，构成了一个“内联盒子”。由于都受line- height:32px 影响，因此，这两个“内联盒子”的高度都是32px 。下面关键的来了，对字符而言，font-size 越大字符的基线位置越往下，因为文字默认全部都是基线对齐，所以当字号大小不一样的两个文字在一起的时候，彼此就会发生上下位移，如果位移距离足够大，就会超过行高的限制，而导致出现意料之外的高度，如图所示：



解决方法：

既然行高变大产生原因是行框盒子里面的内联盒子字体大小不一致，外加vertical-align默认baseline导致的，那么，我们就有一些对应的解决方法

首先，我们可以让“幽灵空白节点”和后面<span>元素的字号一样大，也就是

<style>

.box {

line-height: 32px;

font-size:24px

}

.box > span { font-size: 24px; }

</style>

其次，我们也可以修改vertical-align的值，让他不等于默认值baseline，比如让他顶部对齐，这样就不会有位移了

<style>

.box {

line-height: 32px;

}

.box > span {

font-size: 24px;

vertical-align:top;

}

</style>

看完了场景一，接下来我们来看看奇怪场景二：

现象是这样的：任意一个块级元素，里面若有图片，则块级元素高度基本上都要比图片的高度高。例如：

<style>

.box {

　 width: 280px;

　 outline: 1px solid #aaa;

　 text-align: center;

}

.box > img {

　 height: 96px;

}

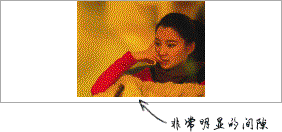
</style>

<div class="box">

　 <img src="1.jpg">

</div>

效果：



产生的原因还是因为幽灵空白节点。。在这里为了方便演示，我们用字符x替代幽灵空白节点，然后再看看：



可以明显看到，x的下边缘还有一些空间，我们在上面讲了，其实这个空间是半行距。那么这里半行距是怎么来的呢？

这是因为因为默认的font-size是14px，默认的line-height是normal（1点几倍），然后line-height就会比font-size多几个px，半行距 = (行高 - font-size)/2，那么半行距是有值的，这里的空间也是确实存在的。

解决方法：

知道了间隙产生的原因，下面来讲讲怎么处理，这里处理的方法有很多：

1. 图片块状化 。即直接设置display:block

都变成块状了，前面讲的什么line-height，幽灵空白节点，vertical-align就没关系了

1. 容器 line-height 设置为0

只要半行间距小到字母x的下边缘位置或者再往上，自然就没有了撑开底部间隙高度空间了

1. 容器 font-size 设置为0
2. 图片设置其他 vertical-align 属性值   
    间隙的产生原因之一就是基线对齐，所以我们设置vertical-align 的值为top、middle、bottom 中的任意一个都是可以的。

讲最后一个奇怪场景，场景三：

内敛特性导致margin无效

代码如下：

<style>.box > img {

　 height: 96px;

　 margin-top: -200px;

}</style>

<div class="box">

　 <img src="mm1.jpg">

</div>

此时，按照理解，-200px 远远超过图片的高度，图片应该完全跑到容器的外面，但是，图片依然有部分在.box 元素中，而且就算margin-top 设置成-99999px ，图片也不会继续往上移动，完全失效。其原理和上面图片底部留有间隙实际上是一样的，图片的前面有个“幽灵空白节点”，而在CSS世界中，非主动触发位移的内联元素是不可能跑到计算容器外面的，导致图片的位置被“幽灵空白节点”的vertical-align:baseline 给限死了。我们不妨把看不见的“幽灵空白节点”使用字符x代替，原因就一目了然了，如图所示：



因为字符x下边缘和图片下边缘对齐，字符x非主动定位，不可能跑到容器外面，所以图片就被限死在此问题，margin-top 失效。

最后来列举并说明一下vertical-align的属性值

### **5.3vertical-align的属性值**

vertical-align 属性值分为以下4类：

1. 数值百分比类 ，如20px 、2em 、20% 等
2. 线性类 ，如baseline （默认值）、top 、middle 、bottom
3. 文本类 ，如text-top 、text-bottom
4. 上标下标类 ，如sub 、super

这里我讲着重讲一下前面2类，因为后面2种（文本类和上标下标类）基本不用

#### **5.3.1百分和数值类**

具体的数值和百分比表现是一致的：根据计算值，相对于基线往上或者往下偏移，如果是正值往上偏移，如果是负值，往下偏移。

先讲一下百分比

vertical-align的百分比是相对于line-height计算的

事实上，百分比也是用得很少的，因为在现在网页布局里面，line-height的计算值都是相对固定且已知的，因此，直接使用具体的数值会更方便。所以知道上面那一点就可以了

接着讲一下具体的数值：

比如veticall-align:-5px这种，其实是很好用的

场景一：

首先我们有

* 上面我们提到那四条线（顶中基低）的时候有提到，基线其实所在的地方是字母x的下边缘
* 对于图片等替换元素，会使用元素本身的下边缘作为基线
* 中文和部分英文的下边缘会低于字母x的下边缘

基于上面三点，我们在文字和图片一起使用的时候会特别明显会感觉文字是明显偏下的，如下图：



在这里我们就可以给文字内容使用vertical-align:10px，让他往上移



场景二：

假设有一个display 值为inline-block 的尺寸为20像素×20像素的小图标，默认状态下，文字是明显偏下的，类似下图中“请选择”三个字和后面三角图形的位置关系。

IMG_274

这时候，相比于用什么vertical-align:middle甚至用margin，relative等，其实我们更适合使用vertical-align:-5px这样的方法去达到我们的效果

IMG_275

#### **5.3.2线性类属性值**

* baseline

vertical-align的默认值，在文本里面就是之前4条线图解的基线，对于替换元素则是替换元素的 下边缘

但是这里有一个点需要注意，如果是inline-block 元素，则规则要复杂一些：

一个 inline-block 元素，如果里面没有内联元素，或者 overflow 不是 visible ，则该元素的 基线就是其 margin 底边缘；否则其基线就是元素里面最后一行内联元素的基线

例子代码如下

<style>

.dib-baseline {

display: inline-block;

width: 150px; height: 150px;

border: 1px solid #cad5eb;

background-color: #f0f3f9;

}

</style>

<span class="dib-baseline"></span>

<span class="dib-baseline">x-baseline</span>

效果如下：



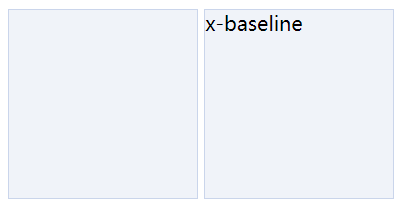
在这里，我们解决的方案是，给第二个盒子修改他的垂直对齐方式，比如设置顶端对齐，让他不 再是基线对齐（即不管他里面是否有内联元素，我都让他的顶端与行框顶端对齐）

span:nth-child(2){

vertical-align:top

}

效果如下：



* top/bottom

vertial-align:top 和vertial-align:bottom 基本表现类似，只是一个“上”一个“下”，因此合在 一起讲。

顾名思义，vertial-align:top 就是垂直上边缘对齐，具体定义如下。

内联元素 ：元素底部和当前行框盒子的顶部对齐。

table-cell 元素 ：元素底padding 边缘和表格行的顶部对齐。

用更通俗的话解释就是：如果是内联元素，则和这一行位置最高的内联元素的顶部对齐；如果 display 计算值是table-cell 的元素，我们不妨脑补成<td> 元素，则和<tr> 元素上边缘对齐。

vertial-align:bottom 声明与此类似，只是把“顶部”换成“底部”，把“上边缘”换成“下边缘”。

需要注意的是，内联元素的上下边缘对齐的这个“边缘”是当前“行框盒子”的上下边缘，并不是块 状容器的上下边缘。

* middle

我们经常使用这个属性值去做居中，事实上，他也非常好用。但是，在这里我想说明一点：

对于内联元素，他会使得使用该属性值的元素和行框盒子基线往上 1/2个x字符高度 去对齐

然后在各个字体中，x的位置都是稍微偏下一点的

因此，这里的居中是近似居中，可能会有一点点的偏差

## **总结**

通过上面的阅读，你可以尝试以下回答下面几个问题（如果不知道，请重新阅读本文）：

1. 什么是内联元素？
2. 什么是内联盒模型？
3. 什么是幽灵空白节点？
4. 有哪4条线？行距和半行距是什么以及和行高有什么关系？
5. line-height：

是什么？

他有什么作用：

在不同情况下是如何对高度产生作用的？

如何使用line-height做一个内联元素的垂直居中？

line-height的属性值有哪些？以及有哪些需要注意的地方？

1. vertical-align：

是什么？

他作用的前提是什么？

和行高以及strut一起作用时，有哪几种奇怪的场景，怎么解决？

重点属性值是什么，需要注意哪些东西？

## **参考资料**

《CSS世界》

《CSS权威指南》

[CSS幽灵空白节点](https://www.lirls.com/post/8d6a/)

[CSS中的line-height](https://segmentfault.com/a/1190000003038583)