1. CSS的发展历程

css 版本有2.0和3.0版本，因为CSS3有兼容性问题，后边要标注好，做特殊标明。css 主要是解决 html 混乱的结构和样式的问题，例如

<hr width = “500” color = “red”/> 等等，有一些标签并不能直接用 color 改变颜色。

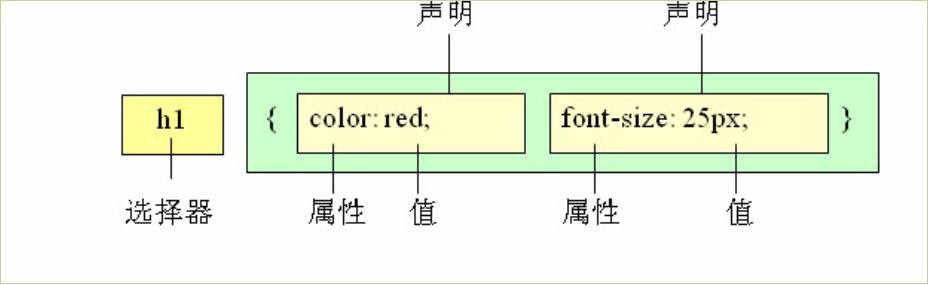
css让 HTML 专注做结构。而CSS 专注于样式。

1. CSS初识

CSS(Cascading Style Sheets)。

CSS通常称为CSS样式表或层叠样式表（级联样式表）。

1. CSS样式规则





1.选择器用于指定CSS样式作用的HTML对象，花括号内是对该对象设置的具体样式。

2.属性和属性值以“键值对”的形式出现。

3.属性是对指定的对象设置的样式属性，例如字体大小、文本颜色等。

4.属性和属性值之间用英文“:”连接。

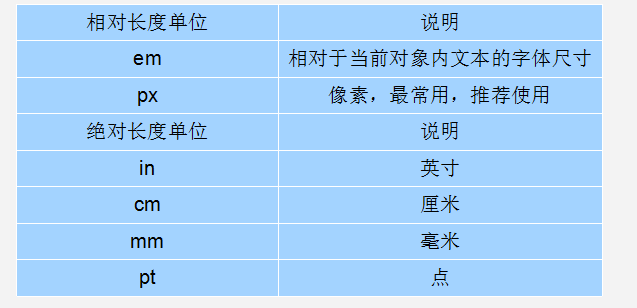
5.多个“键值对”之间用英文“;”进行区分。

选择器：选择谁，哪个html标签。

注：CSS里边只有冒号，没有等于号。

1. CSS字体样式属性
2. font-size：字号大小。

该属性的值可以使用相对长度单位，也可以使用绝对长度单位。其中，相对长度单位比较常用，推荐使用像素单位px。绝对长度单位使用较少，几乎是不用。具体如下：



1. font-family:字体

(1). 现在网页中普遍使用14px+。

(2). 尽量使用偶数的数字字号。ie6等老式浏览器支持奇数会有bug。

(3). 各种字体之间必须使用英文状态下的逗号隔开。

(4). 中文字体需要加英文状态下的引号，英文字体一般不需要加引号。当需要设置英文字体时，**英文字体名必须位于中文字体名之前**。

(5). 如果字体名中包含**空格、#、$等符号**，则该字体必须加英文状态下的**单引号或双引号**，例如font-family: "Times New Roman";。

(6). 尽量使用系统默认字体，保证在任何用户的浏览器中都能正确显示。

**注**：如果所有的字体都找不到，则使用系统的字体。

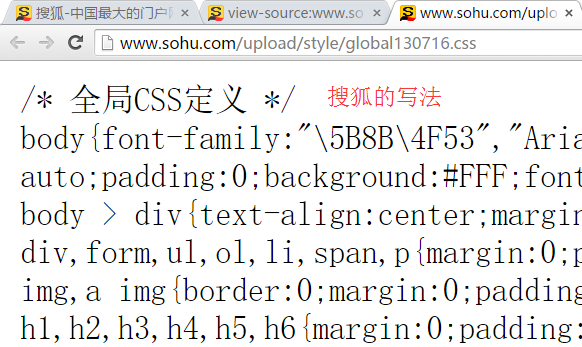
1. CSS Unicode字体

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。

方案一： 你可以使用英文来替代。 比如 font-family:"Microsoft Yahei"。

方案二： 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”。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **字体名称** | **英文名称** | **Unicode 编码** |  |  |
| 宋体 | SimSun | \5B8B\4F53 |  |  |
| 新宋体 | NSimSun | \65B0\5B8B\4F53 |  |  |
| 黑体 | SimHei | \9ED1\4F53 |  |  |
| 微软雅黑 | Microsoft YaHei | \5FAE\8F6F\96C5\9ED1 |  |  |
| 楷体\_GB2312 | KaiTi\_GB2312 | \6977\4F53\_GB2312 |  |  |
| 隶书 | LiSu | \96B6\4E66 |  |  |
| 幼园 | YouYuan | \5E7C\5706 |  |  |
| 华文细黑 | STXihei | \534E\6587\7EC6\9ED1 |  |  |
| 细明体 | MingLiU | \7EC6\660E\4F53 |  |  |
| 新细明体 | PMingLiU | \65B0\7EC6\660E\4F53 |  |  |

注：为了照顾不同电脑的字体安装问题，我们尽量只使用宋体和微软雅黑中文字体。

4、font-weight:字体粗细

字体加粗除了用 b 和 strong 标签之外，可以使用CSS 来实现，但是CSS 是没有语义的(从字面上看不出什么意思，但是能实现更多的样式，如粗细度的调节)。

font-weight属性用于定义字体的粗细，其可用属性值：normal、bold、bolder、lighter、100~900（100的整数倍）。

数字 400 等价于 normal，而 700 等价于 bold。 但是我们更喜欢用数字来表示。

注：<span></span>一个普通的、没有语义的盒子，可以用来写样式，可以有较多的显示效果。

5、font-style:字体风格

字体倾斜除了用 i 和 em 标签之外，可以使用CSS 来实现，但是CSS 是没有语义的。

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：

normal：默认值，浏览器会显示标准的字体样式。

italic：浏览器会显示斜体的字体样式。

oblique：浏览器会显示倾斜的字体样式(跟italic似乎没有什么区别，比较少用)。

平时我们很少给文字加斜体，反而喜欢给斜体标签（em，i）改为普通模式。

em {

font-style: normal;

}

6、font:综合设置字体样式 (重点)

font属性用于对字体样式进行综合设置，其基本语法格式如下：

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

使用font属性时，必须按上面语法格式中的**按顺序书写，不能更换顺序**，各个属性以空格隔开。不按着顺序写无法实现效果。

注意：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。

五、CSS注释

CSS规则是使用 /\* 需要注释的内容 \*/ 进行注释的，即在需要注释的内容前使用 “/\*” 标记开始注释，在内容的结尾使用 “\*/”结束。

七、开发者工具（chrome）

此工具是我们的必备工具，以后代码出了问题，我们首先第一反应就是：

“按F12”或者是 “shift+ctrl+i” 打开 开发者工具。

菜单： 右击网页空白出---查看

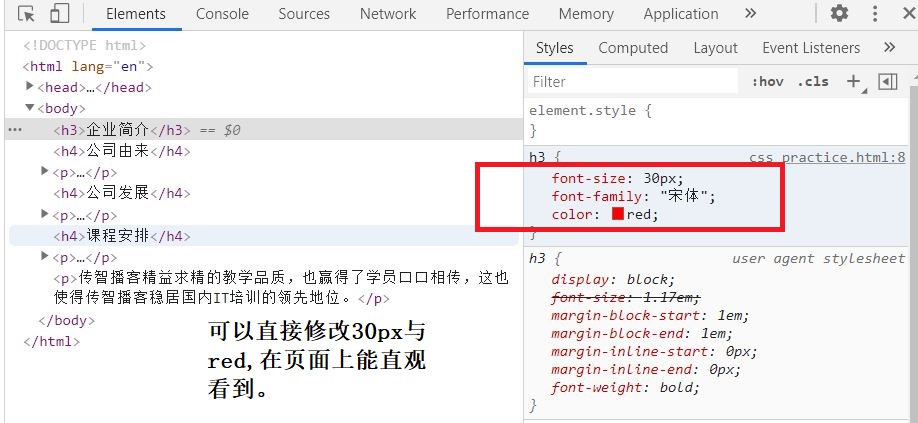
1、ctrl+滚轮 可以 放大开发者工具代码大小。

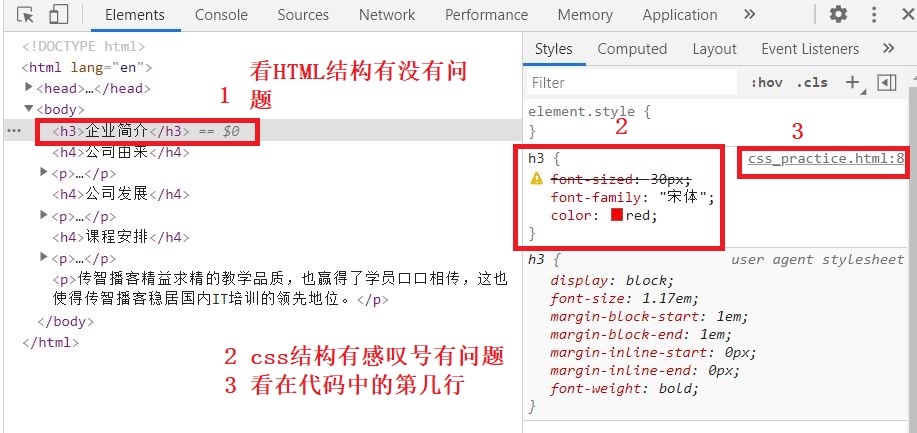
2、左边是HTML元素结构 右边是CSS样式。

3、右边CSS样式可以改动数值和颜色查看更改后效果。

调试步骤：

1. 先检查左边的HTML结构。
2. 检查右边的CSS样式，是否有出现好我们设置的值。在CSS会显示在第几行出现了问题。
3. 可以在右边实时更改数值和颜色查看更改后的效果，再将数值复制到html代码中。如字体的大小可以通过键盘的上下键来更改。





1. 选择器（重点）

1、标签选择器（元素选择器）

标签选择器是指用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。其基本语法格式如下：

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

元素名{属性1:属性值1; 属性2:属性值2; 属性3:属性值3; }

标签选择器最大的**优点**是能快速为页面中同类型的标签统一样式，同时这也是他的**缺点，不能设计差异化样式。**

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，其基本语法格式如下：

.类名{属性1:属性值1; 属性2:属性值2; 属性3:属性值3; }

标签调用的时候用 class=“类名” 即可。

类选择器**最大的优势**是可以为元素对象定义单独或相同的样式。

<style type="text/css">

.mingren {/\*声明类样式\*/

color: orange;

}

</style>

<div class="mingren">鸣人</div><!-- 使用class引用类样式 -->

1.长名称或词组可以使用中横线来为选择器命名。如：Hi-hello

2.不建议使用“\_”下划线来命名CSS选择器。浏览器兼容问题 (比如使用\_tips的选择器命名，在IE6是无效的)

3.不要纯数字、中文等命名，尽量使用英文字母来表示。中文有时候能显示，为了兼容性还是尽量用英文。在chrome能显示。

.卡卡西 {

color: deeppink;

}

3、多类名选择器

我们可以给标签指定多个类名，从而达到更多的选择目的。搭配比较自由。

<span class="red font50 font25">o</span>

注：

(1). 样式显示效果跟HTML元素中的**类名先后顺序没有关系，**受**CSS样式书写的上下顺序有关**。

(2). 各个类名中间用空格隔开。

4、id选择器

id选择器使用**“#”**进行标识，后面紧跟id名，其基本语法格式如下：

#id名{属性1:属性值1; 属性2:属性值2; 属性3:属性值3; }

该语法中，id名即为HTML元素的id属性值，大多数HTML元素都可以定义id属性，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选择器和类选择器结合记忆比较好。

#fontWeight {

font-weight: bold;

}

<p id="fontWeight">temp2</p>

5、id选择器和类选择器区别

W3C标准规定，在同一个页面内，不允许有相同名字的id对象出现，但是允许相同名字的class。

类选择器（class）好比人的名字，是可以**多次重复使用**的， 比如 张伟 王伟 李伟 李娜

id选择器好比人的身份证号码，全中国是唯一的，不得重复。**只能使用一次**。

id选择器和类选择器最大的不同在于使用次数上。

6、通配符选择器

通配符选择器用**“\*”**号表示，他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。

例如下面的代码，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\*/

}

九、伪类选择器

伪类选择器用于向某些(特殊)选择器添加特殊的效果。比如给链接添加特殊效果， 比如可以选择 第1个，第n个元素。

为了和我们刚才学的类选择器相区别， 类选择器是一个点 比如 .demo {} 而我们的伪类 用 2个点 就是 冒号 比如 :link{}

1、链接伪类选择器

a:link /\* 未访问的链接 \*/

a :visited /\* 已访问的链接 我们已经点击过一次\*/

a :hover /\* 鼠标移动到链接上 \*/

a- :active /\* 选定的链接 当我们点击的时候别松开鼠标，此时显示的状态\*/

注意写的时候，他们的顺序不能颠倒。按照 lvha 的顺序。 lv 包包 非常 hao

例子：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CSS练习</title>

<style type="text/css">

a:link {

font-size: 20px;

color: gray;

}

a:visited {

font-size: 40px;

color: blue;

}

a:hover {

font-size: 15px;

color: red;

}

a:active {

font-weight: bold;

color: green;

}

</style>

</head>

<body>

<div>

<a href="http://www.baidu.com" target="\_blank">秒杀</a>

<a href="http://www.baidu.com" target="\_blank">闪购</a>

</div>

</body>

</html>

想要达到的现象：打开页面，链接显示灰色，点击打开链接显示后显示蓝色，重新刷新界面链接又变成灰色，鼠标停留在界面上时呈现红色。

(1)在 IE 浏览器能达到效果。

**(2)在 chrome 中，如果访问历史记录中有该记录，则会一直显示蓝色，重新打开也是一样，刷新也还是蓝色，除非把纪录清除后，就能看到效果。后续研究这个问题怎么解决？？？**<a href=""或者"#" target="\_blank">闪购</a>都会有问题。在IE则ok。

实际工作中，我们简单写链接伪元素好了。

a { /\* a是标签选择器 所有的链接 \*/

font-weight: 700;

font-size: 16px;

color: gray;

}

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 \*/

}

2、结构(位置)伪类选择器（CSS3)

一般在 li 标签中比较常用。其他标签也可以进行使用。

- :first-child :选取属于其**父元素**的首个子元素的指定选择器(意思是**选择父元素中的第一个子元素，它是一个这样子的选择器**)

- :last-child :选取属于其父元素的最后一个子元素的指定选择器

- :nth-child(n) ： 匹配属于其父元素的第 N 个子元素，不论元素的类型，从第一个元素开始的。从上往下。

- :**nth-child(odd / even / n)：**可以选择所有的odd 奇数/ even 偶数的孩子标签。如果直接写 n 就会选中所有的孩子标签，会从0开始。如果是 2n 就会选中 2、4、6、8、10偶数。如果是 2n+1会选中所有奇数 1、3、5......以此类推。

- :nth-last-child(n) ：选择器匹配属于其元素的第 N 个子元素的每个元素，不论元素的类型，从最后一个子元素开始数，就是说最后一个子元素的位置是0，当作头部处理。从下往上。

n 可以是数字、关键词或公式

<style type="text/css">

input:last-child {

font-weight: 700;

color: red;

}

</style>

-----------------------------------------------------------------

<div>

<p>111</p>

<p>222</p>

<p>333</p>

<input type="button" value="hello">

</div>

<div>

<input type="button" value="hi">

</div>

效果：



3、目标伪类选择器(CSS3)

:target目标伪类选择器 :选择器可用于选取当前活动的目标元素。

当前谁被处于选中的状态。目前在测试锚点时候有使用到。 其它似乎没有用到

:target {

color: red;

}

---------------------------------------------

<body>

<a href="#test">目标伪类选择器</a>

<p id="test">目标伪类选择器测试</p>

</body>

十、CSS外观属性

1、color:文本颜色

color属性用于定义文本的颜色，其取值方式有如下3种：

(1).预定义的颜色值，如red，green，blue等。

(2).十六进制，如#FF0000，#FF6600，#29D794等。实际工作中，十六进制是最常用的定义颜色的方式。颜色在特定情况下，可以进行简写，如#ffffff #fff、#ff00ff #f0f、#ff1123这样子的就不能够进行简写。

(3).RGB代码，如红色可以表示为rgb(255,0,0)或rgb(100%,0%,0%)。

需要注意的是，如果使用RGB代码的百分比颜色值，取值为0时也不能省略百分号，必须**写为0%**。

2、line-height:行间距

ine-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或者8像素左右就可以了。浏览器默认是16px；设置25px差不多。较美观。

3、text-align:水平对齐方式

text-align属性用于设置文本内容的水平对齐，相当于html中的align对齐属性。其可用属性值如下：

left：左对齐（默认值）

right：右对齐

center：居中对齐

注：还有其他的属性看手册。

4、text-indent:首行缩进

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。

5、letter-spacing:字间距

letter-spacing属性用于定义**字间距，对英文字母也是有效的**，所谓字间距就是字符与字符之间的空白。其属性值可为不同单位的数值，允许使用负值，默认为normal。

6、word-spacing:单词间距

word-spacing属性用于定义**英文单词**之间的间距，对中文字符无效。和letter-spacing一样，其属性值可为不同单位的数值，允许使用负值，默认为normal。

word-spacing和letter-spacing均可对英文进行设置。不同的是letter-spacing定义的为字母之间的间距，而word-spacing定义的为英文单词之间的间距。

7、颜色半透明(css3)

文字颜色到了CSS3我们可以采取半透明的格式了语法格式如下：

color: rgba(r,g,b,a) ：a是alpha透明的意思 取值范围0~1之间（越接近0越透明）如：color: rgba(0,0,0,0.3) 。

8、文字阴影(CSS3)

以后我们可以给我们的文字添加阴影效果了 Shadow影子。

text-shadow:水平位置 垂直位置 模糊距离 阴影颜色;实际效果可以在浏览器F12进行动态调试，看到现象。



注：前两项是必须写的。后两项可以选写。

十一、引入CSS样式表（书写位置）

CSS可以写到那个位置？ 是不是一定写到html文件里面呢？

1、行内式（内联样式）

内联样式，又有人称行内样式、行间样式、内嵌样式。是通过标签的style属性来设置元素的样式，其基本语法格式如下：

<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>

语法中style是标签的属性，实际上任何HTML标签都拥有style属性，用来设置行内式。其中属性和值的书写规范与CSS样式规则相同，行内式只对其所在的标签及嵌套在其中的子标签起作用。

<div style="color: red;font-size: 52px;">

<p>父元素有内联样式</p>

</div>

<div>

<p>无样式</p>

</div>



用途：修改一小点样式的时候，可以用过内联样式表快速完成。

2、内部样式表

内嵌式是将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，其基本语法格式如下：

<head>

<style type="text/CSS">

选择器 {属性1:属性值1; 属性2:属性值2; 属性3:属性值3;}

</style>

</head>

语法中，style标签一般位于head标签中title标签之后，也可以把他放在HTML文档的任何地方（<body>里边、<html>在chrome中都是有效的）。

type="text/CSS"（类型是CSS文本）在html5中可以省略，写上也比较符合规范，所以这个地方可以写也可以省略。具体也可以参照工作中的要求。

3、外部样式表（外链式）

外链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，其基本语法格式如下：

<head>

<link href="CSS文件的路径" type="text/CSS" rel="stylesheet" />

</head>

注意：link 是个单标签哦!!!

该语法中，link标签需要放在**head头部标签中**，并且**必须指定**link标签的**三个属性**，具体如下：

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。

type：定义所链接文档的类型，在这里需要指定为“text/CSS”，表示链接的外部文件为CSS样式表。

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。

十二、三种样式表总结

| **样式表** | **优点** | **缺点** | **使用情况** | **控制范围** |
| --- | --- | --- | --- | --- |
| 行内样式表 | 书写方便，权重高 | 没有实现样式和结构相分离 | 较少 | 控制一个标签（少） |
| 内部样式表 | 部分结构和样式相分离 | 没有彻底分离 | 较多 | 控制一个页面（中） |
| 外部样式表 | 完全实现结构和样式相分离 | 需要引入 | 最多，强烈推荐 | 控制整个站点（多） |

1. 标签显示模式（display）

标签的类型(显示模式)

HTML标签一般分为**块标签**和**行内标签**两种类型，它们也称块元素和行内元素。

1、块级元素(block-level)

每个块元素通常都会独自占据一整行或多整行，可以对其设置宽度、高度、对齐等属性，常用于网页布局和网页结构的搭建。

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

块级元素的特点：(掌握)

（1）总是从新行开始

（2）高度，行高、外边距以及内边距都可以控制。

（3）宽度默认是容器的100%，高度默认跟内容高度适配。

（4）可以容纳内联元素和其他块元素。

2、行内元素(inline-level)

行内元素（内联元素）不占有独立的区域，仅仅靠自身的字体大小和图像尺寸来支撑结构，一般不可以设置宽度、高度、对齐等属性，常用于控制页面中文本的样式。

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

行内元素的特点：

（1）和相邻行内元素在一行上。

（2）高、宽无效，但水平方向的padding和margin可以设置，垂直方向的无效。

（3）默认宽度就是它本身内容的宽度。

（4）行内元素只能容纳文本或则其他行内元素。（a特殊）

注意：

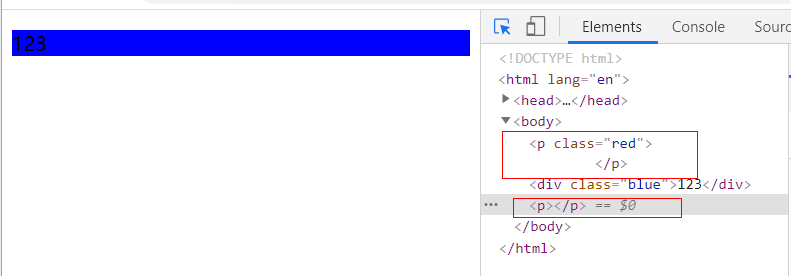
(1). 只有文字才能组成段落因此p里面不能放块级元素，同理还有这些标签h1、h2、h3、h4、h5、h6、dt，他们都是文字类块级标签，里面不能放其他块级元素。

p里边放块级标签会解析出错，在F12中才能看到。

<p class="red">

<div class="blue">123</div>

</p>



(2). 链接里面不能再放链接。这样子你点击的时候不知道要去往哪一个链接。

<a href="123">123<a href="456">456</a></a>



3、行内块元素（inline-block）

在行内元素中有几个特殊的标签——<img />、<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：

（1）和相邻行内元素（行内块）在一行上,但是之间会有空白缝隙。

（2）默认宽度就是它本身内容的宽度。

（3）高度，行高、外边距以及内边距都可以控制。

简单说就是它们可以放在一行又能设置高度、宽度等等。

可以在一行放多个，有宽度和高度，盒子的大小是由内容决定的。

十三、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;

块、行内标签转换为行内快，都会在一行上。

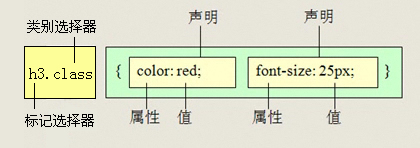
此阶段，我们只需关心这三个，其他的是我们后面的工作。

十四、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,目的是为了可以选择更准确更精细的目标元素标签。

1、交集选择器

交集选择器由两个选择器构成，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。



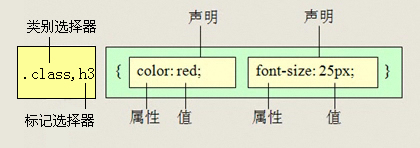
交集选择器是并且的意思。 即...又...的意思

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，不太建议使用。

2、并集选择器

并集选择器（CSS选择器分组）是各个选择器通过 **逗号** 连接而成的，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。如果某些选择器定义的**样式完全相同，或部分相同**，就可以利用并集选择器为它们定义相同的CSS样式。



逗号

并集选择器 和 的意思，就是说，只要逗号隔开的，所有选择器都会执行后面样式。

比如 .one, p , #test {

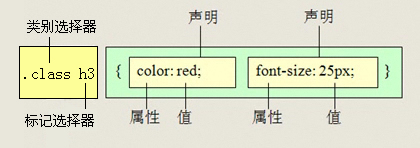
color: #F00 ;

}

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。

3、后代选择器

后代选择器又称为包含选择器，用来选择元素或元素组的后代，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。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。

<!-- 要求让上边的三个变颜色，下边不变颜色 -->

<div class="nav">

<ul>

<li>百度</li>

<li>搜狐</li>

<li>谷歌</li>

</ul>

</div>

<hr />

<div>

<ul>

<li>百度</li>

<li>搜狐</li>

<li>谷歌</li>

</ul>

</div>

<style type="text/css">

.nav ul li {

color: #ff0000;

}

如果是写为 div ul li 那么下边的也会变颜色

写成 .nav li也能变颜色，但这样不太规范

</style>

4、子元素选择器

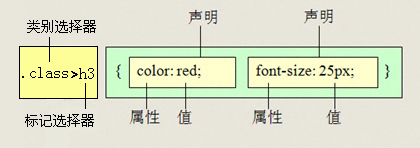
子元素选择器只能选择作为某元素子元素的元素。其写法就是把父级标签写在前面，子级标签写在后面，中间跟一个 **>** 进行连接，注意，符号左右两侧各保留一个空格。

这里的子 指的是 亲儿子 不包含孙子 重孙子之类。

比如： .demo > h3 {

color: red;

} 说明 h3 一定是demo 亲儿子。demo 元素包含着h3。



<body>

<!-- 要求让所有的一级菜单变颜色 二级菜单不变-->

<ul class="item">

<li>一级菜单

<ul>

<li>二级菜单</li>

<li>二级菜单</li>

</ul>

</li>

<li>一级菜单

<ul>

<li>二级菜单</li>

<li>二级菜单</li>

</ul>

</li>

</ul>

</body>

<style type="text/css">

.item li {

color: #f00

}/\*如果这个样式去除的话，那么所有的菜单都会变成蓝色。我个人理解这样子出现的原因是这样的：它一级菜单的颜色先确定下来，是被设置为蓝色，然后二级菜单是继承了一级菜单的颜色，因为二级菜单并没有设置颜色，所以出现了蓝色。

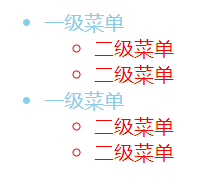
如果这个样式不去除，先继承蓝色，然后发现 .item li是后代选择器，二级菜单属于该范围，被设置为了红色，所以它有自己的颜色\*/

.item > li {

color: skyblue;

}

</style>



5、测试题

<style type="text/css">

/\*.site-r > a {

color: #f00;

font:normal 400 14px "microsoft yahei";

}

.nav a {

color: blue;

font:normal 400 14px "microsoft yahei";

}

.site-l {

font:normal 400 14px "microsoft yahei";

}

.nav>ul>li>a {

color: green;

}\*/

.site-r a {

color: #f00;

}

.nav ul li a {

color: skyblue;

}

/\*字体也是可以继承的\*/

.sitenav, .nav{

font: normal 400 14px "microsoft yahei";

}

.nav > ul > li > a {

color: green;

}

</style>

<body>

<!-- 在不修改以上代码的前提下，完成以下任务：

1. 链接 登录 的颜色为红色,同时主导航栏里面的所有的链接改为蓝色 (简单)

2. 主导航栏和侧导航栏里面文字都是14像素并且是微软雅黑。（中等)

3. 主导航栏里面的一级菜单链接文字颜色为绿色。（难)

-->

<div class="nav"> <!-- 主导航栏 -->

<ul>

<li><a href="#">公司首页</a></li>

<li><a href="#">公司简介</a></li>

<li><a href="#">公司产品</a></li>

<li><a href="#">联系我们</a>

<ul>

<li><a href="#">公司邮箱</a></li>

<li><a href="#">公司电话</a></li>

</ul>

</li>

</ul>

</div>

<div class="sitenav"> <!-- 侧导航栏 -->

<div class="site-l">左侧侧导航栏</div>

<div class="site-r"><a href="#">登录</a></div>

</div>

</body>

6、属性选择器

选取标签带有某些特殊属性的选择器（<a href=”” title=”xx”>xx</a>，title是标签a的一个属性，class也可以，href试了一下不行？？？），我们成为属性选择器。

|  |  |
| --- | --- |
| 选择器 | 含义 |
| E[attr] | 存在attr属性 |
| E[attr=val] | 属性值**完全等于**val |
| E[attr\*=val] | 属性值里包含val字符并且在“任意”位置 |
| E[attr^=val] | 属性值里包含val字符并且在“开始”位置 |
| E[attr$=val] | 属性值里包含val字符并且在“结束”位置 |

div[class^=**font**] { /\* class^=font 表示 font 开始位置就行了 \*/

color: pink;

}

div[class$=**footer**] { /\* class$=footer 表示 footer 结束位置就行了 \*/

color: skyblue;

}

div[class\*=**tao**] { /\* class\*=tao \*= 表示tao 在任意位置都可以 \*/

color: green;

}

<div class="**font**12">属性选择器</div>

<div class="**font**12">属性选择器</div>

<div class="**font**24">属性选择器</div>

<div class="**font**24">属性选择器</div>

<div class="**font**24">属性选择器</div>

<div class="24font">属性选择器123</div>

<div class="sub-**footer**">属性选择器footer</div>

<div class="jd-**footer**">属性选择器footer</div>

<div class="news-**tao**-nav">属性选择器</div>

<div class="news-**tao**-header">属性选择器</div>

<div class="**tao**-header">属性选择器</div>

7、元素选择器（CSS3)

（1） E::first-letter文本的第一个单词或字（如中文、日文、韩文等）

（2） E::first-line 文本第一行；

（3） E::selection 可改变选中文本的样式：即选中的文本可以根据需求更改文字的颜色或者背景色

p::first-letter {

font-size: 20px;

color: hotpink;

}

/\* 首行特殊样式 \*/

p::first-line {

color: skyblue;

}

p::selection {

/\* font-size: 50px; \*/

color: orange;

}

（4）E::before和E::after

在E元素内部的开始位置和结束位创建一个元素，该元素为行内元素，且必须要结合content属性使用。后边在盒子模型时有更多的作用，现在先理解是怎么工作的。

div::before {

content:"开始";

}

div::after {

content:"结束";

}



E:after、E:before 在旧版本里是伪元素，CSS3的规范里“:”用来表示伪类，“::”用来表示伪元素，但是在高版本浏览器下E:after、E:before会被自动识别为E::after、E::before，这样做的目的是用来做兼容处理。

E:after、E:before后面的练习中会反复用到，目前只需要有个大致了解

**":" 与 "::" 区别在于区分伪类和伪元素**

十五、CSS书写规范

开始就形成良好的书写规范，是你专业化的开始。

1、空格规范

【强制】 选择器 与 { 之间必须包含空格。

示例： .selector { }

【强制】 **属性名**与之后的**冒号**之间不允许包含空格， : 与 属性值 之间必须包含空格。

示例： font-size: 12px;

2、选择器规范

【强制】 当一个 rule 包含多个 selector 时，每个选择器声明必须独占一行。

示例：

/\* good \*/

.post,

.page,

.comment {

line-height: 1.5;

}

/\* bad \*/

.post, .page, .comment {

line-height: 1.5;

}

【建议】 选择器的嵌套层级应不大于 3 级，位置靠后的限定条件应尽可能精确。

示例：

/\* good \*/

#username input {}

.comment .avatar {}

/\* bad \*/

.page .header .login #username input {}

.comment div \* {}

3、属性规范

【强制】 属性定义必须另起一行。

示例：

/\* good \*/

.selector {

margin: 0;

padding: 0;

}

/\* bad \*/

.selector { margin: 0; padding: 0; }

【强制】 属性定义后必须以分号结尾。

示例：

/\* good \*/

.selector {

margin: 0;

}

/\* bad \*/

.selector {

margin: 0

}

十六、CSS 背景(background)

CSS 可以添加背景颜色和背景图片，以及来进行图片设置。

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **background-color** | **背景颜色** |  |  |
| background-image | 背景图片地址 |  |  |
| background-repeat | 是否平铺 |  |  |
| background-position | 背景位置 |  |  |
| background-attachment | 背景固定还是滚动 |  |  |
| 背景的合写（复合属性） |  |  |  |
| background:背景颜色 背景图片地址 背景平铺 背景滚动 背景位置 |  |  |  |

1、背景图片(image)

语法：

background-image : none | url (url)

参数：

none : 　无背景图（默认的）

url : 　使用绝对或相对地址指定背景图像。

background-image 属性允许指定一个图片展示在背景中（只有CSS3才可以多背景）可以和 background-color 连用。 如果图片不重复地话，图片覆盖不到地地方都会被背景色填充。 如果有背景图片平铺，则会覆盖背景颜色。

小技巧： 我们提倡背景图片后面的地址，**url不要加引号**。

2、背景平铺（repeat）

语法：

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

参数：

repeat : 　背景图像在纵向和横向上平铺（默认的）

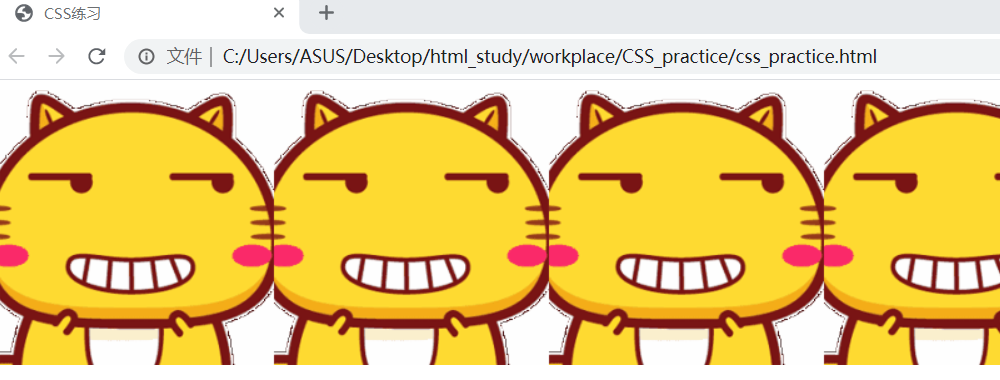
no-repeat : 　背景图像不平铺

repeat-x : 　背景图像在横向上平铺

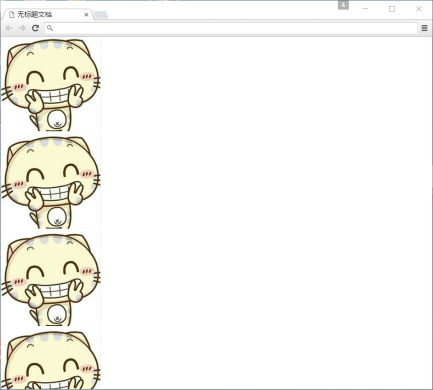
repeat-y : 　背景图像在纵向平铺

设置背景图片时，默认把图片在水平和垂直方向平铺以铺满整个元素。

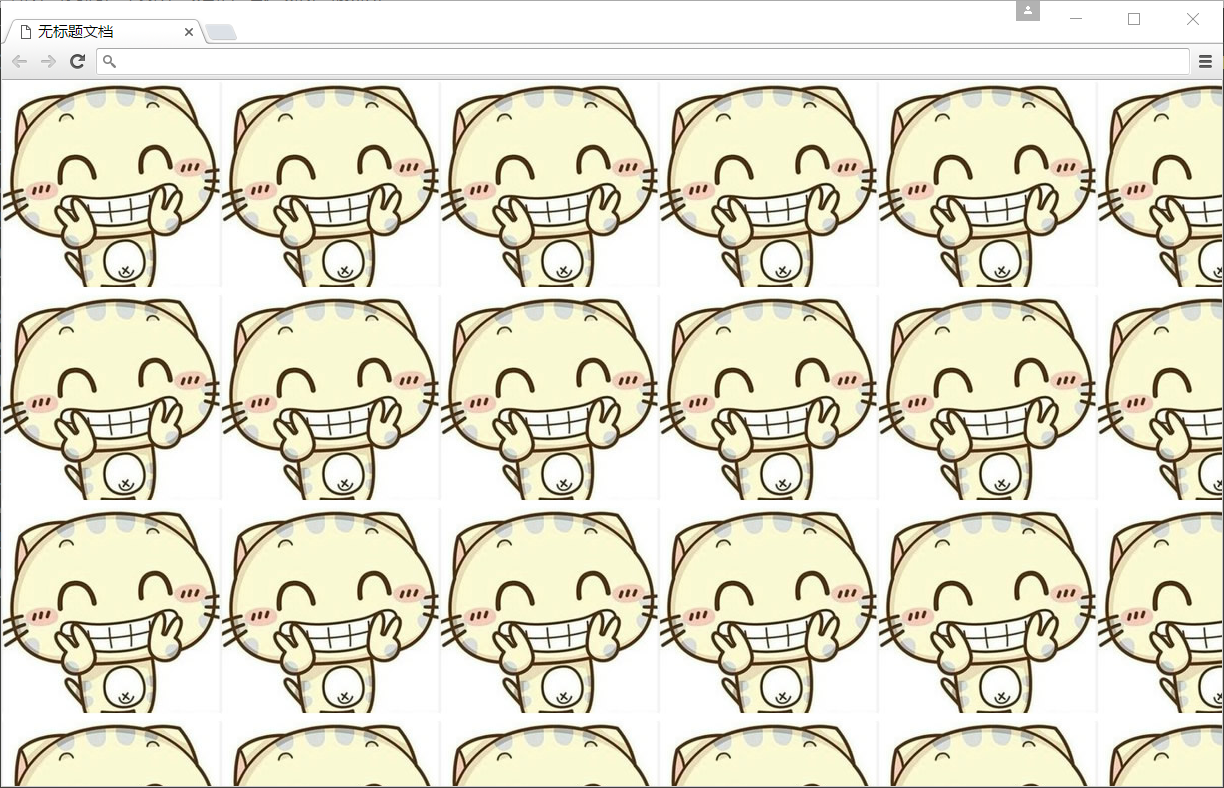
repeat-x : 　背景图像在横向上平铺



repeat-y : 　背景图像在纵向平铺



设置背景图片时，默认把图片在水平和垂直方向平铺以铺满整个元素。



3、背景位置(position)

语法：

background-position : length || length

background-position : position || position

参数：

position : 　top | center | bottom | left | center | right----->方位名词

默认值为 left top默认是在左上角，**方位名词没有顺序**，top left效果是一致的。如果方位名词只有一个，那么另外一个默认是center，如top背景图片的位置会在贴着**上边框**，**水平中心。**

**如果设置top bottom，是对立的，会被chrome判断为无效属性值，那么图片还是在原来的位置。**

说明：

设置或检索对象的背景图像位置。必须先指定background-image属性。

对于一个网页，它的坐标是这样子的。

X

Y

(1). 利用方位名词：top left等等来更改图片的位置。

(2). 精确单位：第一个值是x坐标，第二个值一定是y 坐标。负数的话，图片会往左边或上边移动，超出部分不显示。

(3). 混搭：如果和精确单位和方位名字混合使用，则必须是x坐标在前，y坐标后面。比如 background-position: 15px top; 则 15px 一定是x坐标，top是 y坐标。

注：50px top是ok。50px buttom是无效的，因为buttom的拼写是错的，是**bottom**待理解。50px right/left这样会冲突，变成无效值。

background-position:50% 50%;

图片水平和垂直居中。与 background-position:center center;效果等同。

等同于x：{容器(container)的宽度—背景图片的宽度}\*x百分比，超出的部分隐藏。

等同于y：{容器(container)的高度—背景图片的高度}\*y百分比，超出的部分隐藏。

例如： .container{

width:300px;

height:150px;

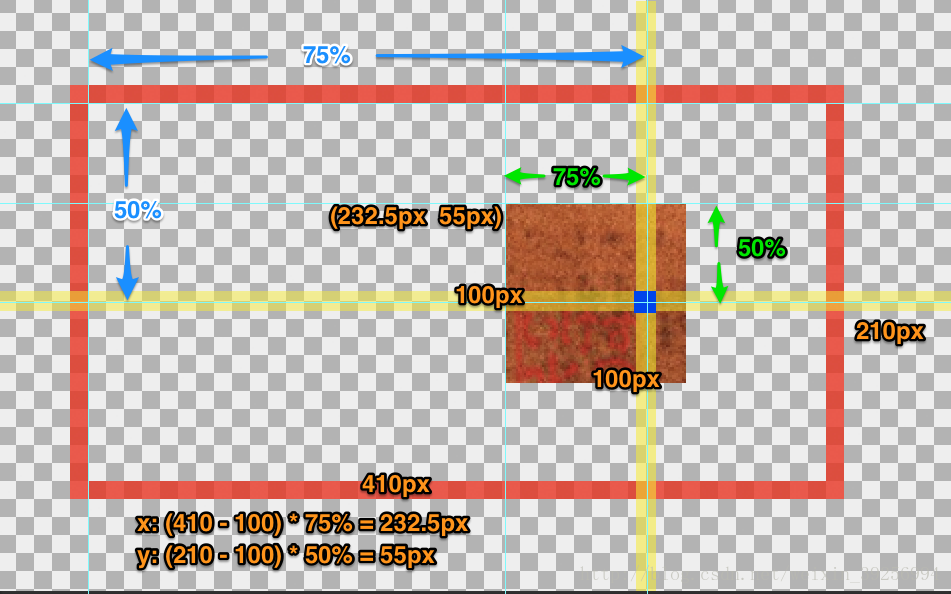
background:transparent url(bg.jpg) no-repeat scroll 50% 50%;

border:5px solid green;

}

其x=(300-210)\*50%=45px;

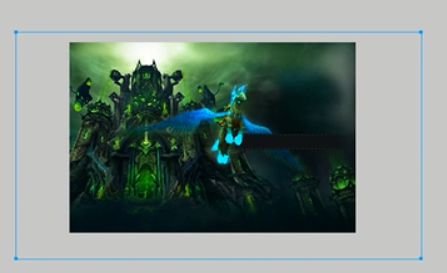
y=(150-120)\*50%=15px;



1. 用法例子：

注：在div中如果没有指定高度，直接设置背景图片是看不到效果的。后边盒子模型会研究

到？？？



当图片的大小比较大时，浏览器无法显示整个图片，可以把图片水平居中，然后垂直相对靠上边设置即可。

4、背景附着

语法：

background-attachment : scroll | fixed

参数：

scroll : 　背景图像是随对象内容滚动（默认是滚动的）

fixed : 　背景图像固定

说明：

设置或检索背景图像是随对象内容滚动还是固定的。

5、背景简写

background属性的值的书写顺序官方并没有强制标准的。为了可读性，建议大家如下写：

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

如果省略的话，就会按默认值处理，白色、xy平铺、滚动

background: transparent url(image.jpg) repeat-y scroll 50% 0 ;

6、盒子背景透明(CSS3)

CSS3支持背景半透明的写法语法格式是:

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

**最后一个参数是alpha 透明度 取值范围 0~1之间**

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。如果要改变盒子内的文字颜色使用的是color属性。

7、背景缩放(CSS3)

通过background-size设置背景图片的尺寸，就像我们设置img的尺寸一样，在移动Web开发中做屏幕适配应用非常广泛。

其参数设置如下：

1. 可以设置长度单位(px)或百分比（设置百分比时，**是占盒子的百分比，不是图片自身的百分比**，参照**盒子的宽高**）

插入的图片使用<img>标签时，直接可以通过width、height来设置图片的尺寸。但是背景图片不行。

运用在盒子的时候，当background-size 只设置一个值时，那么另外一个值也会**同比例缩小**，这样不会造成图片扭曲或者失真太多。

b) 设置为cover时，会自动调整缩放比例，保证图片始终填充满背景区域，如有溢出部分则会被隐藏。我们平时用的cover 最多。

图片会自动进行**等比例**缩放，图片一定要保证宽度和高度同时满足盒子的大小。就像把一个按住shift进行缩放，全部覆盖整个盒子，溢出的部分就隐藏，看不见了。

c) 设置为contain会自动调整缩放比例，保证图片始终完整显示在背景区域。

图片会自动进行**等比例**缩放，如果图片的高度或者宽度有一个和盒子一样大了，就不在缩放，这样就保证了图片的完整性，不会有缺失的一部分，但是盒子会铺不满。

如果图片与盒子都是正方形，则两者能吻合。

background-image: url('images/gyt.jpg');

background-size: 300px 100px;

/\* background-size: contain; \*/

/\* background-size: cover; \*/

8、多背景(CSS3)

以逗号分隔可以设置多背景，可用于自适应布局 做法就是 用逗号隔开就好了。

- 一个元素可以设置多重背景图像。

- 每组属性间使用逗号分隔。

- 如果设置的多重背景图之间存在着交集（即存在着重叠关系），**前面的**背景图会**覆盖**在**后面的**背景图之上。

- 为了避免背景色将图像盖住，背景色通常都**定义在最后一组上。**

background:url(test1.jpg) no-repeat scroll 10px 20px/50px 60px ,

url(test1.jpg) no-repeat scroll 10px 20px/70px 90px ,

url(test1.jpg) no-repeat scroll 10px 20px/110px 130px c **#aaa**;

注： background: url(images/3.jpg) no-repeat left top hotpink,

url(images/h.jpg) no-repeat right bottom ;

在chrome中是被认为是无效的。待验证？？？

9、凹凸文字

凸起文字显示的原理：文字跟背景是同样的颜色，然后通过对文字的左、下添加黑色阴影，右、上添加白色亮光，两者结合起来就显示凸起效果。



<style type="text/css">

body {

background-color: #ccc;

}

div {

font: normal bold 80px "microsoft yahei";

color: #ccc;

}

div:first-child {

**text-shadow: 3px 3px 1px #000, -3px -3px 1px #fff;**

}

div:last-child {

**text-shadow: -3px -3px 1px #000, 3px 3px 1px #fff;**

}

</style>

</head>

<body>

<div>我是凸起的文字</div>

<div>我是凹下的文字</div>

</body>

其对比度可以通过调节x,y两个的偏移来看。

10、导航栏案例

文本的装饰text-decoratio：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。 |
| underline | 定义文本下的一条线。下划线 也是我们链接自带的 |
| overline | 定义文本上的一条线。 |
| line-through | 定义穿过文本下的一条线。 |

使用技巧：在一行内的盒子内，我们设定行高等于盒子的高度，就可以使文字垂直居中。

<style type="text/css">

.bg-black {

background-color: #000;

}

a {

display: inline-block; **---->如果只是转换为块元素，则四个链接会各占一行，达不到要求；转换为行内快，既能设置高宽，又能在一行**

width: 200px;

height: 50px;

font-size: 25px;

line-height: 50px;**---->设置行的高度与盒子的高度一致，就可以使文本文字垂直居中**

text-align: center;

text-decoration: none;

color: white;

}

a:hover {

background: url(images/h.png) no-repeat;

}

</style>

<div class="bg-black">

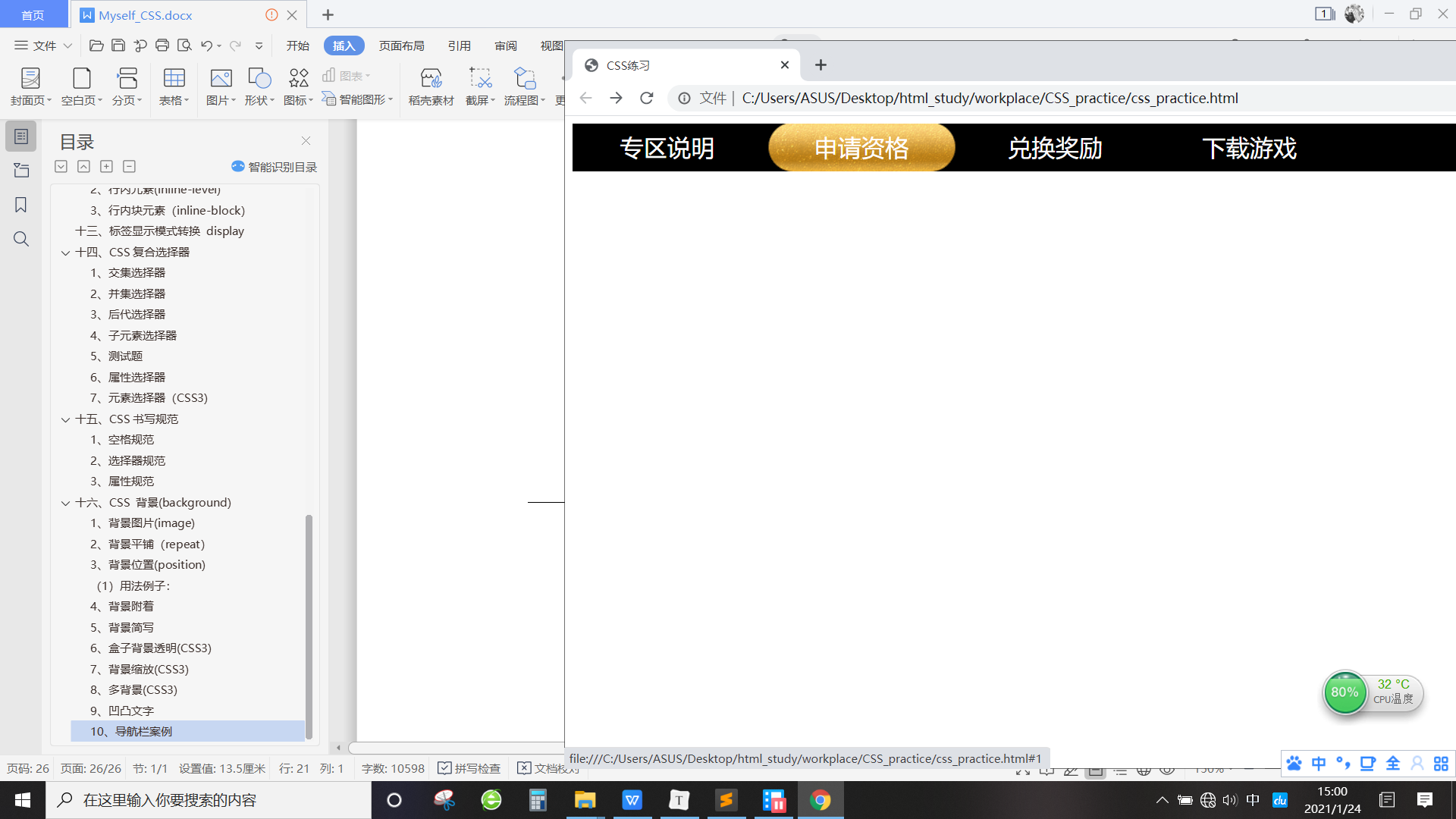
<a href="#1">专区说明</a>

<a href="#1">申请资格</a>

<a href="#1">兑换奖励</a>

<a href="#1">下载游戏</a>

</div>



十七、CSS 三大特性

**层叠、继承、优先级**是我们学习CSS 必须掌握的三个特性。

1、CSS层叠性

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。

比如先给某个标签指定了内部文字颜色为红色，接着又指定了颜色为蓝色，此时出现一个标签指定了相同样式不同值的情况，这就是样式冲突。

一般情况下，如果出现**样式冲突**，则会按照CSS**书写的顺序**，以最后的样式为准。

（1）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。

（2）样式不冲突，不会层叠。

2、CSS继承性

所谓继承性是指书写CSS样式表时，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。

简单的理解就是：子承父业，**跟字体相关的一般可以继承。**

恰当地使用继承可以简化代码，降低CSS样式的复杂性。子元素可以继承父元素的样式（text-，font-，line-这些元素开头的都可以继承，以及color属性）

但是如果在网页中所有的元素都大量继承样式，那么判断样式的来源就会很困难，所以对于字体、文本属性等网页中的通用样式可以使用继承。例如字体、字号、颜色、行距等都可以在body元素中统一设置，然后通过继承影响文档中的所有文本。

并不是所有CSS属性都可以继承，例如下面的属性就不具有继承性：**边框、外边距、内边距、背景、定位、元素高度属性等等**。（**宽度是可以继承，默认是父容器的100%，块级元素继承块级元素是ok的，但是继承人如果是行内块元素、行内元素就无法）**

3、CSS优先级

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，这时就会出现优先级的问题。

在考虑权重时，初学者还需要注意一些特殊的情况，具体如下：

**继承样式的权重为0。**即在嵌套结构中，**不管父元素样式的权重多大（即使在父元素颜色后边加上！Important，继承后也是0）**，被子元素继承时，他的权重都为0，也就是说子元素定义的样式会覆盖继承来的样式。

行内样式优先。应用style属性的元素，其行内样式的权重非常高，可以理解为远大于100。总之，他拥有比上面提高的选择器都大的优先级。

权重相同时，CSS遵循就近原则。也就是说靠近元素的样式具有最大的优先级，或者说排在最后的样式优先级最大。

CSS定义了一个!important命令，该命令被赋予最大的优先级。也就是说不管权重如何以及样式位置的远近，!important都具有最大优先级。

（1）CSS特殊性（Specificity）

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，我们称为CSS 特性或称非凡性，它是一个衡量CSS值优先级的一个标准 具体规范入如下：

specificity用一个四位的数 字串(CSS2是三位)来表示，更像四个级别，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。

|  |  |
| --- | --- |
| 继承或者\* 的贡献值 | 0,0,0,0 |
| 每个元素（标签）贡献值为 | 0,0,0,1 |
| 每个类，伪类贡献值为 | 0,0,1,0 |
| 每个ID贡献值为 | 0,1,0,0 |
| 每个行内样式贡献值 | 1,0,0,0 |
| 每个!important贡献值 | ∞ 无穷大 |

注：伪类选择器指的是---> “:first-child”，而不是“a:first-child”，它们的权重是不一样的，后者叠加了。

权重是可以叠加的

比如的例子：

|  |  |
| --- | --- |
| 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 |
| #nav p | 0,1,0,1 |

注意：

（1）数位之间没有进制 比如说： 0,0,0,5 + 0,0,0,5 =0,0,0,10 而不是 0, 0, 1, 0， 所以不会存在10个div能赶上一个类选择器的情况。

（2）继承的权重是0

<style type="text/css">

.red {

color: red;

}

p {

color: blue;

}

</style>

<div class="red">

<p>hello world!</p>

</div>

文字的颜色显示为蓝色。.red它是属于div，对于p来说是继承的，优先级为0。

（2）总结优先级：

1. 使用了 !important声明的规则。

2. 内嵌在 HTML 元素的 style属性里面的声明。

3. 使用了 ID 选择器的规则。

4. 使用了类选择器、属性选择器、伪元素和伪类选择器的规则。

5. 使用了元素选择器的规则。

6. 只包含一个通用选择器的规则。

7. 同一类选择器则遵循就近原则。

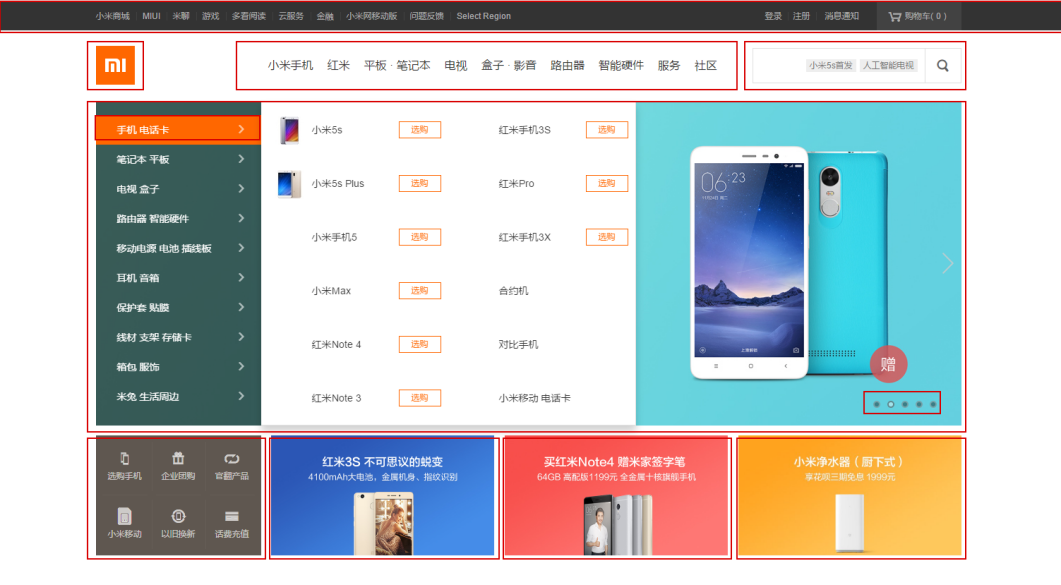
总结：权重是优先级的算法，层叠是优先级的表现

十八、盒子模型（CSS重点）

其实，CSS就三个大模块：**盒子模型**、**浮动**、**定位**。其余的都是细节。要求这三部分，无论如何也要学的**非常精通**。

所谓盒子模型就是把HTML页面中的元素看作是一个矩形的盒子，也就是一个盛装内容的容器。每个矩形都由元素的内容、内边距（padding）、边框（border）和外边距（margin）组成。

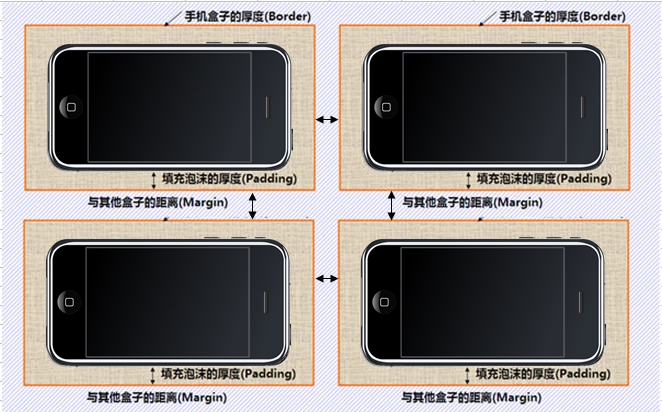
看透网页布局的本质：**把网页元素比如文字图片等等，放入盒子里面，然后利用CSS摆放盒子的过程，就是网页布局。**



1、盒子模型（Box Model）

这里略过 老旧的ie盒子模型（IE6以下），对不起，我都没见过IE5的浏览器。

首先，我们来看一张图，来体会下什么是盒子模型。



2、盒子边框（border）

边框就是那层皮。

语法：

border : border-width || **border-style** || border-color 没有样式就显示不出来，如果 单有样式，其他两个会有默认值。

边框属性—设置边框样式（border-style），边框样式用于定义页面中边框的风格，常用属性值如下：

none：**没有边框即忽略所有边框的宽度（默认值）**--- 如果这个值不存在，那么颜色与宽度的存在就没有意义了。

solid：边框为单实线(最为常用的)

dashed：边框为虚线

dotted：边框为点线

double：边框为双实线

3、盒子边框写法总结表

|  |  |  |
| --- | --- | --- |
| 设置内容 | 样式属性 | 常用属性值 |
| 上边框 | border-top-style:样式; border-top-width:宽度;border-top-color:颜色;border-top:宽度 样式 颜色; |  |
| 下边框 | border-bottom-style:样式;border- bottom-width:宽度;border- bottom-color:颜色;border-bottom:宽度 样式 颜色; |  |
| 左边框 | border-left-style:样式; border-left-width:宽度;border-left-color:颜色;border-left:宽度 样式 颜色; |  |
| 右边框 | border-right-style:样式;border-right-width:宽度;border-right-color:颜色;border-right:宽度 样式 颜色; |  |
| 样式综合设置 | border-style:上边 [右边 下边 左边]; | none无（默认）、solid单实线、dashed虚线、dotted点线、double双实线 |
| 宽度综合设置 | border-width:上边 [右边 下边 左边]; | 像素值 |
| 颜色综合设置 | border-color:上边 [右边 下边 左边]; | 颜色值、#十六进制、rgb(r,g,b)、rgb(r%,g%,b%) |
| 边框综合设置 | border:四边宽度 四边样式 四边颜色; |  |

3、表格的细线边框

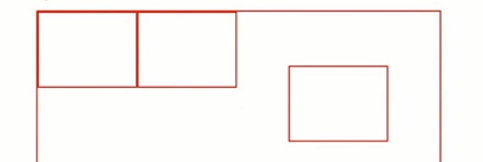
以前学过的html表格边框很粗，这里只需要CSS一句话就可以美观起来。

**table {**

**border-collapse:collapse; }**  collapse 单词是合并的意思

border-collapse:collapse; 表示边框合并在一起。

注：当设置了单元格之间无间距、内容与单元格边界无间距时。由于单元格有宽度，它们不会叠加在一起，堆在一起时就会偏大，边框相对来说会增大。



例子：

table {

height: 100px;

width: 200px;

border: 3px solid red;

}

td {

border: 3px solid red;

}

<table cellspacing="0" cellpadding="0">

<tbody>

<tr>

<td>1</td>

<td>1</td>

</tr>

</tbody>

</table>

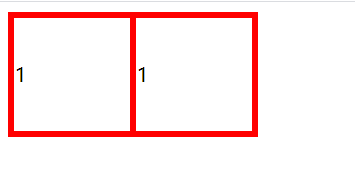


table {

height: 100px;

width: 200px;

border: 3px solid red;

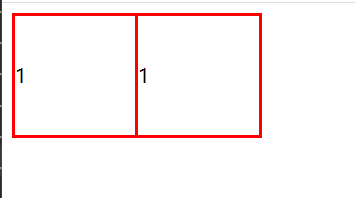
border-collapse: collapse;/\*合并之后，边框就是设置的大小了\*/

}

td {

border: 3px solid red;

}



1. 使用table选择器定义的样式与table中的border属性的效果是不一样的是针对整个 table，所以在 table 上设置了 border 属性，table 中的每一个 td都会有边框。

css 或者说 style，是只针对单一标签的（**border属性是无法被继承的**），所以如果仅仅在 table 标签设定 css 的 border 的话，就只有 table 有边框，里面的 td是没有边框的

<table border="1" height="100" width="200">

<tbody>

<tr>

<td>1</td>

<td>1</td>

</tr>

</tbody>

</table>

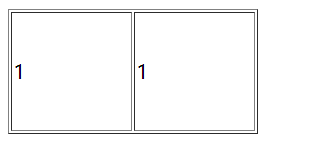
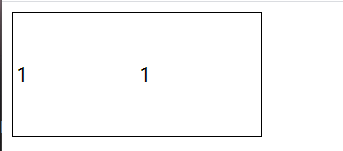


table {

border: 1px solid;

}

<table height="100" width="200">



4、圆角边框(CSS3)

语法格式：

border-radius: 左上角 右上角 右下角 左下角;不用的数量的值会有不一样的效果，具体看例子。

例子:

div {

width: 200px;

height: 200px;

border: 1px solid red;

}

div:first-child { **/\* 结构伪类选择器 \*/**

border-radius: 10px; /\*一个数值表示**4个角**都是相同的 10px 的弧度\*/

}

div:nth-child(2) {

border-radius: 50%; /\* 100px或者50%取宽度和高度一半则会变成一个圆形 \*/

}

div:nth-child(3) {

border-radius: 10px 40px; /\* 左上角和右下角是10px 。右上角和左下角 是40px，呈现对角线 \*/

}

div:nth-child(4) {

border-radius: 10px 40px 80px; /\* 左上角10；右上角和左下角40；右下角80 \*/

}

div:nth-child(5) {

border-radius: 10px 40px 80px 100px; /\* 左上角10；右上角40；右下角 80；左下角100 \*/

}

div:nth-child(6) {

border-radius: 100px;

height: 100px;

}

div:nth-child(7) {

border-radius: 100px 0;

}

</style>

5、内边距（padding）

**内边距的使用自动会撑大盒子。使得盒子看起来比较大。**

padding属性用于设置内边距。 是指**边框与内容**之间的距离。

padding-top:上内边距

padding-right:右内边距

padding-bottom:下内边距

padding-left:左内边距

|  |  |
| --- | --- |
| 值的个数 | 表达意思 |
| 1 | 上下左右内边距。比如padding: 3px; 表示：上下左右都是3像素。 |
| 2 | 上下、左右内边距。比如 padding: 3px 5px; 表示：上下3像素、左右 5像素。 |
| 3 | 上、左右、下内边距 比如 padding: 3px 5px 10px; 表示：上是3像素、左右是5像素、下是10像素。 |
| 4 | 上、右、下、左内边距 比如: padding: 3px 5px 10px 15px; 表示：上3px、右是5px、下 10px、左15px。顺时针。 |

Firework测量工具的使用，切片工具？？？nav标签是啥？？？

6、新浪导航栏练习

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CSS练习</title>

<style type="text/css">

nav {

height: 41px;

background-color: #FCFCFC;

border-top: 3px solid #FF8500;

border-bottom: 1px solid #EDEEF0;

}

a {

display: inline-block;

color: #4c4c4c;

font-size: 14px;

height: 41px;

**/\*width: 100px;如果限定了宽度，那么每个链接的字数不一样，链接之间的间距就大小不一，参差不齐，不美观。给定**

**边内距后，它会自动去撑开盒子。链接之间的间距一致美观，达到要求\*/**

/\*padding-left: 20px;

padding-right: 20px;\*/

padding: 0 20px;

text-align: center;

line-height: 41px;

text-decoration: none;

}

a:hover {

background-color: orange;

}

</style>

</head>

<body>

<nav>

<a href="#2">首页</a>

<a href="#2">手机新浪网</a>

<a href="#2">土豆网</a>

<a href="#2">优酷网</a>

</nav>

</body>

</html>

7、外边距（margin）

margin属性用于设置外边距。设置外边距会在元素之间创建“空白”，这段空白通常不能放置其他内容。

margin-top:上外边距

margin-right:右外边距

margin-bottom:下外边距

margin-left:上外边距

margin:上外边距 右外边距 下外边距 左外边

**取值顺序跟内边距相同。**

**注**：上、左外边距容易看出效果，但是右、下可能看不出，因为它们本来到浏览器边框距离很大，可以通过F12中的元素看到各个边距是多少。

（1）外边距实现盒子居中

可以让一个盒子实现水平居中，需要满足一下两个条件：

1. 必须是**块级元素**。

2. 盒子必须**指定了宽度（width），没有指定，没有效果**

然后就给左右的外边距都设置为**auto**，就可使块级元素**水平**居中。

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;}

1. 文字、盒子居中

1. 文字水平居中是 text-align: center

2. 盒子水平居中 左右margin 改为 auto

text-align: center; /\* 文字居中水平 \*/

margin: 10px auto; /\* 盒子水平居中 左右margin 改为 auto 就阔以了 \*/

1. 插入图片<img>和背景图片background区别
2. 插入图片，我们用的最多比如产品展示类

改变插入图片的位置，可以通过图片设置外边距margin或者设置内边距padding属性，只设置一个，两者能达到同样的效果。**img是属于行内块元素，图片相当于是内容文字。**

2. 背景图片，我们一般用于小图标背景 或者 超大背景图片。

|  |
| --- |
| section {  width: 300px; /\* 插入图片更改大小使用的是 width 和 height \*/  height: 300px;  border: 1px solid red;  /\*padding-top: 30px;  padding-left: 30px;\*/  }  section img {  width: 100px;  height: 100px;  margin-top: 30px;/\* 插入图片更改位置可以用margin 或padding 插入图片相当于盒模型。\*/  margin-left: 30px;  }  aside {  width: 400px;  height: 400px;  border: 1px solid blue;  **/\*background: skyblue url(images/3.jpg) no-repeat fixed;\*/**  **这个background-attachment: fixed的属性用在非body中会使得图片无法显示，使用scroll则没有问题，这个问题待理解清楚。？？？**  background: skyblue url(images/3.jpg) no-repeat;  background-size: 200px 210px; /\* 背景图片更改大小只能用 background-size \*/  background-position: 30px 50px; /\*背景图片更该位置只用background-position \*/  } |
| <section>  123  <img src="images/h.jpg" alt="图片路径有误">  </section>  <aside>  123  </aside> |

（4）清除元素的默认内外边距

为了更方便地控制网页中的元素，制作网页时，可使用如下代码清除元素的默认内外边距：

|  |
| --- |
| \* {  padding:0; /\* 清除内边距 \*/  margin:0; /\* 清除外边距 \*/  } |

注意：行内元素是只有**左右外边距(margin-left/margin-right)**的，是没有上下外边距的。内边距在ie6等低版本浏览器也会有问题。我们尽量不要给行内元素指定**上、下**的**内、外**边距就好了。**内边距、外边距只在水平方向有作用，垂直方向没有作用。虽然设置了上下左右50px，但是其效果不行。**

|  |
| --- |
| span {  background-color: red;  padding: 50px;  } |
| <span>11111</span> |
|  |

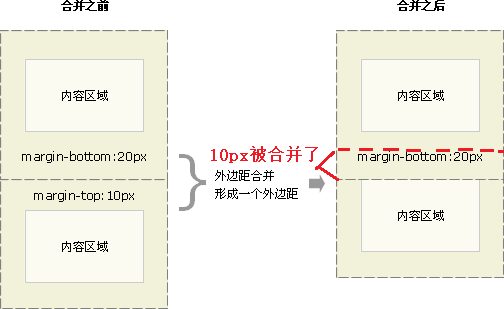
Chrome 中div 默认的margin是8px，其他浏览器默认的可能不是8px可能是其他，这个要做兼容性处理。它们各自的默认样式可能不太一样。

8、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，而是两者中的较大者。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**）。它们的外边距并没有消失，只是重叠了。



解决方式：可以只给定上盒子的下外边距，或者下盒子的上外边距，它们两不要同时给

（2）嵌套块元素垂直外边距的合并，**后边遇到这个问题比较多**

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，则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，即使**父元素的上外边距为0，也会发生合并**。

|  |
| --- |
| <style type="text/css">  .father {  width: 200px;  height: 200px;  background-color: red;  }  .son {  width: 100px;  height: 100px;  background-color: blue;  margin-top: 100px;  }  </style> |
| <div class="father">  <div class="son"></div>  </div> |
| 预期：2实际效果：1 |

解决方式：

1. 可以为父元素定义**1像素的上边框**或**上内边距**。（理解把它们外边距两分隔开，大家以这个border作为界限，里边的盒子，不能超出范围）

2. 可以为父元素添加overflow:hidden。

|  |
| --- |
| .father {  width: 200px;  height: 200px;  background-color: red;  **/\*border-top: 1px solid;\*//\*定义1像素的上边框\*/**  /\*padding-top: 1px;\*//\*上内边距\*/  overflow: hidden;/\*跟后边的BFC知识相关  } |

十九、盒子宽度和高度

使用宽度属性width和高度属性height可以对盒子的大小进行控制。

width和height的属性值可以为不同单位的数值或相对于父元素的百分比%，实际工作中最常用的是**像素值**。

大多数浏览器，如Firefox、IE6及以上版本都采用了W3C规范，符合CSS规范的盒子模型的总宽度和总高度的计算原则是：

/\*外盒尺寸计算（元素空间尺寸）\*/

Element空间高度 = content height + padding + border + margin

Element 空间宽度 = content width + padding + border + margin

/\*内盒尺寸计算（元素实际大小）\*/

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

Element Width = content width + padding + border （Width为内容宽度）

注意：

1、宽度属性width和高度属性height仅适用于块级元素，对行内元素无效（ **img 标签和 input除外**）。

2、计算盒子模型的**总高度时**，还应考虑上下两个盒**子垂直外边距合并（会取决于大者）**的情况。

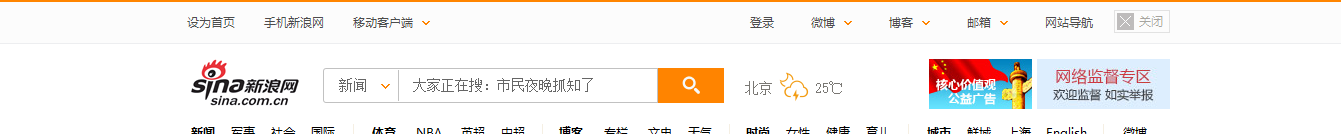
3、如果一个盒子**没有**给定**宽度/高度**或者继承父亲的宽度/高度？？？（高度还能被继承吗），则padding 不会影响本盒子大小。

|  |
| --- |
| div {  height: 100px;  /\*width: 100%;\*//\*默认的情况是容器的100%,但是写出来就明确指定了宽度，没有写表示没有宽度。\*/  background-color: red;  padding-left: 50px;  } |
| <div>不指定宽度的盒子</div> |
| 1 |
| 2 |

宽度继承，高度不能被继承

|  |
| --- |
| nav {  height: 200px;  width: 200px;  background-color: red;  }  div {  border: 1px solid #000;  padding-left: 100px;  }  实际上继承后width=98px。因为border左右各占了1px。 |
| <nav>  <div>儿子</div>  </nav> |
| 1 |

1. 例子



|  |
| --- |
| div {  width: 59px;/\*量取新闻的方框实际为79px\*/  height: 33px;  border: 1px solid #C1C1C1;  color: #666666;  line-height: 33px;  padding-left: 15px;/\*新闻的左边距量取之后为15px，这个左边距会加大盒子的宽度，所以宽度的设置要在原来的基础上减去这边距15px\*/  border-radius: 3px 0 0 3px;  } |
| <div>新闻</div> |
| 新闻字体大小、颜色、所占宽度、长度、边框大小，颜色，垂直居中，左边距，矩形圆角设置 |

二十、盒子模型布局稳定性

开始学习盒子模型，同学们最大的困惑就是，分不清内外边距的使用，什么情况下使用内边距，什么情况下使用外边距？

实际情况的使用决定：如下例子

|  |
| --- |
| nav {  height: 200px;  width: 200px;  background-color: red;  }  div {  background-color: skyblue;  padding-left: 100px;  } |
| <nav>  <div>儿子</div>  </nav> |
|  |
| div {  background-color: skyblue;  margin-left: 100px;  } |
|  |
| 总结：如果单单只是针对于文字内容，那么padding-left与margin-left都可以。  如果还要考虑到颜色就得看具体的需求，它们的效果不一样。 |

答案是：其实他们大部分情况下是可以混用的。就是说，你用内边距也可以，用外边距也可以。你觉得哪个方便，就用哪个。

但是，总有一个最好用的吧，我们根据稳定性来分，建议如下：

按照优先使用 宽度（width） 其次使用内边距（padding）再次外边距（margin）。

**width > padding > margin**

原因：

1. 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。

2. padding 会影响盒子大小， 需要进行加减计算（麻烦） 其次使用。

3. width 没有问题，我们经常使用宽度剩余法，高度剩余法来做。

宽度剩余法：



二十一、CSS3盒模型

CSS3中可以通过**box-sizing 来指定盒模型**，即可指定为content-box、border-box，这样我们计算盒子大小的方式就发生了改变。

可以分成两种情况：

1. box-sizing: content-box盒子大小为 **width + padding + border**。content-box:此值为其默认值，其让元素维持W3C的标准Box Mode。
2. box-sizing: border-box盒子大小为 width。就是说**padding 和 border 是包含到width里面的**。

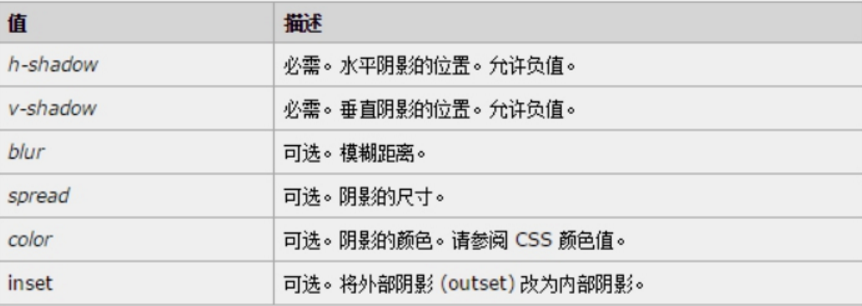
注：上面的标注的width指的是CSS属性里设置的width: xxxpx，content的值是会自动调整的。

|  |
| --- |
| div:first-child {  width: 200px;  height: 200px;  background-color: pink;  **box-sizing: content-box; /\* 就是以前的标准盒模型 w3c \*/**  padding: 10px;  border: 15px solid red;  /\* 盒子大小为 width + padding + border。content-box:此值为其默认值，其让元素维持W3C的标准Box Mode \*/  }    div:last-child {  width: 200px;  height: 200px;  background-color: purple;  padding: 10px;  **box-sizing: border-box; /\* padding border 不撑开盒子 \*/**  border: 15px solid red;  /\* margin: 10px; \*/  /\* 盒子大小为 width 就是说padding和border是包含到width里面的 \*/  } |

二十二、盒子阴影

语法格式：

box-shadow:水平阴影 垂直阴影 模糊距离 阴影尺寸 阴影颜色 内/外阴影；



1. 前两个属性是必须写的。其余的可以省略。

2. 外阴影 (outset) 但是**不能写**（默认是outset外阴影）（box-shadow: 3px 3px 3px 5px red outset;效果是无效的），想要内阴影inset。

|  |
| --- |
| div {  width: 200px;  height: 200px;  box-sizing: border-box;  margin: 0 auto;  box-shadow: 3px 3px 3px 5px red inset;  text-align: center;  line-height: 200px;  } |
| inset insetoutsetoutset |

1、搜索趣图小案例

清空默认的内外边距，几乎是我们写CSS，必须要写的

\* {

margin: 0;

padding: 0;

}

html, body, ul, li, ol, dl, dd, dt, p, h1, h2, h3, h4, h5, h6, form, fieldset, legend, img在

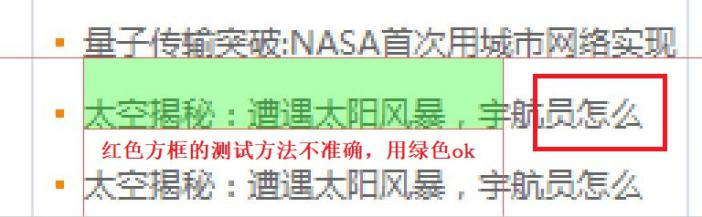
borer与border-top这两个语句之间的顺序会影响效果，位于下边的语句会起决定作用，它们之间的样式冲突，所以采用就近原则。

|  |
| --- |
| 未看视频前自己搞的 |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>CSS练习</title>  <style type="text/css">  .outside-box {  height: 294px;  width: 238px;  border: 1px solid #D9E0EE;  border-top: 3px solid #FF8400;  }  .outside-box div {  height: 35px;  border-bottom: 1px solid #D9E0EE;  line-height: 35px;  padding-left: 12px;/\*这里使用了margin-left会造成下边框也跟着移动，下边框前边会有空白出现，不合符要求。\*/  font-size: 13px;  }  .outside-box img {  padding-left: 8px;  padding-top: 7px;  }  ul {  list-style-type: square;  padding-left: 27px;  }  li::marker{  color: #FF8400;  }  .outside-box ul li a {  color: #666;  text-decoration: none;  font-size: 14px;  }  .outside-box ul li a:hover {  color: orange;  }  </style>  </head>  <body>  <div class="outside-box">  <div>趣图搜索</div>  <img src="images/ms.jpg" alt="图片路径有问题" height="161" width="218">  <ul>  <li><a href="#123">西瓜</a></li>  <li><a href="#123">苹果</a></li>  <li><a href="#123">西红柿</a></li>  </ul>  </div>  </body>  </html> |
| 原图：搜索趣图初次自己搞：  总结：1、一些类选择器的命名不太符合规范。   1. 在选择器中写属性时。有联系的尽量写在一起，例如字体 2. 在列表中的橙色小方形的理解误差还是比较大的，理解成了它使用的是列表自带的样式，暂时不知道怎么调这个小方形的大小，以及没有注意到 li 标签的高度是多少，只是表面看起来可能一样，但是实际上是有细微区别的。 3. 看完视频后，在 ul 列表那块的差别比较大，它把 ul 的样式都设置为none，去看了网站上的源码也是这样，然后使用一种图片作为 li 背景。   建议：在使用盒子时，先加上背景颜色，可以直观看到盒子的状态。 |

看完视频更改后：黄色部分为新添代码。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>CSS练习</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  ul {  list-style: none;/\*自带的小圆圈样式没有了\*/  }  .outside-box {  height: 294px;  width: 238px;  border: 1px solid #D9E0EE;  border-top: 3px solid #FF8400;  margin: 100px;/\*这样可以看到 ul 那个样式的小圆圈，不让会顶出去，被隐藏了\*/  }  .outside-box div {  height: 35px;  border-bottom: 1px solid #D9E0EE;  line-height: 35px;  padding-left: 12px;/\*这里使用了margin-left会造成下边框也跟着移动，下边框前边会有空白出现，不合符要求。\*/  font-size: 13px;  }  .outside-box img {  padding-left: 8px;  padding-top: 7px;  }  .outside-box ul li {  margin-left: 7px;/\*这里也可以在父元素 ul 中使用margin-left: 7px;能达到同样效果\*/  background: url(images/square.png) no-repeat;  background-position: left center;  height: 26px;  line-height: 26px;  }  .outside-box ul li a {  color: #666;  text-decoration: none;  font-size: 14px;  padding-left: 7px;/\*这里使用margin-left也是ok的\*/  }  .outside-box ul li a:hover {  color: #FF8400;  text-decoration: underline;  }  </style>  </head>  <body>  <div class="outside-box">  <div>趣图搜索</div>  <img src="images/ms.jpg" alt="图片路径有问题" height="161" width="218">  <ul>  <li><a href="#123">西瓜</a></li>  <li><a href="#123">苹果</a></li>  <li><a href="#123">西红柿</a></li>  </ul>  </div>  </body>  </html> |

1. 测量 li 的高度



二十三、浮动(float)

1、普通流(normal flow)

这个单词很多人翻译为 文档流 ， 字面翻译普通流 或者 标准流都可以。

前面我们说过，网页布局的核心，就是用CSS来摆放盒子位置。如何把盒子摆放到合适的位置？

CSS的定位机制有3种：普通流（标准流）、浮动和定位。

html语言当中另外一个相当重要的概念----------标准流！或者普通流。普通流实际上就是一个网页内标签元素正常从上到下，从左到右排列顺序的意思，比如块级元素会独占一行，行内元素会按顺序依次前后排列；按照这种大前提的布局排列之下绝对不会出现例外的情况叫做普通流布局。

注：把块级元素转换成行内块元素，虽然能让盒子在一行排列，但是它们之间会有距离，那个盒子的间距也不好调试。



2、浮动(float)

浮动最早是用来控制图片，以便达到其他元素（特别是文字）实现“环绕”图片的效果。

|  |
| --- |
| div {  width: 300px;  height: 300px;  border: 1px solid #000;  margin: 0 auto;  }  img {  float: right;  } |
| <div>html语言当中另外一个相当重要的概念----------标准流！或者普通流。普通流实际上就是一个网页内标签元素正常从上到下，<img src="images/h.jpg" width="100" height="100">从左到右排列顺序的意思，比如块级元素会独占一行，行内元素会按顺序依次前后排列；按照这种大前提的布局排列之下绝对不会出现例外的情况叫做普通流布局</div> |
|  |

后来，我们发现浮动有个很有意思的事情：就是让任何盒子可以一行排列,因此我们就慢慢的偏离主题(主题就是“最开始的时候，浮动是用来做文字环绕效果的”)，用浮动的特性来布局了。（CSS3已经我们真正意义上的网页布局，具体CSS3我们会详细解释）

（1）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，移动到其父元素中指定位置的过程。

在CSS中，通过float属性来定义浮动，其基本语法格式如下：

|  |
| --- |
| 选择器 {  float:属性值;  } |

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

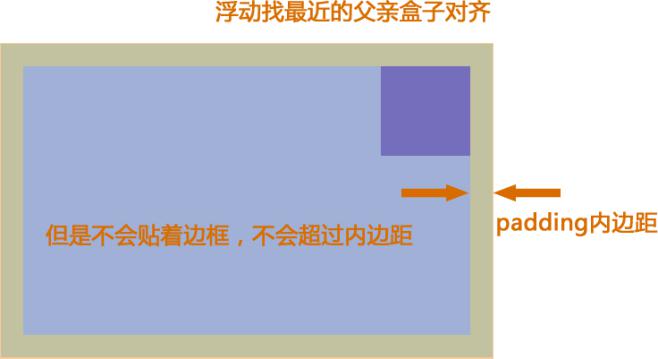
如果两个盒子都不加浮动，就是标准流，块级元素自上而下显示。

浮动现的漂浮的概念，漂浮在标准流的上面（就像布一样覆盖在标准流上边）

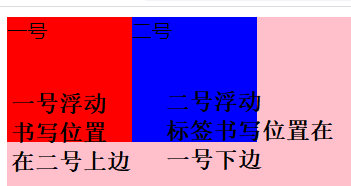
（2）浮动详细内幕特性

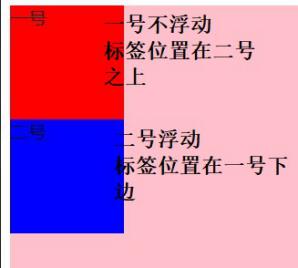
浮动脱离标准流，不占位置，会影响**它下边**标准流。浮动只有左右浮动。**它无法浮在文字上面。**

浮动首先创建包含块的概念（包裹）。就是说， 浮动的元素总是找理它最近的父级元素对齐。但是不会超出内边距的范围。



浮动的元素排列位置，跟上一个元素（块级--）（一号）有关系。如果上一个元素有浮动，则A元素（二号）顶部会和上一个元素（一号）的顶部对齐；如果上一个元素（一号）是标准流，则A元素（二号）的顶部会和上一个元素（一号）的底部对齐。





一个父盒子里面的子盒子，如果其中一个子级有浮动的，则其他子级都需要浮动。这样才能一行对齐显示。（如果前面一个盒子没有浮动，会独占一行，那么下一个盒子就会被挤到下一行）

元素添加浮动后，元素会具有**行内块元素**的特性。元素的大小（宽度、高度）完全取决于定义的大小或者默认的内容多少浮动根据元素书写的位置（宽度如果没有设置，它会根据内容宽度一致）来显示相应的浮动。

|  |
| --- |
| <style type="text/css">  div:first-child {  height: 100px;  background-color: red;  }  div:nth-child(2) {  height: 100px;  background-color: blue;  float: left;  }  span {  background-color: pink;  height: 100px;  float: left;  }  </style> |
| <div>一号</div>  <div>二号</div>  <span>行元素设置高度</span> |
| 未添加float属性：1  添加了float属性：2 |
| 行元素设置float，有行内快属性，其预先设置的高度就起作用了 |

（3）总结

浮动的**目的**就是为了让多个块级元素同一行上显示。

float：浮 漏 特

浮： 加了浮动的元素盒子是浮起来的，漂浮在其他的标准流盒子上面。

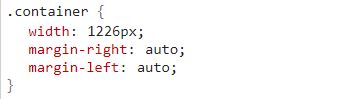
漏： 加了浮动的盒子，不占位置的，它浮起来了，它原来的位置漏给了标准流的盒子。

特： 特别注意，首先浮动的盒子需要和标准流的父级搭配使用，其次特别的注意浮动可以使元素显示模式体现为行内块特性。

二十四、版心和布局流程

阅读报纸时容易发现，虽然报纸中的内容很多，但是经过合理地排版，版面依然清晰、易读。同样，在制作网页时，要想使页面结构清晰、有条理，也需要对网页进行“排版”。

“版心”(可视区) 是指网页中主体内容所在的区域。一般在浏览器窗口中水平居中显示，常见的宽度值为960px、980px、1000px、1200px等。红色版心区域可能很长。



布局流程

为了提高网页制作的效率，布局时通常需要遵守一定的布局流程，具体如下：

1. 确定页面的版心（可视区）。

2、分析页面中的行模块，以及每个行模块中的列模块。先找出绿色部分，然后把里边的每个小红色先搞好



3、制作HTML结构 。

4、CSS初始化（清除所有的边距），然后开始运用盒子模型的原理，通过DIV+CSS布局来控制网页的各个模块。

（1）一列固定宽度且居中

最普通的，最为常用的结构。

