

「学习笔记」CSS 基础

原创 饭老板 前端 fan

收录于话题

#前端入门 4

#CSS 基础 2



「学习笔记」CSS 基础

前言

拖延了一周的 CSS 学习笔记终于利用周末去补齐了，本篇文章着重梳理之前所学的 CSS 知识点，查漏补缺。同时，试着用 git 将重点案例存放到远程仓库中，更近一步贴近公司流程。□□

CSS 构造块

「1. HTML 的局限性」

HTML 满足不了设计者的需求，可以将网页结构与样式相分离，这样就可以在不更改网页结构的前提下，更换网站的样式。

操作 html 属性不方便

HTML 里面添加样式带来的是无尽的臃肿和繁琐

「2. CSS 网页的美容师」

让我们的网页更加丰富多彩，布局更加灵活自如。

CSS 最大的贡献：让 HTML 从样式中脱离，实现了 HTML 专注去做结构呈现，样式交给 CSS

「3. CSS」CSS(Cascading Style Sheets)通常称为 CSS 样式表或层叠样式表(级联样式表)。

作用

主要用于设置 HTML 页面中的文本内容(字体、大小、对齐方式等)\图片的外形(宽高、边框样式、边距等)以及版面的布局和外观显示样式。

CSS 以 HTML 为基础，提供了丰富的功能，如字体、样式、背景的控制及整体排版等，而且可以针对不同的浏览器设置不同的样式。

「4. CSS 注释」

```
/* 这是注释 */
```

引入 CSS 样式表

「1.行内式(内联样式)」

通过标签的 style 属性来设置元素的样式

style 其实就是标签的属性

样式属性和值中间是:

多组属性值直接用;隔开

只能控制当前的标签和以及嵌套在其中的子标签，造成代码冗余。

缺点:没有实现样式和结构相分离。

```
<标签名 style="属性 1:属性值 1; 属性 2:属性值 2; 属性 3:属性值 3;"> 内容 </标签名>
```

例如：

```
<div style="color: red; font-size: 12px;">青春不常在，抓紧谈恋爱</div>
```

「2.内部样式表(内嵌样式表)」

也称为内嵌式，将 CSS 代码集中写在 HTML 文档的 head 头部标签中，并且用 style 标签定义。

style 标签一般位于 head 标签中，当然理论上他可以放在 HTML 文档的任何地方。

type="text/css" 在 html5 中可以省略。

只能控制当前的页面

缺点:没有彻底分离结构与样式

```
<head>
<style type="text/CSS">
    选择器 ( 选择的标签 ) {
        属性 1: 属性值 1;
        属性 2: 属性值 2;
        属性 3: 属性值 3;
    }
</style>
</head>
```

「3.外部样式表(外链式)」

也称链入式，是将所有的样式放在一个或多个以 .css 为扩展名的外部样式表文件中，通过 link 标签将外部样式表文件链接到 HTML 文档中。

rel:定义当前文档与被链接文档之间的关系，在这里需要指定为 "stylesheet"，表示被链接的文档是一个样式表文件。

href:定义所链接外部样式表文件的 URL，可以是相对路径，也可以是绝对路径。

```
<link rel="stylesheet" href="index.css">
```

「4.团队约定-代码风格」

```
/*1.紧凑格式 (Compact)*/
h3 { color: deeppink;font-size: 20px;}
// 2.一种是展开格式 ( 推荐 )
h3 {
    color: deeppink;
    font-size: 20px;
}

/* 团队约定-代码大小写 */
```

```
/* 样式选择器，属性名，属性值关键字全部使用小写字母书写，属性字符串  
允许使用大小写。*/  
/* 推荐 */  
h3{  
  color: pink;  
}  
  
/* 不推荐 */  
H3{  
  COLOR: PINK;  
}
```

CSS 基础选择器

CSS 选择器作用

找到指定的 HTML 页面元素，选择标签。

CSS 基础选择器

「1. 标签选择器」

标签选择器（元素选择器）是指用 **HTML 标签名称** 作为选择器，按标签名称分类，为页面中某一类标签指定统一的 CSS 样式。

作用：可以把某一类标签 **全部** 选择出来。

优点：快速为网页中同类型的标签统一样式

缺点：不能设计差异化样式。

标签名 { 属性 1: 属性值 1; 属性 2: 属性值 2; 属性 3: 属性值 3; }

「2. 类选择器」

类选择器使用 "." (英文点号) 进行标识，后面紧跟 **类名**。

语法：**类名 选择器**

```
.类名 {  
    属性 1:属性值 1;  
    属性 2:属性值 2;  
    属性 3:属性值 3;  
}
```

```
<p class='类名'></p>
```

优点：可以为元素对象定义单独或相同的样式。可以选择一个或者多个标签。

注意：类选择器使用“.”（英文点号）进行标识，后面紧跟类名(自定义，我们自己命名的)

长名称或词组可以使用**中横线**来为选择器命名。

不要纯数字、中文等命名，尽量使用英文字母来表示。

多类名选择器：各个类名中间用空格隔开。

「3. id 选择器」id 选择器使用#进行标识，后面紧跟 id 名

元素的 id 值是唯一的，只能对应于文档中某一个具体的元素。

```
#id 名 {属性 1:属性值 1; 属性 2:属性值 2; 属性 3:属性值 3; }
```

```
<p id="id 名"></p>
```

「4. 通配符选择器」

通配符选择器用*号表示，* 就是选择所有的标签。它是所有选择器中**作用范围最广**的，能匹配页面中所有的元素。

注意：会匹配页面所有的元素，降低页面响应速度，不建议随便使用

```
* { 属性 1:属性值 1; 属性 2:属性值 2; 属性 3:属性值 3; }
```

例如下面代码，使用通配符选择器定义 CSS 样式，清除所有 HTML 标记的默认边距。

```
* {  
    margin: 0;           /* 定义外边距*/  
    padding: 0;          /* 定义内边距*/  
}
```

「5. 基础选择器总结」

选择器作用缺点使用情况用法 标签选择器可以选出所有相同的标

签，比如 p 不能差异化选择较多 p { color: red;}类选择器可以选出 1 个或者多个标签可以根据需求选择非常多.nav { color: red;}id 选择器一次只能选择器 1 个标签只能使用一次不推荐使用 #nav {color: red;}通配符选择器选择所有的标签选择的太多，有部分不需要不推荐使用 * {color: red;}

「6. 团队约定-选择器」

尽量少用通配符选择器 *。

尽量少用 ID 选择器

不使用无具体语义定义的标签选择器。

/* 推荐 */

.jdc {}

li {}

p {}

/* 不推荐 */

* {}

#jdc {}

div {} 因为 div 没有语义，我们尽量少用

CSS 复合选择器

复合选择器是由两个或多个基础选择器，通过不同的方式组合而成的

「1. 后代选择器」又称为包含选择器

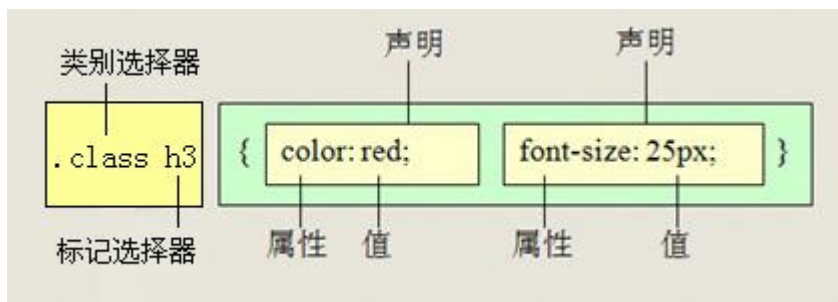
用来选择元素或元素组的子孙后代

其写法就是把外层标签写在前面，内层标签写在后面，中间用「空格」分隔，先写父亲爷爷，再写儿子孙子。

子孙后代都可以这么选择。或者说，它能选择任何包含在内的标签。

父级 子级 {属性:属性值;属性:属性值;}

.class h3 {color:red;font-size:16px;}



当标签发生嵌套时，内层标签就成为外层标签的后代。

子孙后代都可以这么选择。或者说，它能选择任何包含在内的标签。

「2. 子元素选择器」

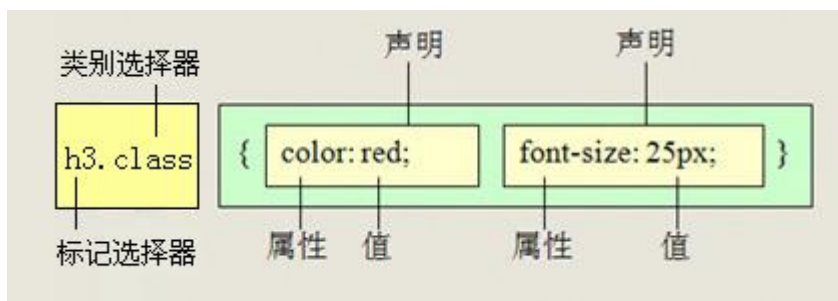
子元素选择器只能选择作为某元素子元素(亲儿子)的元素。

其写法就是把父级标签写在前面，子级标签写在后面，中间跟一个 > 进行连接

这里的子,指的是亲儿子。不包含孙子 重孙子之类。

```
.class>h3 {color:red;font-size:14px;}
```

「3. 交集选择器」



其中第一个为标签选择器，第二个为 class 选择器，两个选择器之间不能有空格，如 h3.special。

交集选择器是并且的意思,即...又...的意思

比如：`p.one` 选择的是：类名为 `.one` 的段落标签。

*/*用的相对来说比较少，不建议使用。*/*

「4. 并集选择器」如果某些选择器定义的相同样式，就可以利用并集选择器，可以让代码更简洁。并集选择器（CSS 选择器分组）是各个选择器通过,连接而成的，通常用于集体声明。

任何形式的选择器（包括标签选择器、class 类选择器 id 选择器

等)，都可以作为并集选择器的一部分。

并集选择器通常用于**集体声明**，逗号隔开的，所有选择器都会执行后面样式，逗号可以理解为**和**的意思。

比如

```
.one,  
p,  
#test {color: #F00;}
```

表示 `.one` 和 `p` 和 `#test` 这三个选择器都会执行颜色为红色。
通常用于集体声明。

「5. 链接伪类选择器」

用于向某些选择器添加特殊的效果。写的时候，他们的顺序尽量不要颠倒,按照 **lvha** 的顺序。否则可能引起错误。

链接伪类，是利用交集选择器。

a:link 未访问的链接

a:visited 已访问的链接

a:hover 鼠标移动到链接上

a:active 选定的链接

实际工作中，很少写全四个状态，一般写法如下：

```
a { /* a 是标签选择器 所有的链接 */  
  font-weight: 700;  
  font-size: 16px;  
  color: gray;  
  text-decoration: none; /* 清除链接默认的下划线 */  
}  
a:hover { /* :hover 是链接伪类选择器 鼠标经过 */  
  color: red; /* 鼠标经过的时候，由原来的 灰色 变成了红色 */  
}
```

「6. 复合选择器总结」

选择器作用特征使用情况
符号及用法
后代选择器用来选择元素
后代是选择所有的子孙
后代较多符号是空格 `.nav a`
子代选择器选择最近一级元素
只选亲儿子较少符号是 `>` `.nav>p`
交集选择器选择两个标签交集的部分
既是 又是较少没有符号 `p.one`
并集选择器

选择某些相同样式的选择器可以用于集体声明较多符号是逗号 .nav, .header 链接伪类选择器给链接更改状态较多重点记住 a{} 和 a:hover 实际开发的写法

CSS 字体样式

font 字体

[1. font-size]

font-size 属性用于设置字号(字体大小)

谷歌浏览器默认的文字大小为 16px

不同浏览器可能默认显示的字号大小不一致，我们尽量给一个明确值大小，不要默认大小。一般给 body 指定整个页面文字的大小。

```
p { font-size:20px; }
```

单位

相对长度单位、绝对长度单位

相对长度单位	说明
em	相对于当前对象内文本的字体尺寸
px	像素，最常用，推荐使用
绝对长度单位	说明
in	英寸
cm	厘米
mm	毫米
pt	点

[2. font-family]

`font-family` 属性用于设置哪一种字体。

```
p { font-family:"微软雅黑";}
```

指定多个字体，如果浏览器不支持第一个字体就会尝试下一个直到找到合适的字体，如果都没有，以电脑默认字体为准。

```
p {font-family: Arial,"Microsoft Yahei", "微软雅黑";}
```

CSS Unicode 字体

在 CSS 中设置字体名称，直接写中文是可以的。但是在文件编码（GB2312、UTF-8 等）不匹配时会产生乱码的错误。

xp 系统不支持 类似微软雅黑的中文。

解决方案：英文来替代。比如 `font-family:"Microsoft Yahei"`。在 CSS 直接使用 Unicode 编码来写字体名称可以避免这些错误。使用 Unicode 写中文字体名称，浏览器是可以正确的解析的。

`font-family: "\5FAE\8F6F\96C5\9ED1";` 表示设置字体为“微软雅黑”。

[3. font-weight]

属性值描述 `normal` 默认值（不加粗的）`bold` 定义粗体（加粗的）
100~900 400 等同于 `normal`，而 700 等同于 `bold`（数字表示粗细用的多）

[4. font-weight]

`font-style` 属性用于定义字体风格，如设置斜体、倾斜或正常字体，其可用属性值如下：

属性作用 `normal` 默认值，浏览器会显示标准的字体样式 `font-style: normal;italic` 浏览器会显示斜体的字体样式。

[5. font:综合设置字体样式]

```
选择器 { font: font-style font-weight font-size/line-height font-family;}
```

注意：使用 `font` 属性时，必须按上面语法格式中的顺序书写，不能更换顺序，各个属性以空格隔开

其中不需要设置的属性可以省略(取默认值),但必须保留 `font-size` 和 `font-family` 属性，否则 `font` 属性将不起作用。

[6. font 总结]

属性表示注意点 font-size 字号我们通常用的单位是 px 像素，一定要跟上单位 font-family 字体实际工作中按照团队约定来写字体 font-weight 字体粗细记住加粗是 bold 或者 bold 不加粗是 normal 或者 400 记住数字不要跟单位 font-style 字体样式记住倾斜是 italic 不倾斜是 normal 工作中我们最常用 normal 字体连写 1. 字体连写是有顺序的 不能随意换位置 2. 其中字号 和 字体 必须同时出现

CSS 外观属性

[1. color]

color 属性用于定义文本的颜色

其取值方式有以下 3 种：

实际工作中，用 16 进制的写法是最多的，且我们更喜欢简写方式比如 #f0 代表红色。

表示表示属性值预定义的颜色值 red, green, blue, pink 十六进制 #FF0000, #FF6600, #29D794 RGB 代码 rgb(255,0,0) 或 rgb(100%,0%,0%)

[2. text-align]

text-align 属性用于设置文本内容的水平对齐方式，相当于 html 中的 align 对齐属性。

注意：是让盒子里面的文本内容水平居中，而不是让盒子居中对齐

其可用属性值如下：

属性解释 left 左对齐（默认值） right 右对齐 center 居中对齐

[3. line-height] line-height 属性用于设置行间距，就是行与行之间的距离，即字符的垂直间距，一般称为行高。

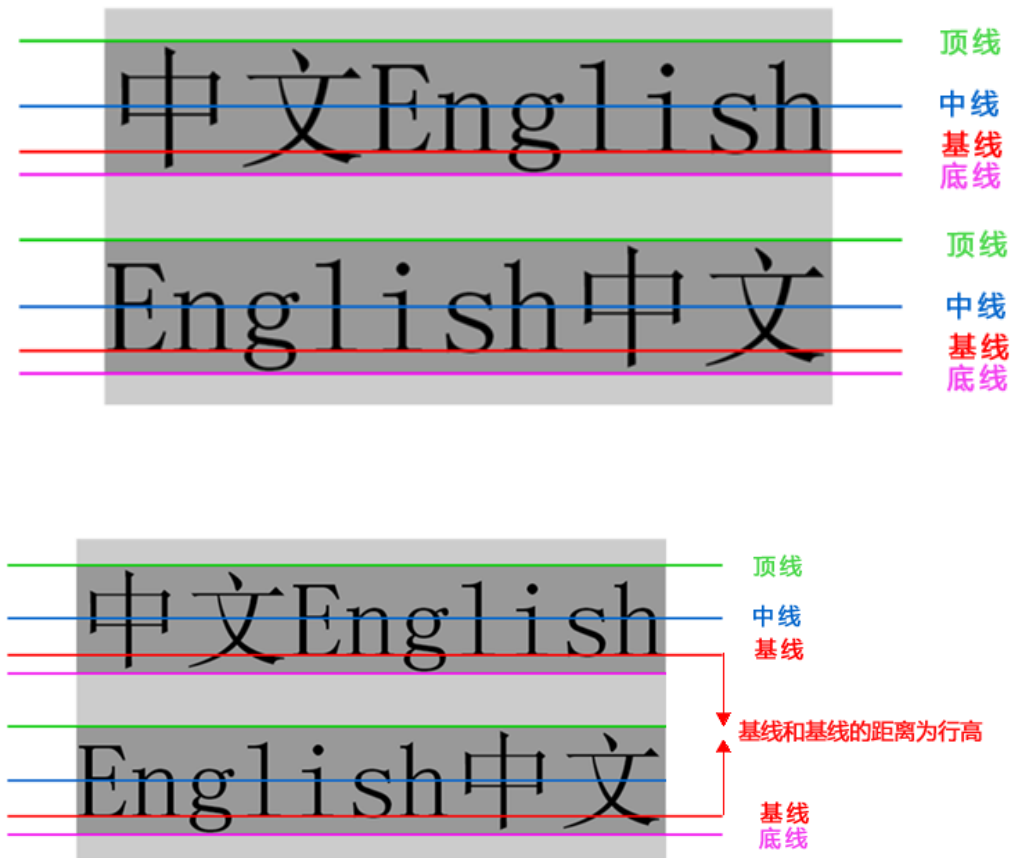
line-height 常用的属性值单位有三种，分别为像素 px，相对值 em 和百分比%，实际工作中使用最多的是像素 px

一般情况下，行距比字号大 7--8 像素左右就可以了。

line-height: 24px;

行高测量

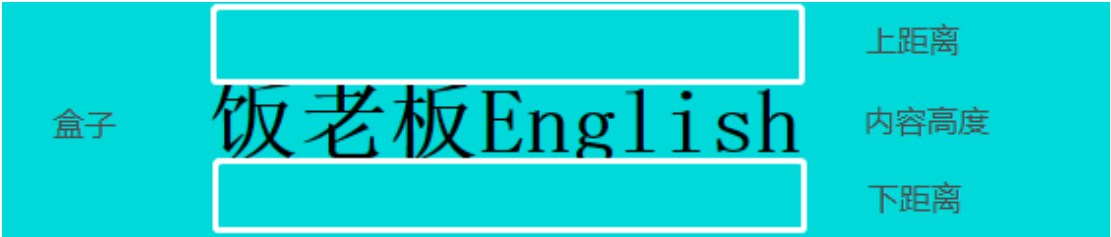
行高测量方法：



行高测量方法行高我们利用最多的一个地方是：可以让单行文本在盒子中垂直居中对齐。

文字的行高等于盒子的高度。行高 = 上距离 + 内容高度 + 下距离

上距离和下距离总是相等的，因此文字看上去是垂直居中的。



行高与高度的三种关系

如果 行高 等 高度 文字会 垂直居中

如果行高 大于 高度 文字会 偏下

如果行高小于高度 文字会 偏上

```
/*line-height 要设置在 font 属性下面，否则无效，例如：*/  
height: 80px;  
text-align: center;  
font: normal bold 30px "宋体";  
line-height: 80px;
```

可以使用 display:flex;布局方式让文字水平垂直居中

```
display: flex;  
align-items: center; /* 侧轴对齐方式 */  
justify-content: center; /* 主轴对齐方式 */
```

[4. text-indent]

text-indent 属性用于设置首行文本的缩进

其属性值可为不同单位的数值、em 字符宽度的倍数、或相对于浏览器窗口宽度的百分比%，允许使用负值。

建议使用 em 作为设置单位。

1em 就是一个字的宽度。如果是汉字的段落，1em 就是一个汉字的宽度

```
p {  
    /*行间距*/  
    line-height: 25px;  
    /*首行缩进 2 个字 em 1 个 em 就是 1 个字的大小*/  
    text-indent: 2em;  
}
```

[5. text-decoration] 文本的装饰

text-decoration,通常我们用于给链接修改装饰效果

值描述 none 默认。定义标准的文本。取消下划线（最常用）

underline 定义文本下的一条线。下划线 也是我们链接自带的（常用）

overline 定义文本上的一条线。（不用）line-through 定义穿

过文本下的一条线。(不常用)

「6. CSS 外观属性总结」

属性表示注意点 color 颜色我们通常用 十六进制 比如 而且是简写形式 #fffline-height 行高控制行与行之间的距离 text-align 水平对齐可以设定文字水平的对齐方式 text-indent 首行缩进通常我们用于段落首行缩进 2 个字的距离 text-indent: 2em;text-decoration 文本修饰记住 添加 下划线 underline 取消下划线 none

标签显示模式(display)

标签显示模式是标签以什么方式进行显示。HTML 标签一般分为块标签和行内标签两种类型，它们也称为块元素和行内元素。

标签显示模式转换 display

块转行内：`display:inline;`

行内转块：`display:block;`

块、行内元素转换为行内块：`display: inline-block;`

「1. 块级元素(block-level)」

常见的块元素有<h1>~<h6>、<p>、<div>、、、等，其中<div>标签是最典型的块元素。

块级元素的特点

独占一行

高度，宽度，外边距以及内边距都可以控制。

宽度默认是容器(父级宽度)的 100%

是一个容器及盒子，里面可以放行内或者块级元素

注意：只有文字才能组成段落，因此 p 标签里面不能放块级元素，特别是 p 不能放 div。同理，还有 h1~h6，dt,它们都是文字类块级标签，里面不能放其他块级元素。

「2. 行内元素(inline-level)」

有的地方也称为内联元素

常见的行内元素有 <a>、、、、<i>、、<s>、<ins>、<u>、等，其中标签最典型的行内元素。

行内元素的特点

相邻行内元素在一行上，一行可以显示多个。

高度、宽度直接设置是无效的。

默认高度就是它本身内容的宽度。

行内元素只能容纳文本或其他行内元素。

注意

链接里面不能再放链接

特殊情况 a 里面可以放块级元素，但是给 a 转换一下块级模式最安全。

「3. 行内块元素(inline-block)」

在行内元素中有几个特殊的标签——、<input>、<td>，可以对它们设置宽高和对齐属性，有些资料可能会称它们为行内块元素。

行内块元素的特点

和相邻行内元素(行内块)在一行上，但是之间会有空白风险。一行可以显示多个

默认宽度就是它本身内容的宽度。

高度，行高，外边距以及内边距都可以控制。

三种模式总结

元素模式 元素排列 设置样式 默认宽度 包含块级元素 一行只能放一个
块级元素 可以设置宽度 高度 容器的 100% 容器级 可以包含任何标签
行内元素 一行可以放多个 行内元素 不可以直接设置宽度 高度 它本身
内容的宽度 容纳文本 或则其他行内元素 行内块元素 一行放多个 行内
块元素 可以设置宽度和高度 它本身内容的宽度

CSS 背景(background)

「1. 背景颜色」

`background-color`: 颜色值; 默认的值是 `transparent` 透明的

「2. 背景图片(image)」

语法:

`background-image`: `none` | `url(url)`;

例如:

`background-image: url(images/1.png);`

「3. 背景平铺 (repeat)」

`background-repeat`: `repeat` | `no-repeat` | `repeat-x` | `repeat-y`

参数作用 `repeat` 背景图像在纵向和横向上平铺 (默认的) `no-repeat` 背景图像不平铺 `repeat-x` 背景图像在横向上平铺 `repeat-y` 背景图像在纵向平铺

「4. 背景位置(position)」

`background-position`: `length` || `length`

`background-position`: `position` || `position`

参数值 `length` 百分数 | 由浮点数字和单位标识符组成的长度值

`position` `top` | `center` | `bottom` | `left` | `center` | `right` 方位名词

注意：

必须先指定 background-image 属性

position 后面是 x 坐标和 y 坐标。可以使用方位名词或者 精确单位。

如果指定两个值，两个值都是方位名字，则两个值前后顺序无关，比如 left top 和 top left 效果一致

如果只指定了一个方位名词，另一个值默认居中对齐。

如果 position 后面是精确坐标，那么第一个，肯定是 x 第二个一定是 y

如果只指定一个数值，那该数值一定是 x 坐标，另一个默认垂直居中

如果指定的两个值是 精确单位和方位名字混合使用，则第一个值是 x 坐标，第二个值是 y 坐标

背景简写：

background：属性的值的书写顺序官方没有强制的标准。为了可读性，建议如下写：

background: 背景颜色 背景图片地址 背景平铺 背景滚动 背景位置；

/ 有背景图片背景颜色可以不用写 */*

background: transparent url(image.jpg) repeat-y scroll center top；

「5. 背景半透明(CSS3)」

background: rgba(0, 0, 0, 0.3);

background: rgba(0, 0, 0, .3);

等同于 background-color: rgba(0, 0, 0, .3)

最后一个参数是 alpha 透明度 取值范围 0~1 之间

我们习惯把 0.3 的 0 省略掉 这样写 background: rgba(0, 0, 0, .3);

注意：背景半透明是指 盒子背景半透明，盒子里面的内容不受影响
低于 IE 9 的版本不支持

盒子半透明 opacity

设置 opacity 元素的所有后代元素会随着一起具有透明性，一般用于调整图片或者模块的整体不透明度

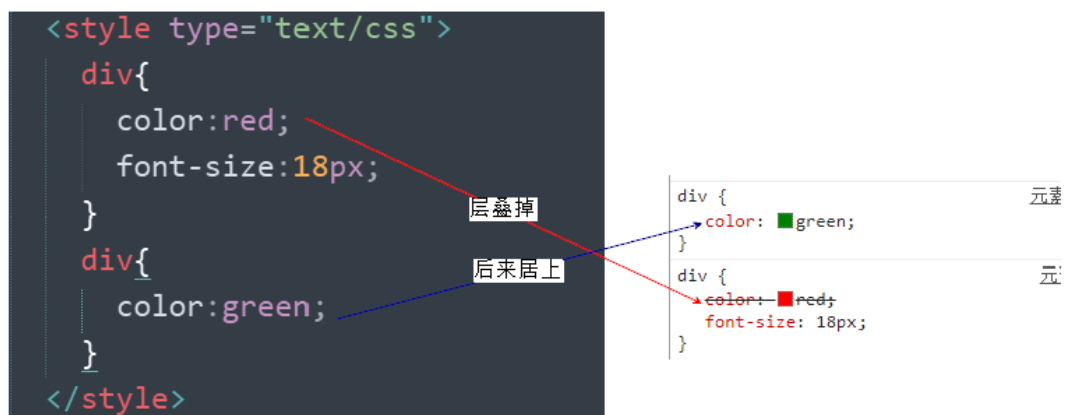
opacity: .2;

「6. 背景总结」

属性作用值 background-color 背景颜色预定义的颜色值/十六进制/RGB 代码 background-image 背景图片 url(图片路径)background-repeat 是否平铺 repeat/no-repeat/repeat-x/repeat-ybackground-position 背景位置 length/position 分别是 x 和 y 坐标，切记如果有精确数值单位，则必须按照先 X 后 Y 的写法 background-attachment 背景固定还是滚动 scroll/fixed 背景简写更简单背景颜色 背景图片地址 背景平铺 背景滚动 背景位置；他们没有顺序背景透明让盒子半透明 background: rgba(0,0,0,0.3); 后面必须是 4 个值

CSS 三大特性

「1. CSS 层叠性」



- 概念：

所谓层叠性是指多种 CSS 样式的 **叠加**

是浏览器处理冲突的一个能力,如果一个属性通过两个相同选择器设置到同一个元素上,那么这个时候一个属性就会将另一个属性层叠掉

-原则:

样式冲突,遵循的原则是 **就近原则**。那个样式离着结构近,就执行那个样式。

样式不冲突,不会层叠。

「2. CSS 继承性」



-概念:

子标签会继承父标签的某些样式,如 **文本颜色和字号**。

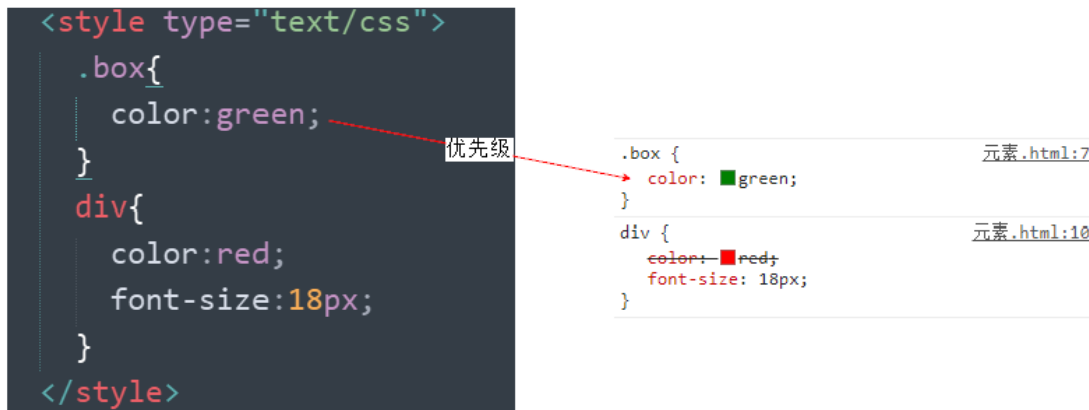
想要设置一个可继承的属性,只需将它应用于父元素即可。

-注意:

恰当地使用继承可以简化代码,降低 CSS 样式的复杂性。比如有很多子级孩子都需要某个样式,可以给父级指定一个,这些孩子继承过来就好了。

子元素可以继承父元素的样式 (**text- , font- , line-这些元素开头的可以继承,以及 color 属性**)

「3. CSS 优先级(CSS 特殊性)」



-概念：定义 CSS 样式时，经常出现两个或更多规则应用在同一元素上，此时，
选择器相同，则执行层叠性
选择器不同，就会出现优先级的。

-权重计算公式：

标签选择器 0,0,0,0 每个元素（标签选择器）0,0,0,1 每个类，伪类 0,0,1,0 每个 ID 0,1,0,0 每个行内样式 style="" 1,0,0,0 每个 !important 最重要的 ∞ 无穷大

值从左到右，左面的最大，一级大于一级，数位之间没有进制，级别之间不可超越。

关于 CSS 权重，我们需要一套计算公式来去计算，这个就是 CSS Specificity（特殊性）

```
div { color: pink !important; }
```

-权重叠加：

div ul li	----->	0,0,0,3
.nav ul li	----->	0,0,1,2
a:hover	----->	0,0,1,1
.nav a	----->	0,0,1,1

-继承的权重是 0：

我们修改样式，一定要看该标签有没有被选中

如果选中了，那么以上面的公式来计权重。谁大听谁的。

如果没有选中，那么权重是 0，因为继承的权重为 0。

盒子模型

css 学习三大重点： **css 盒子模型** 、 **浮动** 、 **定位**

网页布局的本质

首先利用 CSS 设置好盒子的大小，然后摆放盒子的位置。

最后把网页元素比如文字图片等等，放入盒子里面。

1. 盒子模型(Box Model)

盒子模型就是把 HTML 页面中的布局元素看作是一个矩形的盒子，也就是一个盛装内容的容器。

盒子模型由元素的内容、边框 (border)、内边距 (padding)、和外边距 (margin) 组成。

盒子里面的文字和图片等元素是 内容区域

盒子的厚度 我们称为为盒子的边框

盒子内容与边框的距离是内边距

盒子与盒子之间的距离是外边距

W3c 标准盒子模型

标准 w3c 盒子模型的范围包括 margin、border、padding、content

当设置为 **box-sizing: content-box;**时，将采用标准模式解析计算，也是默认模式；

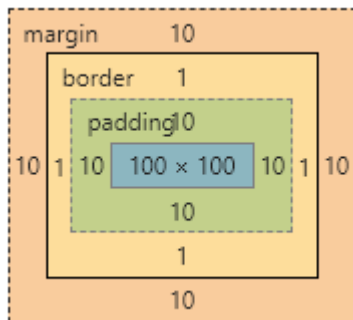
内盒尺寸计算(元素实际大小)

宽度：Element Height = content height + padding + border
(Height 为内容高度)

高度：Element Width = content width + padding + border

(Width 为 内容宽度)

盒子的实际大小：内容的宽度和高度 + 内边距 + 边框



IE 盒子模型

IE 盒子模型的 content 部分包含了 border 和 padding

当设置为 `box-sizing: border-box` 时，将采用怪异模式解析计算；

2. 盒子边框(border)

属性作用 `border-width` 定义边框粗细，单位是 px `border-style` 边框的样式 `border-color` 边框颜色

边框的样式：

`none`：没有边框即忽略所有边框的宽度（默认值）

`solid`：边框为单实线(最为常用的)

`dashed`：边框为虚线

`dotted`：边框为点线

边框综合设置

`border: border-width || border-style || border-color`

`border: 1px solid red;` 没有顺序要求

盒子边框写法总结表：

很多情况下，我们不需要指定 4 个边框，我们是单独给 4 个边框分别指定的。

上边框 下边框 左边框 右边框 `border-top-style: 样式; border-`

border-bottom-style:样式;border-left-style:样式;border-right-style:样式;border-top-width:宽度;border-bottom-width:宽度;border-left-width:宽度;border-right-width:宽度;border-top-color:颜色;border-bottom-color:颜色;border-left-color:颜色;border-right-color:颜色;border-top:宽度 样式 颜色;border-bottom:宽度 样式 颜色;border-left:宽度 样式 颜色;border-right:宽度 样式 颜色;

表格的细线边框：

小明	男
----	---

通过表格的 `cellspacing="0"`,将单元格与单元格之间的距离设置为0,

但是两个单元格之间的边框会出现重叠,从而使边框变粗

通过css属性: `table{ border-collapse:collapse; }`

`collapse` 单词是合并的意思,`border-collapse: collapse;`表示相邻边框合并在一起。

```
<style>
table {
  width: 500px;
  height: 300px;
  border: 1px solid red;
}
td {
  border: 1px solid red;
  text-align: center;
}
table, td {
  border-collapse: collapse; /*合并相邻边框*/
}
</style>
```

2. 内边距(padding)

`padding` 属性用于设置内边距。是指边框与内容之间的距离。

设置

属性作用 padding-left 左内边距 padding-right 右内边距
padding-top 上内边距 padding-bottom 下内边距

padding 简写

值的个数表达意思 1 个值 padding : 上下左右内边距; 2 个值 padding: 上下内边距 左右内边距 ; 3 个值 padding : 上内边距 左右内边距 下内边距 ; 4 个值 padding: 上内边距 右内边距 下内边距 左内边距 ;

当我们给盒子指定 padding 值之后, 发生了 2 件事情:

内容和边框 有了距离, 添加了内边距。

盒子会 **变大**

解决措施: 通过给设置了宽高的盒子, 减去相应的内边距的值, 维持盒子原有的大小。

padding 不影响盒子大小情况: □如果没有给一个盒子 **指定宽度**, 此时, 如果给这个盒子指定 padding, 则不会撑开盒子。

3. 外边距 (margin)

margin 属性用于设置外边距。margin 就是控制 **盒子和盒子之间的距离**

设置

属性作用 margin-left 左外边距 margin-right 右外边距 margin-top 上外边距 margin-bottom 下外边距

margin 值的简写 (复合写法) 代表意思 跟 padding 完全相同。

块级盒子水平居中

盒子必须 **指定宽度 (width)**

然后就给 **左右的外边距都设置为 auto**

实际工作中常用这种方式进行网页布局，示例代码如下：

```
.header { width: 960px; margin: 0 auto;}
```

常见的写法，以下三种都可以。

```
margin-left: auto;    margin-right: auto;
```

```
margin: auto;
```

```
margin: 0 auto;
```

文字居中和盒子居中区别

盒子内的文字水平居中是 `text-align: center;` 而且还可以让 行内元素 和 行内块 居中对齐

块级盒子水平居中 左右 margin 改为 `auto`

插入图片和背景图片区别

插入图片 我们用的最多 比如产品展示类 移动位置只能靠盒模型 padding margin

背景图片 我们一般用于 小图标背景 或者 超大背景图片、背景图片，移动位置只能通过 `background-position`

清除元素的默认内外边距

行内元素为了照顾兼容性,尽量只设置左右内外边距，不要设置上下内外边距。

```
* {  
  padding:0;      /* 清除内边距 */  
  margin:0;       /* 清除外边距 */  
}
```

4.外边距合并

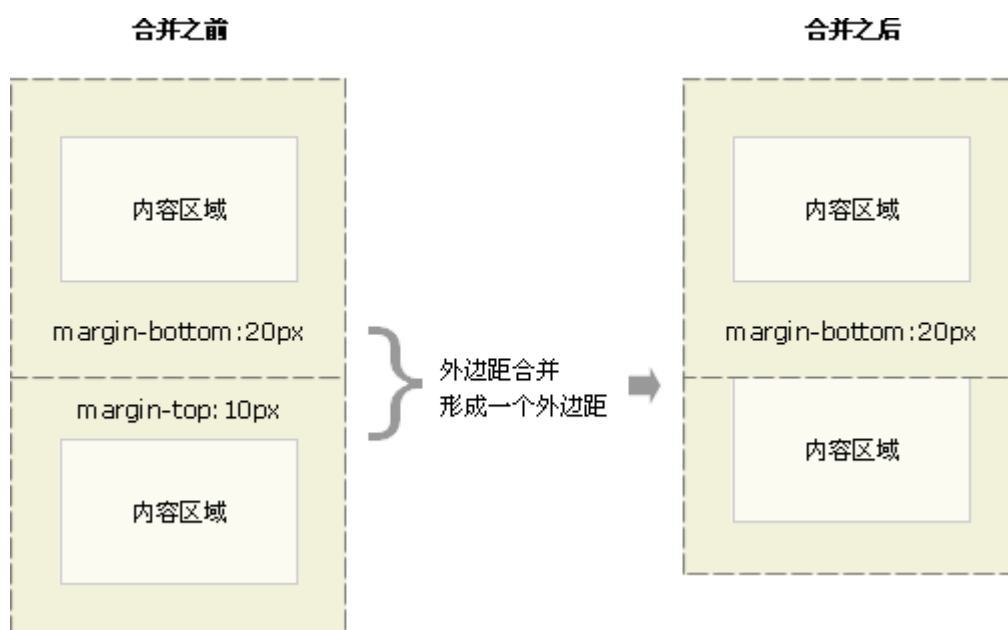
使用 margin 定义块元素的「垂直外边距」时，可能会出现外边距的合并。

(1). 相邻块元素垂直外边距的合并

当上下相邻的两个块元素相遇时，如果上面的元素有下外边距 `margin-bottom`

下面的元素有上外边距 `margin-top`，则他们之间的垂直间距不是 `margin-bottom` 与 `margin-top` 之和

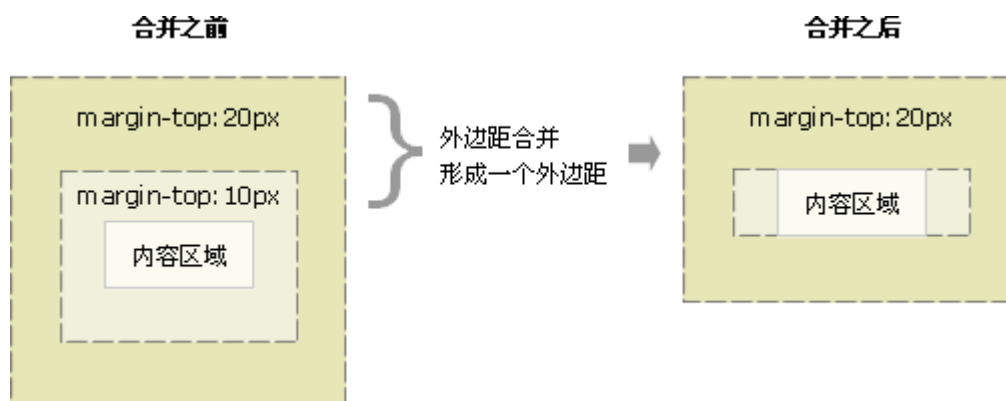
「取两个值中的较大者」这种现象被称为相邻块元素垂直外边距的合并（也称外边距塌陷）。



「解决方案：尽量给只给一个盒子添加 `margin` 值」。

(2). 嵌套块元素垂直外边距的合并（塌陷）

对于两个嵌套关系的块元素，如果父元素没有上内边距及边框
父元素的上外边距会与子元素的上外边距发生合并
合并后的外边距为两者中的较大者



「解决方案 :」

可以为父元素定义上边框。

可以为父元素定义上内边距

可以为父元素添加 `overflow: hidden`。

还有其它方法，比如浮动、固定、绝对定位的盒子不会有问题，后面咱们再总结。。。

盒子模型布局稳定性

优先使用 宽度（width） 其次 使用内边距（padding） 再次 外边距（margin）

`width > padding > margin`

原因：

margin 会有外边距合并 还有 ie6 下面 margin 加倍的 bug（讨厌）所以最后使用。

padding 会影响盒子大小， 需要进行加减计算（麻烦） 其次使用。

width 没有问题（嗨皮）我们经常使用宽度剩余法 高度剩余法来做。

5. CSS3 新增

圆角边框：

`border-radius:length;`

`border-top-left-radius` 定义了左上角的弧度

`border-top-right-radius` 定义了右上角的弧度

`border-bottom-right-radius` 定义了右下角的弧度

`border-bottom-left-radius` 定义了左下角的弧度

其中每一个值可以为 数值或百分比的形式。

技巧：让一个正方形 变成圆圈

`border-radius: 50%;`



如果要在四个角上一一指定，可以使用以下规则：

`border-radius:` 左上角 右上角 右下角 左下角；

四个值：第一个值为左上角，第二个值为右上角，第三个值为右下角，第四个值为左下角。

三个值：第一个值为左上角，第二个值为右上角和左下角，第三个值为右下角

两个值：第一个值为左上角与右下角，第二个值为右上角与左下角

一个值：四个圆角值相同

盒子阴影(box-shadow)：

`box-shadow: offset-x offset-y [blur [spread]] [color] [inset]`

值描述 offset-x 阴影的水平偏移量。正数向右偏移，负数向左偏移。

offset-y 阴影的垂直偏移量。正数向下偏移，负数向上偏移。

blur 可选。阴影模糊距离，不能取负数。spread 可选。阴影大小

color 可选。阴影的颜色 inset 可选。表示添加内阴影，默认为外阴影

```
div {  
  width: 200px;  
  height: 200px;  
  border: 10px solid red;  
  /* box-shadow: 5px 5px 3px 4px rgba(0, 0, 0, .4); */  
  /* box-shadow:水平位置 垂直位置 模糊距离 阴影尺寸（影子大小） 阴影颜色 内/外阴影； */  
  box-shadow: 0 15px 30px rgba(0, 0, 0, .4);  
}
```

浮动

浮动

「1. CSS 布局的三种机制」

网页布局的核心——就是用 **CSS 来摆放盒子**。

CSS 提供了 **3 种机制** 来设置盒子的摆放位置，分别是 **普通流**（标准流）、**浮动** 和 **定位**，其中：

A. 普通流（标准流）

块级元素 会独占一行，**从上向下** 顺序排列；

常用元素：div、hr、p、h1~h6、ul、ol、dl、form、table

行内元素 会按照顺序，**从左到右** 顺序排列，碰到父元素边缘则自动换行；

常用元素：span、a、i、em 等

B. 浮动

让盒子从普通流中**浮**起来,主要作用让多个块级盒子一行显示。

C. 定位

将盒子定在浏览器的某一个位置——CSS 离不开定位，特别是后面的 js 特效。

「2. 什么是浮动」元素的浮动是指设置了浮动属性的元素会

脱离标准普通流的控制,不占位置，脱标
移动到指定位置。

作用

让多个盒子(div)水平排列成一行，使得浮动称为布局的重要手段。
可以实现盒子的左右对齐等等。

浮动最早是用来控制图片，实现文字环绕图片效果。

float 属性会改变元素的 display 属性，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。生成的块级框和我们前面的行内块极其相似。

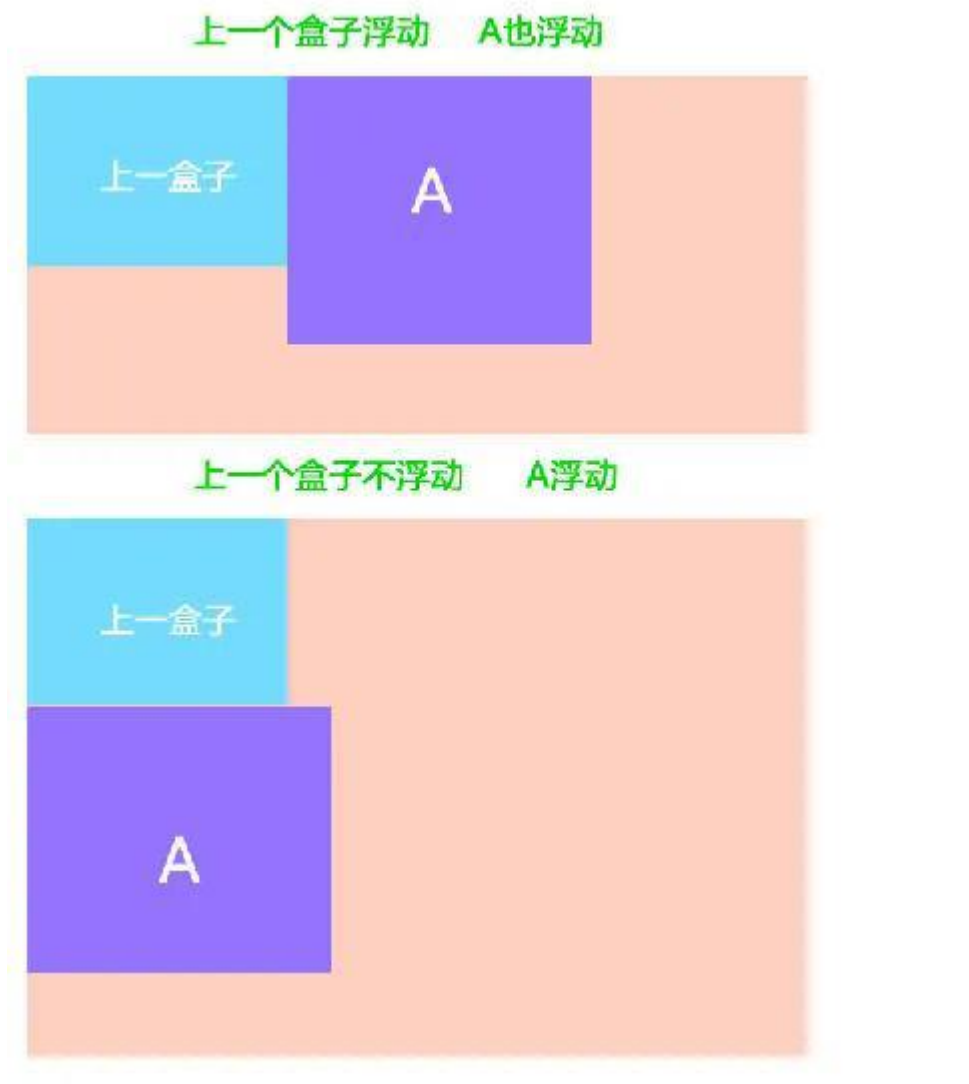
语法

选择器 { float: 属性值; }

属性值描述 none 元素不浮动（默认值）left 元素向左浮动 right 元素向右浮动

浮动只会影响当前的或者是后面的标准流盒子，不会影响前面的标准流。

建议:如果一个盒子里面有多个子盒子，如果其中一个盒子浮动了，其他兄弟也应该浮动。防止引起问题



浮动(float)小结

特点说明浮加了浮动的盒子「**是浮起来**」的，漂浮在其他标准流盒子的上面。漏加了浮动的盒子「**是不占位置的**」，它原来的位置「**漏给了标准流的盒子**」。特「**特别注意**」：浮动元素会改变 display 属性，类似转换为了行内块，但是元素之间没有空白缝隙

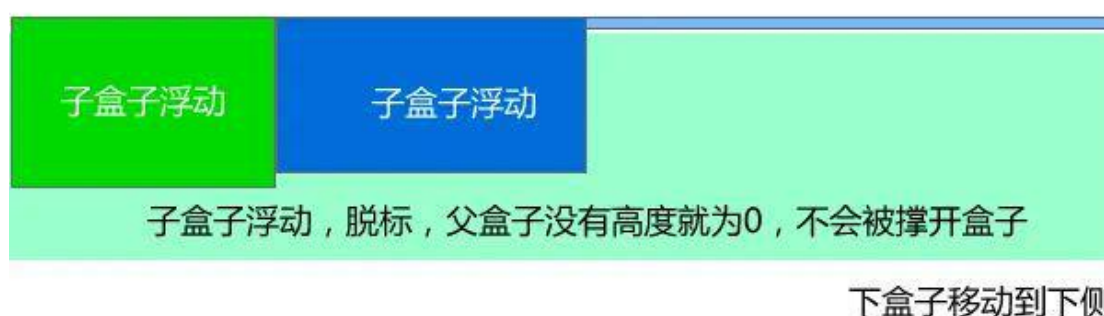
清除浮动

因为父级盒子很多情况下，不方便给高度，但是子盒子浮动就不占有位置，最后父级盒子高度为 0，就影响了下面的标准流盒子。

正常标准流盒子



子盒子浮动



总结：

由于浮动元素不再占用原文档流的位置，所以它会对后面的元素排版产生影响

准确地说，并不是清除浮动，而是清除浮动后造成的影响

清除浮动本质 清除浮动主要为了解决父级元素因为子级浮动引起内部高度为 0 的问题。清除浮动之后，父级就会根据浮动的子盒子自动检测高度。父级有了高度，就不会影响下面的标准流了

清除浮动的方法

选择器 { **clear**: 属性值; } **clear** 清除

属性值描述 left 不允许左侧有浮动元素（清除左侧浮动的影响）

right 不允许右侧有浮动元素（清除右侧浮动的影响） both 同时清除左右两侧浮动的影响

实际工作中,几乎只用 `clear: both`

1).额外标签法(隔墙法)

是 W3C 推荐的做法是通过在浮动元素末尾添加一个空的标签例如 `<div style="clear:both"></div>` , 或则其他标签 `br` 等亦可。

优点：通俗易懂，书写方便

缺点：添加许多无意义的标签，结构化较差。

2).父级添加 `overflow` 属性方法

可以给父级添加：`overflow` 为 `hidden`| `auto`| `scroll` 都可以实现。

优点：代码简洁

缺点：内容增多时候容易造成不会自动换行导致内容被隐藏掉，无法显示需要溢出的元素。

3).使用 `after` 伪元素清除浮动:`after` 方式为空元素额外标签法的升级版，好处是不用单独加标签了

```
.clearfix:after {
    content: "";
    display: block;
    height: 0;
    clear: both;
    visibility: hidden;
}

/* IE6、7 专有 */
.clearfix {
    *zoom: 1;
}
```

优点：符合闭合浮动思想 结构语义化正确

缺点：由于 IE6-7 不支持`:after`，使用 `zoom:1` 触发 `hasLayout`。

4).使用双伪元素清除浮动

```
.clearfix:before,
.clearfix:after {
    content: "";
    display: table;
```

```
}

.clearfix:after {
    clear: both;
}

.clearfix {
    *zoom: 1;
}
```

优点：代码更简洁

缺点：由于 IE6-7 不支持:after，使用 zoom:1 触发 hasLayout。

清除浮动总结

什么时候用清除浮动呢？

父级没高度

子盒子浮动了

影响下面布局了，我们就应该清除浮动了。

清除浮动的方式优点缺点
额外标签法（隔墙法）通俗易懂，书写方便
添加许多无意义的标签，结构化较差。
父级 overflow:hidden; 书写简单
溢出隐藏父级 after 伪元素
结构语义化正确
由于 IE6-7 不支持:after，兼容性问题
父级双伪元素
结构语义化正确
由于 IE6-7 不支持:after，兼容性问题

CSS 属性书写顺序

建议遵循以下顺序：

布局定位属性：display / position / float / clear / visibility / overflow (建议 display 第一个写，毕竟关系到模式)

自身属性：width / height / margin / padding / border / background

文本属性：color / font / text-decoration / text-align / vertical-align / white-space / break-word

其他属性 (CSS3)：content / cursor / border-radius / box-shadow / text-shadow / background:linear-gradient ...

```
.jdc {
  display: block;
  position: relative;
  float: left;
  width: 100px;
  height: 100px;
  margin: 0 10px;
  padding: 20px 0;
  font-family: Arial, 'Helvetica Neue', Helvetica, sans-serif;
  color: #333;
  background: rgba(0,0,0,.5);
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -o-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

定位(position)

「1. 定位详解」

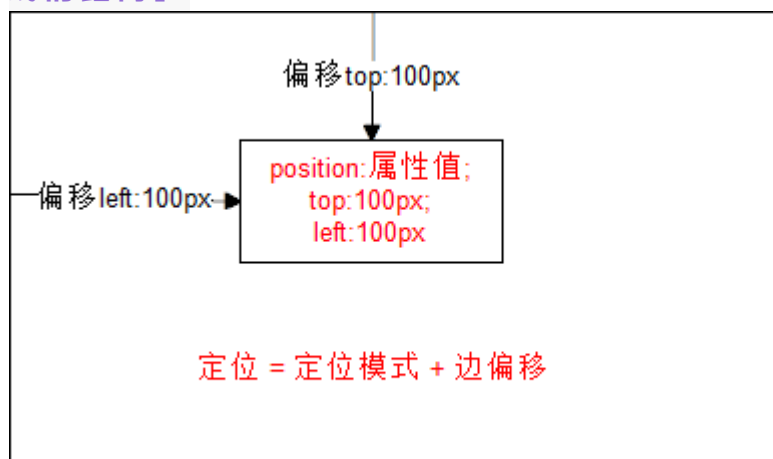
将盒子「定」在某一个「位」置 自由的漂浮在其他盒子(包括标准流和浮动)的上面。

所以，我们脑海应该有三种布局机制的上下顺序□□

标准流在最底层（海底） ----- 浮动 的盒子 在 中间层 （海面） ----- 定位的盒子 在 最上层 （天空）

定位是用来布局的，它有两部分组成：**定位 = 定位模式 + 边偏移**
在 CSS 中，通过 **top**、**bottom**、**left** 和 **right** 属性定义元素的「**边偏移**」：(方位名词)

边偏移属性示例描述 toptop: 80px「顶端」偏移量，定义元素相对于其父元素「上边线的距离」。bottombottom: 80px「底部」偏移量，定义元素相对于其父元素「下边线的距离」。leftleft: 80px「左侧」偏移量，定义元素相对于其父元素「左边线的距离」。rightright: 80px「右侧」偏移量，定义元素相对于其父元素「右边线的距离」



「2. 定位模式(position)」在 CSS 中，通过 position 属性定义元素的「定位模式」，语法如下：

选择器 { position: 属性值; }

值语义 static「静态」定位 relative「相对」定位 absolute「绝对」定位 fixed「固定」定位

「3. 静态定位(static)」

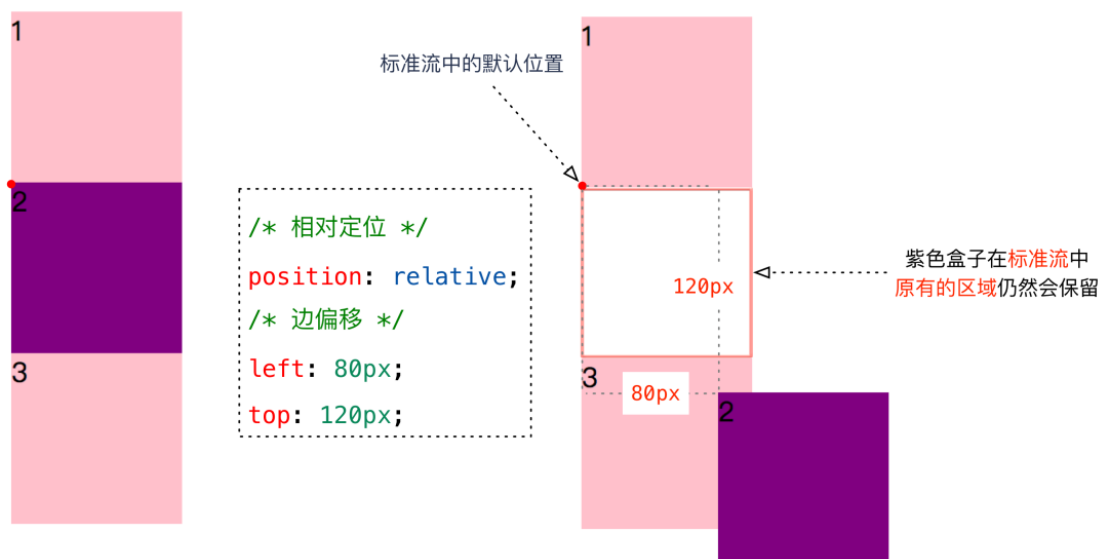
静态定位是元素的默认定位方式，无定位的意思。它相当于 border 里面的 none，不要定位的时候用。

静态定位 按照标准流特性摆放位置。它没有边偏移。

静态定位在布局时几乎不用

「4. 相对定位(relative)」

相对定位是元素相对于它原来在标准流中的位置来说的。



相对于自己原来在标准流中位置来移动的

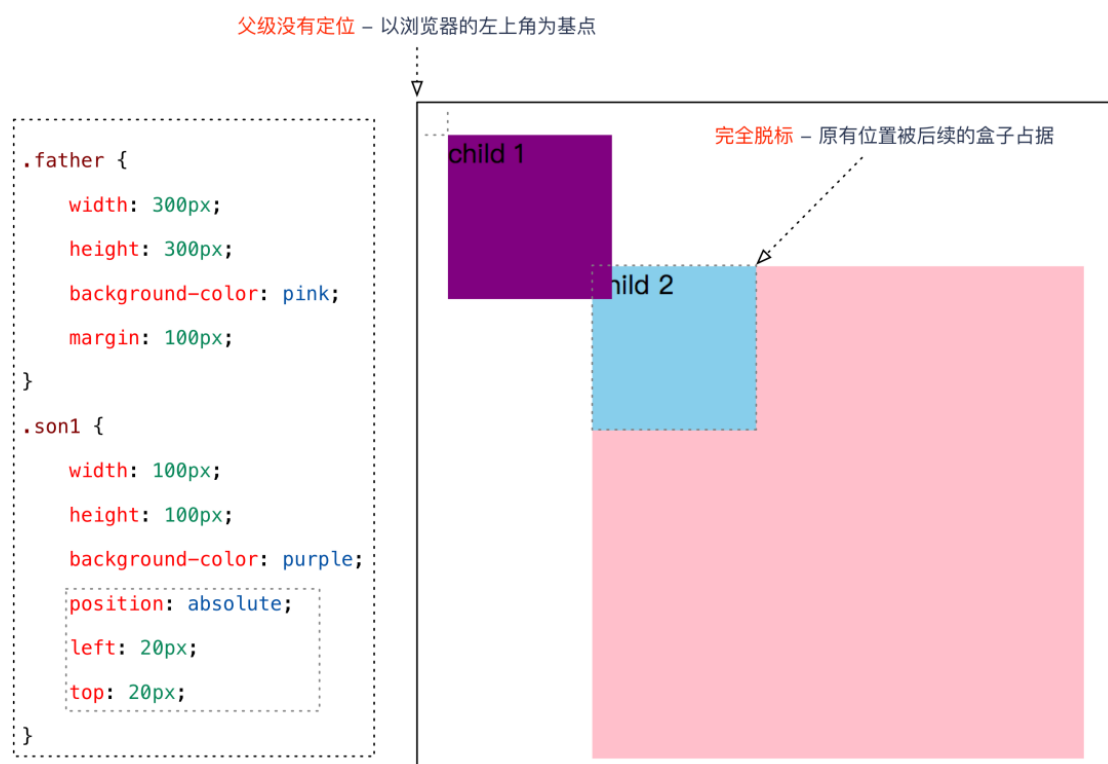
原来在标准流的区域继续占有，后面的盒子仍然以标准流的方式对待它。

「5. 绝对定位 (absolute)」

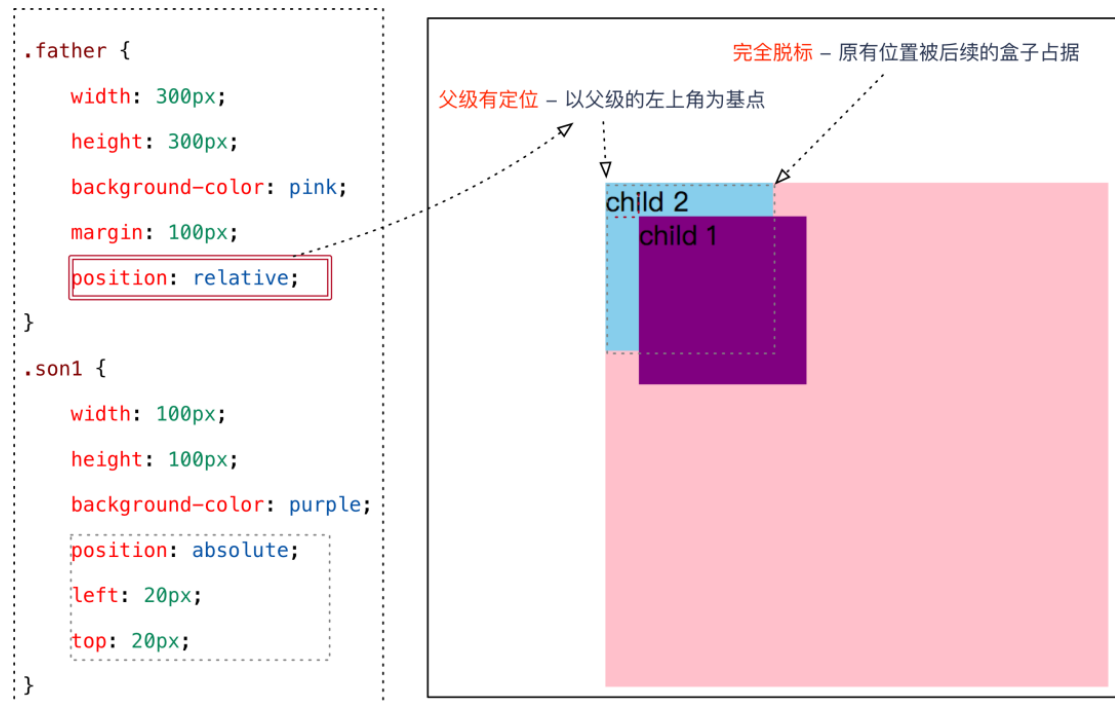
绝对定位是元素以带有定位的父级元素来移动位置

完全脱表--完全不占位置；

父元素没有定位，则以浏览器为准定位(Document 文档)。



父元素有定位



定位口诀--子绝父相

「6. 固定定位(fixed)」

固定定位是绝对定位的一种特殊形式；

完全脱标--完全不占位置；

只认浏览器的可视窗口--浏览器可视窗口+边偏移属性来设置元素的位置

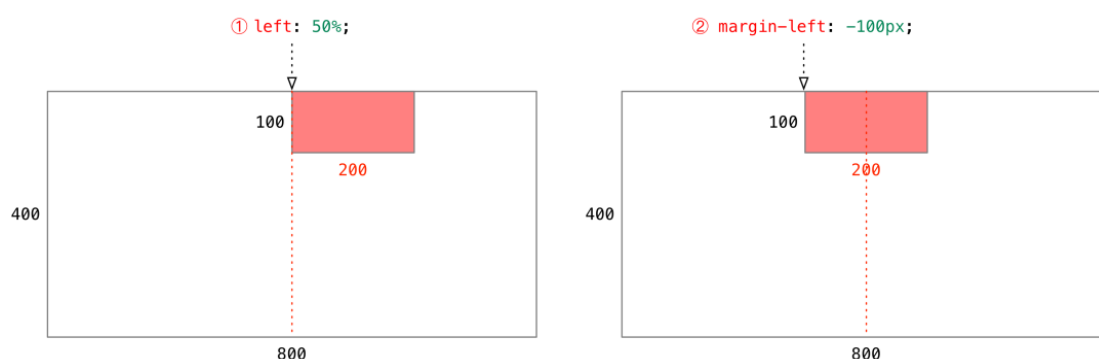
跟父元素没有任何关系；单独使用

不随滚动条滚动

定位(position)的扩展

绝对定位的盒子居中

绝对定位/固定定位的盒子不能通过设置 `margin: auto` 设置水平居中。在使用绝对定位时要向实现水平居中，可以按照下面的方法：



`left: 50%`:让盒子的左侧移动到父级元素的水平中心位置；

`margin-left: -100px`;让盒子向左移动自身宽度的一半。

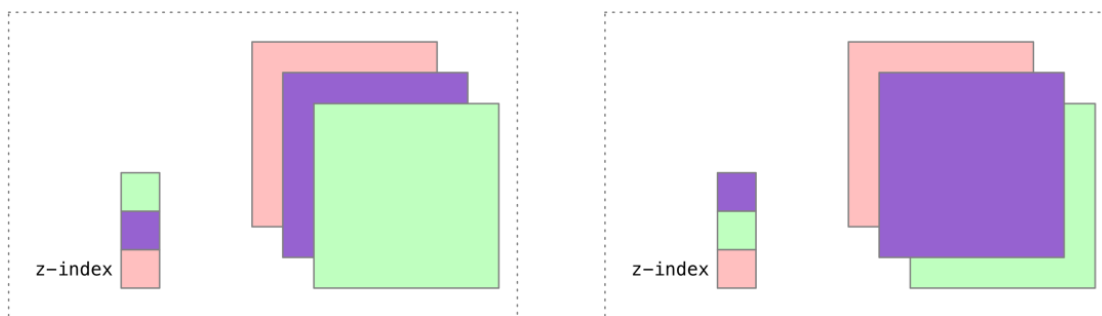
同理垂直居中。

堆叠顺序 (z-index)

在使用「定位」布局时，可能会「出现盒子重叠的情况」。

加了定位的盒子，默认「后来者居上」，后面的盒子会压住前面的盒子。

应用 `z-index` 层叠等级属性可以「调整盒子的堆叠顺序」。如下图所示：



z-index 的特性如下：

属性值：正整数、负整数或 0，默认值是 0，数值越大，盒子越靠上；

如果属性值相同，则按照书写顺序，后来居上；

数字后面不能加单位

z-index 只能用于相对定位、绝对定位和固定定位的元素，其他标准流、浮动和静态定位无效。

定位改变 display 属性

前面提过，display 是显示模式，可以通过以下方式改变显示模式：

可以用 inline-block 转换为行内块

可以用浮动 float 默认转换为行内块（类似，并不完全一样，因为浮动是脱标的）

绝对定位和固定定位也和浮动类似，默认转换的特性转换为行内块。

所以说，一个行内的盒子，如果加了「浮动」、「固定定位」和「绝对定位」，不用转换，就可以给这个盒子直接设置宽度和高度等。

定位小结

定位模式是否脱标占有位置移动位置基准模式转换（行内块）使用情况静态 static 不脱标，正常模式正常模式不能几乎不用相对定位 relative 不脱标，占有位置相对自身位置移动不能基本单独使用绝对定位 absolute 完全脱标，不占有位置相对于定位父级移动位置要和定位父级元素搭配使用固定定位 fixed 完全脱标，不占有位置

相对于浏览器移动位置能单独使用，不需要父级

注意：

边偏移 需要和 定位模式 联合使用，单独使用无效；

top 和 bottom 不要同时使用；

left 和 right 不要同时使用。

CSS 高级技巧

元素的显示与隐藏

目的：让一个元素在页面中消失或者显示出来

场景：类似网站广告，当我们点击关闭就不见了，但是我们重新刷新页面，会重新出现！

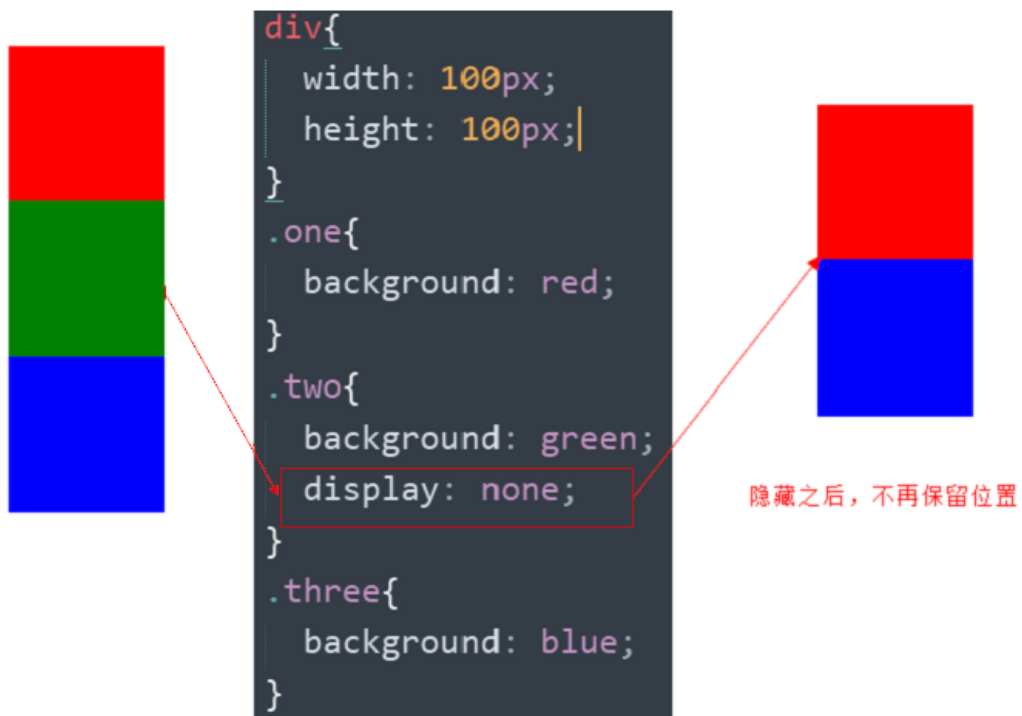
1.1 display 显示（重点）

display 设置或检索对象是否显示或如何显示。

display: none 隐藏对象

特点：隐藏之后，不再保留位置。

display: block 除了转换为块级元素之外，同时还有显示元素的意思。



实际开发场景：配合后面 js 做特效，比如 **下拉菜单**，原先没有，鼠标经过，显示下拉菜单，应用极为广泛

1.2 visibility 可见性

设置或检索是否显示对象

`visibility : visible` ; 对象可视

`visibility : hidden`; 对象隐藏

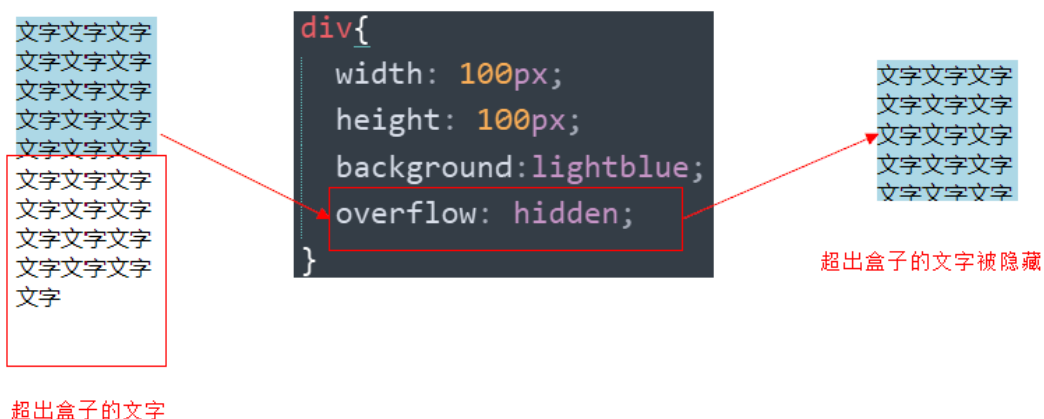
特点：隐藏之后，继续**保留原有位置**。



1.3 overflow 溢出

检索或设置当对象的内容超过其指定高度及宽度时如何管理内容。

属性值描述 visible 不剪切内容也不添加滚动条 hidden 不显示超过对象尺寸的内容，超出的部分隐藏掉 scroll 不管超出内容否，总是显示滚动条 auto 超出自动显示滚动条，不超出不显示滚动条



实际开发场景：

清除浮动

隐藏超出内容，隐藏掉，不允许内容超过父盒子。

1.4 显示与隐藏总结

属性区别用途 display 隐藏对象，不保留位置配合后面 js 做特效，比如下拉菜单，原先没有，鼠标经过，显示下拉菜单，应用极为广泛 visibility 隐藏对象，保留位置使用较少 overflow 只是隐藏超出大小的部分 1. 可以清除浮动 2. 保证盒子里面的内容不会超出该盒子范围

CSS 用户界面样式

所谓的界面样式，就是更改一些用户操作样式，以便提高更好的用户体验。

更改用户的鼠标样式

表单轮廓等。

防止表单域拖拽

2.1 鼠标样式

设置或检索在对象上移动的鼠标指针采用何种系统预定义的光标形状。

属性值描述 default 小白 默认 pointer 小手 move 移动 text 文本 not-allowed 禁止

```
<ul>
  <li style="cursor:default">我是小白 </li>
  <li style="cursor:pointer">我是小手 </li>
  <li style="cursor:move">我是移动 </li>
  <li style="cursor:text">我是文本 </li>
  <li style="cursor:not-allowed">我是文本 </li>
</ul>
```

2.2 轮廓线 outline



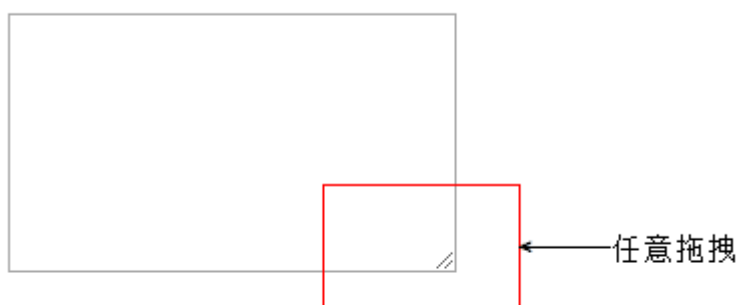
是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

`outline : outline-color || outline-style || outline-width`

但是我们都不关心可以设置多少，我们平时都是去掉的。

最直接的写法是 `outline: 0;` 或者 `outline: none;`

2.3 防止拖拽文本域 resize



```
<textarea style="resize: none;"></textarea>
```

2.4 用户界面样式总结

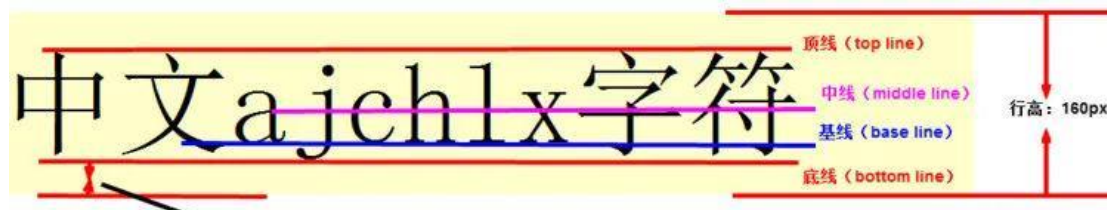
属性用途用途鼠标样式更改鼠标样式 `cursor` 样式很多，重点记住 `pointer` 轮廓线表单默认 `outline` 轮廓线，我们一般直接去掉，`border` 是边框，我们会经常用防止拖拽主要针对文本域 `resize` 防止用户随意拖拽文本域，造成页面布局混乱，我们 `resize:none`

vertical-align 垂直对齐

有宽度的块级元素居中对齐，是 `margin: 0 auto;`

让文字居中对齐，是 `text-align: center;`

`vertical-align` 垂直对齐，它只针对于「行内元素」或者「行内块元素」



设置或检索对象内容的垂直对其方式。

`vertical-align: baseline | top | middle | bottom`

注意：

`vertical-align` 不影响块级元素中的内容对齐，它只针对于「行内元素」或者「行内块元素」，

特别是行内块元素，通常用来控制 图片/表单 与文字的对齐。

3.1 图片、表单和文字对齐

我们可以通过 `vertical-align` 控制图片和文字的垂直关系了。默认的图片会和文字基线对齐。



模式	单词
基线对齐：  默认的是文字和图片基线对齐	<code>vertical-align: baseline;</code>
垂直居中：  默认的是文字和图片基线对齐	<code>vertical-align: middle;</code>
顶部对齐：  默认的是文字和图片基线对齐	<code>vertical-align: top;</code>

3.2 去除图片底侧空白缝隙



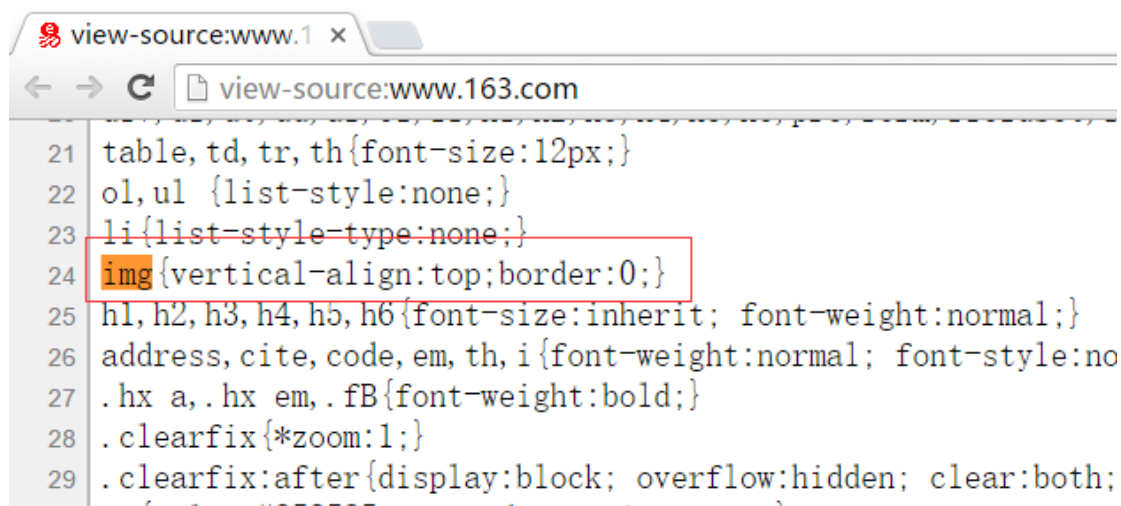
父盒子由图片撑开，图片下面会多出缝隙

原因：图片或者表单等行内块元素，他的底线会和父级盒子的基线对齐。

就是图片底侧会有一个空白缝隙。

解决方法：

给 `img` `vertical-align: middle | top | bottom` 等等。让图片不要和基线对齐。



给 `img` 添加 `display: block`; 转换为块级元素就不会存在问题了。

溢出的文字省略号显示

4.1 white-space

`white-space` 设置或检索对象内文本显示方式。通常我们使用于强制一行显示内容

`white-space:normal` ; 默认处理方式

`white-space:nowrap` ; 强制在同一行内显示所有文本，直到文本结束或者遭遇 `br` 标签对象才换行。

4.2 text-overflow 文字溢出

设置或检索是否使用一个省略标记 (...) 标示对象内文本的溢出

`text-overflow: clip` ; 不显示省略标记 (...), 而是简单的裁切

`text-overflow: ellipsis` ; 当对象内文本溢出时显示省略标记 (...)

「注意」:

一定要首先强制一行内显示，再次和 `overflow` 属性 搭配使用

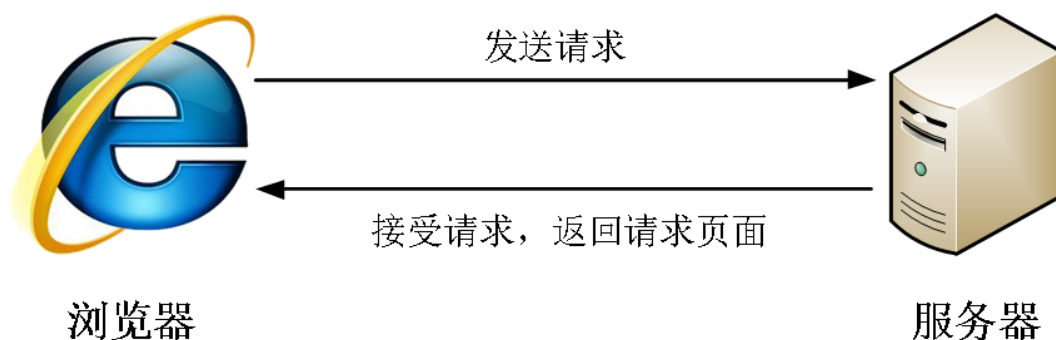
文字文字文...

4.3 总结三步曲

```
/*1. 先强制一行内显示文本*/  
white-space: nowrap;  
/*2. 超出的部分隐藏*/  
overflow: hidden;  
/*3. 文字用省略号替代超出的部分*/  
text-overflow: ellipsis;
```

CSS 精灵技术 (sprite)

CSS 精灵技术 (也称 CSS Sprites、CSS 雪碧)。



图所示为网页的请求原理图，当用户访问一个网站时，需要向服务器发送请求，网页上的每张图像都要经过一次请求才能展现给用户。

然而，一个网页中往往会应用很多小的背景图像作为修饰，当网页中的图像过多时，服务器就会频繁地接受和发送请求，这将大大降低页面的加载速度。

为什么需要精灵技术：为了有效地减少服务器接受和发送请求的次数，提高页面的加载速度。

5.1 精灵技术讲解

CSS 精灵其实是将网页中的一些背景图像整合到一张大图中（精灵图），然而，各个网页元素通常只需要精灵图中不同位置的某个小图，要想精确定位到精灵图中的某个小图。



这样，当用户访问该页面时，只需向服务发送一次请求，网页中的背景图像即可全部展示出来。

我们需要使用 CSS 的：


`background-image`、

`background-repeat`

`background-position` 属性进行背景定位，

其中最关键的是使用 `background-position` 属性精确地定位。

5.2 精灵技术使用的核心总结

首先我们知道，css 精灵技术主要针对于背景图片，插入的图片是不需要这个技术的。

精确测量，每个小背景图片的大小和位置。

给盒子指定小背景图片时，背景定位基本都是负值。

滑动门



6.1 滑动门出现的背景

制作网页时，为了美观，常常需要为网页元素设置特殊形状的背景，比如微信导航栏，有凸起和凹下去的感觉，最大的问题是里面的字数不一样多，咋办？



为了使各种特殊形状的背景能够自适应元素中文本内容的多少，出现了 CSS 滑动门技术。它从新的角度构建页面，使各种特殊形状的背景能够自由拉伸滑动，以适应元素内部的文本内容，可用性更强。最常见于各种导航栏的滑动门。

6.2 核心技术

核心技术就是利用 CSS 精灵（主要是背景位置）和 盒子 padding 撑开宽度，以便能适应不同字数的导航栏。

一般的经典布局都是这样的：

```
<li>
  <a href="#">
    <span>导航栏内容</span>
  </a>
</li>

* {
  padding:0;
  margin:0;
}

body{
  background: url(images/wx.jpg) repeat-x;
}

.father {
  padding-top:20px;
}

li {
  padding-left: 16px;
```

```

height: 33px;
float: left;
line-height: 33px;
margin: 0 10px;
background: url(../images/to.png) no-repeat left ;
}
a {
padding-right: 16px;
height: 33px;
display: inline-block;
color: #fff;
background: url(../images/to.png) no-repeat right ;
text-decoration: none;
}
li: hover,
li: hover a {
background-image: url(../images/ao.png);
}

```

总结：

a 设置 背景左侧，padding 撑开合适宽度。

span 设置背景右侧，padding 撑开合适宽度 剩下由文字继续撑开宽度。

之所以 a 包含 span 就是因为 整个导航都是可以点击的。

CSS 三角形

```

div {

width: 0;


height: 0;
line-height: 0;
font-size: 0;
border-top: 10px solid red;

border-right: 10px solid green;

border-bottom: 10px solid blue;

```

```
border-left: 10px solid #000;  
  
}
```



我们用 css 边框可以模拟三角效果

宽度高度为 0

我们 4 个边框都要写，只保留需要的边框颜色，其余的不能省略，都改为 transparent 透明就好了

为了照顾兼容性 低版本的浏览器，加上 font-size: 0; line-height: 0;