

python 前端 css

阅读目录

- [CSS 选择器](#)
- [选择器的优先级](#)
- [CSS 属性操作](#)
- [CSS 盒子模型](#)

CSS (**C**ascading **S**tyl**S**heet, 层叠样式表) 是一种用来表现 HTML 或 XML 等**文件样式**的计算机语言.

作用:是用来美化 HTML 标签的,相当于给页面化妆.

每个 css 都是有两部分组成: 选择器和声明. 声明有包括属性和值,每个声明后用分号结束.

选择器:就是明确文件样式的作用对象. 是一个选择一个标签或多个标签的过程.

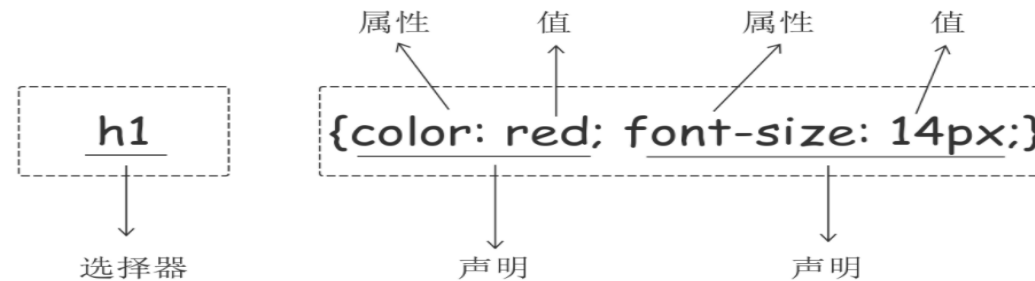
声明:就是样式

CSS 三大特性:

- 1.层叠性:当多个样式作用到同一标签,当发生冲突时,总是执行最后的那个样式代码也就是覆盖.
- 2.继承性: 文字的所有属性都可以继承,h1~h6 不能继承大小,a 标签不能继承颜色
- 3.优先级:

css 的语法:

选择器{属性:值} # 注意: 而不是"=" 号



CSS 注释;

/*这是注释*/

CSS 引入的几种方式

1.行内显示:行内式是在标记的 style 属性中设定 CSS 样式。不推荐大规模使用

```
<p style="color: red">Hello world.</p>
```

2.嵌入式:将 CSS 样式集中写在网页的<head> </head> 标签对的<style> </style> 标签对中。

```
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    p{
      background-color: red;
    }
  </style>
</head>
```

3.外部样式:就是将 css 写在一个 stylesheet 的文件中, 然后在页面中的 head 标签内进行引入即可。推荐使用此方式。

stylesheet 中的代码

```
p {color:red}
```

正文中的代码

```
<head>
  <meta http-equiv="x-ua-compatible" content="IE=edge" charset="UTF-8">
  <title>测试</title>
  <link rel="stylesheet" href="1.css">
</head>
<body>
<p>我是中国人</p>
<p>我爱北京天安门</p>
<p>哈哈</p>
</body>
```

结果:

我是中国人

我爱北京天安门

哈哈

CSS 选择器

基本选择器

1. 标签选择器

```
p {color: "red";}
```

2. ID 选择器

```
#i1 {                                /*注意 ID 开头要用#号*/  
    background-color: red;  
}
```

3. 类别选择器(重点)

```
<!DOCTYPE html>  
<html lang="zh-CN">  
<head>  
    <meta http-equiv="x-ua-compatible" content="IE=edge" charset="UTF-8">  
    <title>测试</title>  
    <style>  
        .c1{color:darkred}  
    </style>  
</head>  
<body>  
<p class="c1">我是中国人</p>    #class 里面可以写多个值  
<p class="c1">我爱北京天安门</p>  
<p>哈哈</p>  
</body>  
</html>
```

结果:

我是中国人
我爱北京天安门
哈哈

注意：
样式类名不要用数字开头（有的浏览器不认）。
标签中的 class 属性如果有多个，要用空格分隔。

4.通用选择器

```
* {  
  color: white;  
}
```

复合选择器

1.并集选择器：为多个标签同时设定样式

格式: 选择器,选择器

```
/*为所有的 div 和 p 标签设置边框属性*/  
div, p {  
  border: 1px solid red;  
}
```

2.后代选择器

格式:选择器(标签|类选择器|ID)空格选择器(标签|类选择器|ID)

特点:无限隔代,可以不用写它父亲,只写他爷爷和孙子就醒了,

```
/*li 内部的 a 标签设置字体颜色*/  
li a {  
  color: green;  
}
```

3.儿子选择器:

```
/*选择所有父级是 <div> 元素的 <p> 元素*/
```

```
div>p {  
    font-family: "Arial Black", arial-black, cursive;  
}
```

4.毗邻选择器

```
/*选择所有紧接着<div>元素之后的<p>元素*/  
div+p {  
    margin: 5px;  
}
```

5.弟弟选择器

```
/*!l 后面所有的兄弟 p 标签*/  
#il~p {  
    border: 2px solid royalblue;  
}
```

6.交集选择器 :同一个标签

格式: 标签+类或 id{属性:值}

```
<head>  
    <meta http-equiv="x-ua-compatible" content="IE=edge" charset="UTF-8">  
    <title>Title</title>  
    <style>  
        .box{  
            font-size: 12px;  
        }  
        div.box{  
            color: red;  
        }  
    </style>  
  
</head>  
<body>  
<div class="box">你好北京</div>  
<p class="box">你好中国</p>
```

</body>

效果:

你好北京

你好中国

属性选择器

/*用于选取带有指定属性的元素。*/

```
p[title] {  
  color: red;
```

/*用于选取带有指定属性和值的元素。*/

```
p[title="213"] {  
  color: green;  
}
```

伪类选择器

a:link {color: #FF0000} /* 未访问的链接 */ 超链接的默认状态

a:visited {color: #00FF00} /* 已访问的链接 */ 连接访问之后的状态

a:hover {color: #FF00FF} /* 鼠标移动到链接上 的状态*/ 不一定是连接, 别的东西也可以

a:active {color: #0000FF} /* 选定的链接不松手时的状态*/

:focus {属性:值} /*获取光标焦点*/

text-decoration:none 连接下无下划线

选择器的优先级

继承:

继承是 CSS 的一个主要特征, 它是依赖于祖先-后代的关系的。继承是一种机制, 它允许样式不仅可以应用于某个特定的元素, 还可以应用于它的后代。例如一个 body 定义了的字体颜色值也会应用到段落的文本中。

```
body {  
  color: red;  
}
```

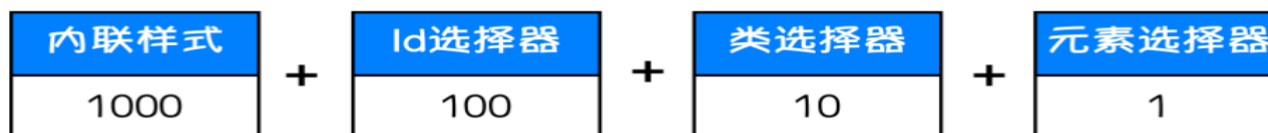
此时页面上所有标签都会继承 body 的字体颜色。然而 CSS 继承性的权重是非常低的，是比普通元素的权重还要低的。我们只要给对应的标签设置字体颜色就可覆盖掉它继承的样式。

```
p {  
  color: green;  
}
```

此外，继承是 CSS 重要的一部分，我们甚至不用去考虑它为什么能够这样，但 CSS 继承也是有限制的。有一些属性不能被继承，如：border, margin, padding, background 等。

样式的优先级

CSS选择器优先级



- 内联样式的权重为1000
- id选择器的权重为100
- 类选择器的权重为10
- 元素选择器的权重为1

万不得已可以使用!import

!import>内联>id 选择器>类选择器>元素选择器>继承

内联样式就是在标签中定义样式

元素选择器就是标签选择器

CSS 属性操作

CSS 文本属性操作

文本颜色

颜色属性被用来设置文字的颜色。

颜色是通过 CSS 最经常的指定：

- 十六进制值 - 如: #FF0000
- 一个 RGB 值 - 如: RGB(255,0,0)
- 颜色的名称 - 如: red

水平对齐

text-align (align [əˈlaɪn]) 排列: 属性规定元素中的文本的水平对齐方式。

注意: 适用于范围:只适用于文本

- left 把文本排列到左边。默认值: 由浏览器决定
- right 把文本排列到右边
- center 把文本排列到中间
- justify 实现两端对齐文本效果

/*

font-size: 10px; 文字大小

line-height: 200px; 文本行高 通俗的讲, 文字高度加上文字上下的空白区域的高度, **结论:当一行文字的行高等于父元素的高度时, 垂直居中显示.**

vertical-align: -4px 设置元素内容的垂直对齐方式, 只对行内元素有效, 对块级元素无效

text-decoration: none text-decoration 属性用来设置或删除文本的装饰。主要是用来删除链接的下划线

font-family: 'Lucida Bright'

font-weight: lighter/bold/bolder/ normal 字体的粗细 细, 粗, 加粗的时候写 700, 不推荐 bold

font-style: oblique 斜体

text-indent: 150px; 首行缩进 150px

letter-spacing: 10px; 字母间距

word-spacing: 20px; 单词间距

text-transform: capitalize/uppercase/lowercase ; 文本转换, 用于所有字句变成大写或小写字母, 或每个单词的首字母大写

*/

文本其他

4.2 文本属性连写

```
font: font-style font-weight font-size/line-height  
font-family;
```

◆：注意：font:后边写属性的值。一定按照书写顺序。|

```
Font:italic 700 16px/40px 微软雅黑;
```

如果想连写,font-family 和 font-size 必须写

背景颜色

/*背景颜色*/

```
background-color: red;
```

/*背景图片*/

```
background-image: url('1.jpg');
```

/*如果这里不设置高度宽度, 图片的大小会根据字体的大小判定*/

/* 背景重复 repeat(默认):背景图片平铺排满整个网页 repeat-x: 背景图片只在水平方向上平铺 repeat-y: 背景图片只在垂直方向上平铺 no-repeat: 背景图片不平铺 */

/*当设置的大小宽度超过了背景图片的大小, 是否用重复背景图片填充*/

```
background-repeat: no-repeat;
```

/*背景位置*/ background-position: right top (20px 20px);

使用背景图片的一个常见案例就是很多网站会把很多小图标放在一张图片上, 然后根据位置去显示图片。减少频繁的图片请求。

边框属性

- border-width

- style>

- p{ border-width: thick; /*只有 border-style 不等于 none 是这个属性才有作用, 这个指的是线条的宽度而不是框的宽度*/
- border-style: solid; /*注意和 border:solid 的区别后者对于 border-width 不起作用*/

- border-color: red;
- }
- </style>

- border-style

- border-style: none dotted solid double dashed; 无边框 点 实线 双线 虚线
- border-bottom-color 底边边框颜色
- border-radius: 10px 把正方形的直角变成圆角

- border-color

通常使用简写方式:

```
#il {  
  border: 2px solid red;  
}
```

背景属性

/*背景颜色*/

background-color: red;

/*背景图片*/

background-image: url('1.jpg'); /*如果这里不设置高度宽度, 图片的大小会根据字体的大小判定*/

/*

背景重复

repeat(默认): 背景图片平铺排满整个网页

repeat-x: 背景图片只在水平方向上平铺

repeat-y: 背景图片只在垂直方向上平铺

no-repeat: 背景图片不平铺

/背景不平铺/

background-repeat: no-repeat;

/*背景定位*/

background-position: left right top bottom center 如果只写一个值, 默认另一个值居中

background-position: right top (20px(水平) 20px(垂直));

background: #ffffff url('1.png') no-repeat right top;

使用背景图片的一个常见案例就是很多网站会把很多小图标放在一张图片上, 然后根据位置去显示图片。减少频繁的图片请求。

display

块级标签的特点: 独占一行,可以设置宽和高

内联标签的特点: 可以放在一行,不可以设置宽和高,行长度根据内容自适应.

属性值:

none: 不显示

block: 将内联标签变为块级标签.

如果有多个内联标签转换为块级标签, 它会把每个内联标签都变成块级标签

```
<span>span1</span>
```

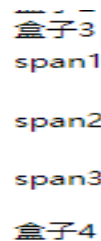
```
<span>span2</span>
```

```
<span>span3</span>
```

style:代码

```
span{  
    display: block;  
    width: 50px;  
    height: 80px;  
    margin-top: 10px;  
  
}
```

结果:



盒子3
span1
span2
span3
盒子4

inline: 将块级标签变为内联标签.

inline-block :即在一行显示又能设置长和宽

当有有内联标签转换为块级标签, 它会把每个内联标签连在一起变成一个块级标签

```
<span>span1</span>
```

```
<span>span2</span>
```

```
<span>span3</span>
```

style:代码

```
span{  
    display: display;  
    width: 50px;  
    height: 80px;  
    margin-top: 10px;  
  
}
```

结果:

盒子1
盒子2
盒子3
span1 span2 span3

盒子4
span4

CSS 盒子模型

- **margin:** 用于控制元素与元素之间的距离；margin 的最基本用途就是控制元素周围空间的间隔，从视觉角度上达到相互隔开的目的。
简单的就是外边距
- **padding:** 用于控制内容与边框之间的距离； 就是 Word 中的内边距
- **Border(边框):** 围绕在内边距和内容外的边框。
- **Content(内容):** 盒子的内容，显示文本和图像。

盒子模型出现的原因:让页面布局更合理.

padding

```
p {  
    padding-top: 5px;  
    padding-right: 10px;
```

```
padding-bottom: 15px;
padding-left: 20px;
}
```

顺序：上右下左

补充：

- 提供一个，用于四边；
- 提供两个，第一个用于上 - 下，第二个用于左 - 右；
- 如果提供三个，第一个用于上，第二个用于左 - 右，第三个用于下；
- 提供四个参数值，将按上 - 右 - 下 - 左的顺序作用于四边；

推荐使用这种方法

```
padding-test {
padding: 5px 10px 15px 20px;
}
```

内边距是否会影响盒子大小?(重点)

1.当子盒子不继承父盒子时,也就是自定义子盒子时:内边距会影响盒子的大小, border 也会影响盒子的大小

盒子的宽度=定义的宽度+border 的宽度+padding 的宽度

实例定义一个盒子 width:500px, height:300px 盒子里边有一个图片(也相当于盒子)width:180px height:180px

如何让一个图片在一个盒子里居中?

方法: 首先(500-180)/2=160 padding-left =160px, 由于内边距会撑大盒子,这时候盒子变成 width=(500+160)=660px 这时候你需要把盒子 width 变成 (500-160px)=340px 就可以了,width 方向就会居中.

在说一下高度方面: (300-180)/2=60px padding-bottom=60px ,由于内边距会撑大盒子,这时候盒子变成 height=(300+60px)=360px 这时候你需要把盒子 height 变成 (300-60px)=240px 就可以了,height 方向就会居中.

方法 2: 首先定义为 display:inline-block 然后 text-align:center 宽度方向就会居中了,高度方向还是按原来的方法做

2.包含 (嵌套) 的盒子, 如果子盒子没有定义宽度, 给子盒子设置左右内边距, 一般(前提)不会撑大盒子的宽度. 注意子盒子的宽度如果你没有定义会继承父盒子的,但是高度不会继承,但是高度会撑大盒子

如何解决内边距影响盒子的大小?

1.如果只是要改变左右的 padding 的话就用上边的第二种继承的方法.

2.如果要改变这种现象的话就用 margin

margin (外边距)

```
div {
margin-top:100px;
```

```
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
}
```

推荐使用简写:

```
div {
    margin: 5px 10px 15px 20px;
}
```

顺序: 上右下左

居中: (一个盒子在另一个盒子里居中 不能用 `text-align:center`,因为它是盒子内容的居中):

```
div {
    margin: 0 auto;
}
```

border 属性

```
border-bottom: solid;
border-bottom-color: red;
```

`border: 0 none` 去边框, 写 0 是为了有更好的兼容性

`outline-style: none` 去掉轮廓线

简写: `border: 像素 线型 颜色` 没有顺序要求的, 线型不能少

边框合并 `border-collapse:collapse`;把两个边框一个像素合成一个像素了

垂直方向外边距的合并

有两个上下排列的盒子, 其中上面的盒子设置 `margin-bottom:20px`, 下边的盒子设置 `margin-top:120px` , 他俩之间的空隙并不是 120px, 而是 100px, 这是浏览器渲染内核时的规则, 只选择最大值

外边距塌陷

外边距塌陷的定义: 当嵌套盒子直接给子盒子设置垂直方向的外边距的时,你会发现大小盒子会一起上下移动.这就是外边距塌陷现象.

如何解决这种现象?

方法一: 给父盒子设置 `border` , 注意父盒子的实际长度要减去 `border` 的厚度. 为什么会这样呢? 如果你没有设置 `border` ,子盒子和父盒子会以浏览器的边框为基准.当你设置了 `border` 时,就会以 `border` 为基准.

方法二: 给父盒子设置 `overflow:hidden` , 也就是溢出 推荐使用这种方法,免去了算 `border` 的宽度的麻烦. 原因:bfc 了解一下就可以

float 属性

`float`, 顾名思义是漂浮, 浮动的意思。但是在 `css` 中, 它被理解成浮动。其实就是 Word 中的文字环绕

float 有四个属性，即

```
1 float:none;
2 float:left;
3 float:right;
4 float:inherit;
```

比较常用的两个属性值是左浮动和右浮动。在接下来的分享中，只会拿左浮动作为例子。其他浮动属性值与左浮动原理相同。

浮动的初衷就是为了文字环绕效果,浮动就是脱离标签

代码:

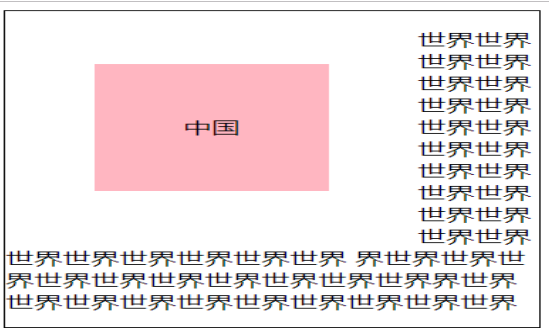
```
<style>
    .container {
    width: 300px;
    height: 300px;
    border: 1px solid black;
    }

    .container .content {
    float: left;
    /*width: 150px;*/
    /*height: 150px;*/
    background-color: lightpink;
    margin: 50px 50px 50px 50px;
    padding: 50px 50px 50px 50px;
    }

</style>
</head>
<body>
<div class="container">
    <div class="content">中国</div>
    <p>
世界世界世界世界世界世界世界世界世界
    </p>
</div>
```

```
</body>
</html>
```

效果:



浮动规则:

- 1.浮动只控制自己.
- 2.如果前面的标签也是浮动的就挨着放.
- 3.如果前面的标签不是浮动的就在下一行挨着放.
- 4,如前面是浮动标签,,而你自己不是浮动标签,则你放在浮动标签下

父标签塌陷: 当你不设父标签的宽高且子标签设置为浮动 且父标签没有内容,这时你会看不到父标签,因为子标签浮动起来了,没有东西来撑起父标签,这种现象就是父标签塌陷. 如何解决这个问题让父标签不塌陷呢?

- 1.给父标签设置高宽
- 2.给父标签添加内容

以上方法都不灵活

- 3.清除浮动就可以了,父标签就可以根据最高的那个标签来设置自己的高

清除浮动:

```
clear:
    left 清除浮动就是当设为清除浮动是这个快要找到一行左边没有浮动快的行把自己放下
    right
    both
    none
```

:after 和:before

清除浮动: 一般用这个

style 代码

```
.clearfix:after {  
    content: "";  
    display: block;  
    clear: both;  
}
```

css 代码

```
<div class="clearfix">  
    <div class="a" style="background-color: red">我是 A</div>  
    <div class="b" style="background-color: green">我是 B</div>  
    <div class="c" style="background-color: gray">我是 C</div>  
</div>
```

定位(position):

position:

relative (相对定位):根据自己原来的位置来做定位 ()

/*在自己原来的位置的基础上向下平移 200, 向右平移 200px

position: relative

left:200px

top:200px

absolute (绝对定位):根往上找已经相对定位的标签的左上角基准点来做定位, 如果没有设置相对定位则这个方法不能用 (通常配合相对定位使用)

fixed (固定定位)

z-index:设置对象的层叠顺序, 数值大的会覆盖在数值小的标签之上。

```
#i2 {  
    z-index: 999;  
}
```