# 一 CSS

CSS是 Cascading Style Sheets 的首字母缩写,表示层叠样式表。有了CSS，html中大部分表现样式的标签就废弃不用了，html只负责文档的结构和内容，表现形式完全交给CSS，html文档变得更加简洁

好处：实现页面结构和样式的分离

# 二 引入方式

## 1 行间样式

只能用于本标签

<div style="border: 1px solid red;width: 200px;">

你好

<a href="#">百度</a>

</div>

## 2 内部样式表

只在本页面起作用

<style type="text/css">

p{

color: blue;

}

/\*CSS中注释方式\*/

</style>

## 3 外部样式表

通过link链接外部样式表到页面中

<link rel="stylesheet" href="css/test.css" />

# 三 选择器

通过选择器可以找到相应的数据进行样式化

语法格式：选择器{属性:属性值;属性:属性值}

选择器是将样式和页面元素关联起来的名称，属性是希望设置的样式属性，每个属性有一个或多个值

## 1 标签选择器

语法：

标签名{

}，如

p{

color: red;

}

## 2 类选择器

通过类名来选择元素，一个类可应用于多个元素，一个元素上也可以使用多个类，应用灵活。

语法：

.class属性名{

}

类选择器优先于标签选择器

多个标签可以使用相同的类名

.pclass{

color: blue;

}

## 3 id选择器

通过id名来选择元素，元素的id名称不能重复，所以一个样式设置项只能对应于页面上一个元素，不能复用

语法：

#id属性值{

}

id选择器优先于类选择器

#pid{

color: brown;

}

## 4 通用选择器

\*{

color:crimson;

}

## 5 交集选择器

选择器1选择器2...{

}

div.cd{

color:darkblue;

}

<div>

日照香炉生紫烟

</div>

<div class="cd">

遥看瀑布挂前川

</div>

<p class="cd">飞流直下三千尺</p>

## 6 并集选择器

对指定的选择器统一样式化

语法：

选择器1,选择器2...{

}

#aid,#sid{

color:aqua;

}

## 7 兄弟选择器

1)相邻兄弟兄弟选择(css2),只能选中紧跟后边的那个标签,不能选中被隔开的标签

格式:

选择器1+选择器2{

属性:值;

}

2)通用兄弟选择器(css3),指定选择器后边特定的所有选择器都被选中

格式:

选择器1~选择器2{

属性:值;

}

## 8 后代选择器

找到指定特定的选择器的所有后代，后代不一定是儿子,也可以是孙子

选择器1 选择器2 ...{

属性:值;

}

## 9 子元素选择器

找到指定选择器中所有特定的直接子元素，只会选择儿子

选择器1>选择器2{

属性:值;

}

## 10 属性选择器

根据指定的属性名称找到对应的标签

[attribute]：根据指定的属性名称找到对应的标签,设置属性

[attribute=value]：找到指定属性的值是value的标签,设置属性

[attribute^=value]：属性的取值是以什么开头的， css3

[attribute$=value]：属性的取值是以什么结尾的 css3

[attribute\*=value]：属性的取值是否包含某个特定的值的 css3

## 11 伪类选择器

/\*伪类选择器 设置标签在某种状态下的样式

\* hover 鼠标经过的样式

\* link 没有点击前的样式

\* visited 点击后的样式

\*

\* hover要写在link和visited后

\* 一次点击后，如果再次运行该界面，显示点击后的颜色，清除缓存后恢复原来状态

\* \*/

a:link{

color: black;

}

a:visited{

color:blue;

}

a:hover{

color:red;

}

例子：鼠标移动到表格的某行时，改变行的背景色

table{

border: 1px solid black;

/\* 边框折叠 \*/

border-collapse: collapse;

}

tr:hover{

/\*color: gray;\*/

background-color: gray;

}

## 12 CSS3新增选择器 了解

E:nth-child(n)：匹配元素类型为E且是父元素的第n个子元素

li{

background-color:gold;

margin-bottom:10px;

}

/\* 匹配li元素，匹配的元素是ul的第一个子元素 \*/

/\*li:nth-child(1){

background-color:green;

}\*/

/\* 匹配li元素，匹配的元素是ul的第一个子元素 \*/

/\*li:nth-child(3){

background-color:pink;

}\*/

/\* 匹配li的奇数行 \*/

/\*li:nth-child(2n+1){

background-color:green;

}\*/

E:nth-last-child(n)：匹配元素类型为E且是父元素的倒数第n个子元素（与上一项顺序相反）

E:first-child：匹配元素类型为E且是父元素的第一个子元素

E:last-child：匹配元素类型为E且是父元素的最后一个子元素

E:only-child：匹配元素类型为E且是父元素中唯一的子元素

E:nth-of-type(n)：匹配父元素的第n个类型为E的子元素

/\* 匹配第三个子元素，而且这个子元素的类型是div，就匹配 \*/

.box div:nth-child(3){

background-color: green;

}

/\* 父级下面第三个div元素，碰到不同类型的元素跳过 \*/

.box div:nth-of-type(3){

background-color: green;

}

<div class="box">

<div>1</div>

<div>2</div>

<p>p标签</p>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

</div>

E:nth-last-of-type(n)：匹配父元素的倒数第n个类型为E的子元素（与上一项顺序相反）

E:first-of-type：匹配父元素的第一个类型为E的子元素

E:last-of-type：匹配父元素的最后一个类型为E的子元素

E:only-of-type：匹配父元素中唯一子元素是E的子元素

E:empty 选择一个空的元素

E:enabled 可用的表单控件

E:disabled 失效的表单控件

E:checked 选中的checkbox

E:not(s) 不包含某元素

<style type="text/css">

div:not(:nth-child(2)){

background-color:red;

}

</style>

......

<!-- 第 3、4、5 子元素div匹配 -->

<div class="list">

<h2>1</h2>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

</div>

E:target 对应锚点的样式

<style type="text/css">

h2:target{

color:red;

}

</style>

......

<!-- 点击链接，h2标题变红 -->

<a href="#tit01">标题一</a>

......

<h2 id="tit01">标题一</h2>

# 四 CSS属性

## 1 背景图样式

background-color 设置背景颜色

background-image 设置背景图片地址

background-repeat 设置背景图片如何重复平铺

background-position 设置背景图片的位置

background-attachment 设置背景图片是固定还是随着页面滚动条滚动

<style type="text/css">

body{

/\*background-color: red;\*/

/\*设置背景图片\*/

background-image: url(img/dog.gif);

/\*背景图片是否重复 repeat no-repeat repeat-x repeat-y\*/

background-repeat: no-repeat;

/\*图像左上角的点距离左边和上边的距离\*/

background-position: 50px 100px;

/\*背景图是否随着网页的滚动而滚动 scorll fixed\*/

background-attachment: scroll;

}

</style>

## 2 边框样式

<style type="text/css">

div{

width: 200px;

height: 200px;

/\*border: 2px solid red;\*/

/\*可以有四个值，顺序上 右 下 左

如果只有一个值，所有的边框都是用该效果

如果设置两个值，先上下，再左右

如果设置三个值：上 左右 下

\* \*/

/\*border-style: solid;\*/

border-style: dotted solid double;

border-color: #F00; /\*可以针对四边\*/

border-bottom-color: #0FF;

border-width: 10px;

border-bottom-width: 20px;

}

</style>

## 3 文字样式

color 设置文字的颜色，如： color:red;

font-size 设置文字的大小，如：font-size:12px;

font-family 设置文字的字体，如：font-family:'微软雅黑';

font-style 设置字体是否倾斜，如：font-style:'normal'; 设置不倾斜，font-style:'italic';设置文字倾斜

font-weight 设置文字是否加粗，如：font-weight:bold; 设置加粗 font-weight:normal 设置不加粗

font 同时设置文字的几个属性，写的顺序有兼容问题，建议按照如下顺序写： font：是否加粗 字号/行高 字体；如： font:normal 12px/36px '微软雅黑';

line-height 设置文字的行高，如：line-height:24px;

text-decoration 设置文字的下划线，如：text-decoration:none; 将文字下划线去掉

text-indent 设置文字首行缩进，如：text-indent:24px; 设置文字首行缩进24px

text-align 设置文字水平对齐方式，如text-align:center 设置文字水平居中

<style type="text/css">

div{

border: 1px solid red;

width: 300px;

height: 200px;

font-size:20px;/\*文字大小\*/

font-style: italic;/\*文字的风格\*/

font-weight: 600;/\*字体的粗细bolder 或者具体数字\*/;

line-height: 30px;/\*字体的行高 文字在水平方向居中\*/

text-indent: 20px;/\*文字的缩进 2em 两个文字的距离\*/

word-spacing: 10px;/\* 单词之间的间隔\*/

letter-spacing: 10px;/\*字符之间的间隔\*/

text-decoration:underline;/\*文字的修饰 none underline等\*/

}

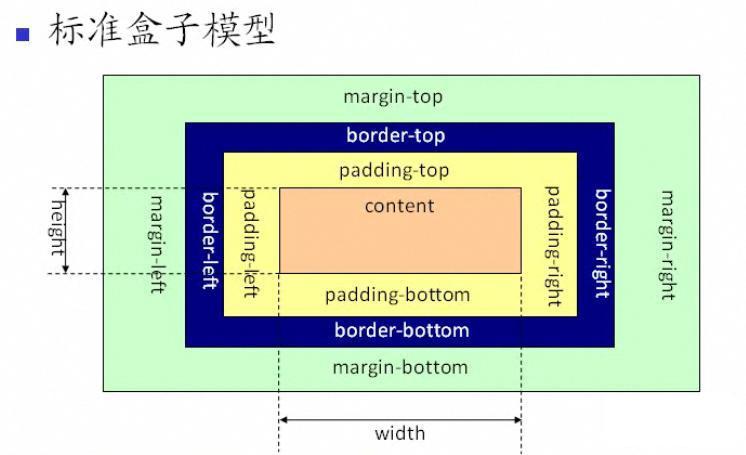
p{

text-align: center;

}

</style>

# 五 盒模型 重点中的重点



margin：外边距

padding：内边距

把元素叫做盒子，设置对应的样式分别为：盒子的边框(border)、盒子内的内容和边框之间的间距(padding)、盒子与盒子之间的间距(margin)。

盒子的真实尺寸计算公式如下：

盒子宽度 = width + padding左右 + border左右+margin左右

盒子高度 = height + padding上下 + border上下+margin上下

**设置边框**

设置一边的边框，比如顶部边框，可以按如下设置：

border-top-color:red; /\* 设置顶部边框颜色为红色 \*/

border-top-width:10px; /\* 设置顶部边框粗细为10px \*/

border-top-style:solid; /\* 设置顶部边框的线性为实线，常用的有：solid(实线) dashed(虚线) dotted(点线); \*/

上面三句可以简写成一句：

border-top:10px solid red;

设置其它三个边的方法和上面一样，把上面的'top'换成'left'就是设置左边，换成'right'就是设置右边，换成'bottom'就是设置底边。

四个边如果设置一样，可以将四个边的设置合并成一句：

border:10px solid red;

**设置内间距padding**

设置盒子四边的内间距，可设置如下：

padding-top：20px; /\* 设置顶部内间距20px \*/

padding-left:30px; /\* 设置左边内间距30px \*/

padding-right:40px; /\* 设置右边内间距40px \*/

padding-bottom:50px; /\* 设置底部内间距50px \*/

上面的设置可以简写如下：

padding：20px 40px 50px 30px; /\* 四个值按照顺时针方向，分别设置的是 上 右 下 左 四个方向的内边距值。 \*/

padding后面还可以跟3个值，2个值和1个值，它们分别设置的项目如下：

padding：20px 40px 50px; /\* 设置顶部内边距为20px，左右内边距为40px，底部内边距为50px \*/

padding：20px 40px; /\* 设置上下内边距为20px，左右内边距为40px\*/

padding：20px; /\* 设置四边内边距为20px \*/

**设置外间距margin**

外边距的设置方法和padding的设置方法相同

## 1 使用例子

<style type="text/css">

\*{

/\*外边距\*/

margin: 0;

/\*内边距\*/

padding: 0;

}

img{

border: 5px solid red;

width: 200px;

padding: 20px;

margin: 20px;

}

body{

border: 3px dashed blue;

}

</style>

<img src="img/pic\_09.jpg" /><img src="img/pic\_12.jpg" />



## 2 margin的特点

1）外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者

<style type="text/css">

div{

width: 200px;

height: 200px;

}

/\* 垂直方向上，margin会根据大值进行合并

\* 水平方向上没有影响

\* \*/

#div1{

background-color: red;

margin: 50px;

}

#div2{

background-color: green;

margin: 30px;

}

</style>

2）margin-top 塌陷

在两个盒子嵌套时候，内部的盒子设置的margin-top会加到外边的盒子上，导致内部的盒子margin-top设置失败

解决方法如下：

外部盒子设置一个边框或者padding；

外部盒子设置 overflow:hidden

<style type="text/css">

/\* 父子元素的margin也可以合并

\* 通过给父元素设置border属性，可以取消合并的影响

\* \*/

#div1{

background-color: red;

width: 150px;

height: 150px;

/\*margin: 50px auto;\*/

/\*border: 1px solid red;\*/

/\*padding: 1px;\*/

overflow: hidden;

}

#div2{

background-color: green;

width: 50px;

height: 50px;

margin: 50px;

}

</style>

<div id="div1">

<div id="div2">

</div>

</div>

练习：居中的登录界面

<style type="text/css">

#outer{

width: 300px;

height: 200px;

border: 1px solid black;

margin: 100px auto;

background-color: grey;

}

#name{

margin-top: 60px;

margin-left: 40px;

}

#pwd{

margin-top: 20px;

margin-left: 40px;

}

</style>

<div id="outer">

<div id="name">

用户名<input type="text"/>

</div>

<div id="pwd">

密&nbsp;码<input type="password"/>

</div>

</div>

# 六 浮动和定位

## 1 浮动 float属性

文档流，是指盒子按照html标签编写的顺序依次从上到下，从左到右排列，块元素占一行，行内元素在一行之内从左到右排列，先写的先排列，后写的排在后面，每个盒子都占据自己的位置。

概括来说，HTML页面的标准文档流(默认布局)是：从上到下，从左到右，遇块(块级元素)换行。

给元素的float属性赋值后，该元素脱离文档流，进行左右浮动，紧贴着父元素(默认为body文本区域)的左右边框。

而此浮动元素在文档流空出的位置，由后续的(非浮动)元素填充上去：块级元素直接填充上去，若跟浮动元素的范围发生重叠，浮动元素覆盖块级元素。

例子1：左右浮动，包括没有浮动的div

<style type="text/css">

div{

float: left;

}

#div1{

/\*float: left;\*/

width: 100px;

height: 100px;

background-color: red;

}

#div2{

/\*float: left;\*/

width: 100px;

height: 150px;

background-color: yellow;

}

#div3{

/\*float: left;\*/

width: 100px;

height: 200px;

background-color: mediumspringgreen;

}

例子2：ul作为菜单

横向

<style>

\*{

margin: 0;

padding: 0;

}

li{

float: left;

list-style: none;

margin-right: 10px;

}

</style>

<ul>

<li>

<a href="#">Link one</a>

</li>

<li>

<a href="#">Link two</a>

</li>

<li>

<a href="#">Link three</a>

</li>

<li>

<a href="#">Link four</a>

</li>

</ul>

例子3：设置li的图像风格

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

ul{

color: red;

font-size: 14px;

list-style: none;/\*列表的风格\*/

/\*list-style-image: url(img/icon.gif);\*/

}

li{

background: url(img/icon.gif) no-repeat;

padding-left:20px ;

}

</style>

## 2 定位Position

规定元素的定位类型。

**absolute**：绝对定位；脱离文档流的布局，遗留下来的空间由后面的元素填充。定位的起始位置为最近的父元素(postion不为static)，否则为Body文档本身。

**relative**：相对定位；不脱离文档流的布局，只改变自身的位置，在文档流原先的位置遗留空白区域。定位的起始位置为此元素原先在文档流的位置（相对于原来的位置）。

**fixed**：固定定位；类似于absolute，但不随着滚动条的移动而改变位置。

**static**：默认值；默认布局。

相对定位：相对于原来的位置

<style type="text/css">

div{

float: left;

border: 1px solid red;

width: 100px;

height: 100px;

}

#div2{

position: relative;

top: 10px;

left: 30px;

}

</style>

绝对定位：

<style type="text/css">

div{

/\*float: left;\*/

border: 1px solid red;

width: 100px;

height: 100px;

}

#div2{

position: absolute;

border-color: blue;

top: 10px;

left: 300px;

}

</style>

固定定位：相对于浏览器，不会随着滚动条滚动而滚动

<style type="text/css">

div{

/\*float: left;\*/

border: 1px solid red;

width: 100px;

height: 100px;

}

#div2{

position: fixed;

border-color: blue;

top: 100px;

left: 300px;

}

</style>

## 3 z-index用法

# 七 CSS3样式

## 1 CSS3圆角

border-top-left-radius: <length> <length> //左上角

border-top-right-radius: <length> <length> //右上角

border-bottom-right-radius:<length> <length> //右下角

border-bottom-left-radius:<length> <length> //左下角

第一个值是圆角水平半径，第二个值是垂直半径

设置某一个角的圆角，比如设置左上角的圆角：

border-top-left-radius:30px 100px ;

同时分别设置四个角： border-radius:30px 60px 120px 150px;

设置四个圆角相同：

border-radius:50%;

.box{

width:200px;

height:200px;

border:2px solid #000;

background-color:gold;

margin:50px auto 0;

border-top-left-radius:100px 50px;

/\*border-top-left-radius:100px;

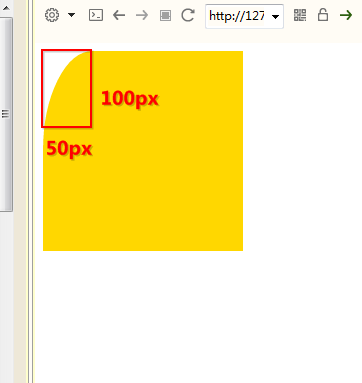
border-top-right-radius:100px;\*/

/\*border-radius:40px;\*/

/\*border-radius:20%;\*/

/\*border-radius:50%;\*/

}



## 2 CSS3阴影

box-shadow：h-shadow v-shadow blur spread color inset;

分别设置阴影：水平偏移 垂直偏移 羽化大小 扩展大小 颜色 是否内阴影

<style type="text/css">

.box{

width:200px;

height:40px;

background-color:gold;

margin:100px auto 0;

box-shadow:10px 10px 10px 5px pink;

}

.box2{

width:200px;

height:40px;

background-color:gold;

margin:100px auto 0;

box-shadow:0px 0px 20px 2px red inset;

}

</style>

<div class="box"></div>

## 3 rgba

rgba(0,0,0,0.1) 前三个数值表示颜色，第四个数值表示颜色的透明度

.box2{

width:200px;

height:200px;

margin:50px auto 0;

border:2px solid rgba(0,0,0,0.3);

border-radius:50%;

text-align:center;

line-height:200px;

/\*opacity:0.3;\*/

background-color:rgba(255,215,0,0.3);

}

## 4 transition 过渡效果 了解

通过过渡transition，可以让web前端开发人员不需要javascript就可以实现简单的动画交互效果

触发过渡的方式：

常见的就是伪类触发:hover，:focus，:active，:checked等。还有如click，keydown等JS事件触发。另外，页面加载也能触发。总之过渡的本质是在时间段内平滑过渡属性值，与怎么触发没有关系。

transition: property duration timing-function delay, property duration timing-function delay,……

其中：

property 设置过渡的属性，比如：width height background-color

duration 设置过渡的时间，比如：1s 500ms

timing-function 设置过渡的运动方式:

linear 匀速

ease 开始和结束慢速

ease-in 开始是慢速

ease-out 结束时慢速

ease-in-out 开始和结束时慢速

cubic-bezier(n,n,n,n)

比如：cubic-bezier(0.845, -0.375, 0.215, 1.335)

delay 设置动画的延迟

如果使用all，表示针对所有的property

transition: all duration timing-function delay

.box{

width:100px;

height:100px;

background-color:gold;

transition:border-radius 500ms ease,width 500ms ease 500ms,height 500ms ease 1s,background-color 500ms ease 1.5s;

/\*transition:all 500ms ease;\*/

}

.box:hover{

width:500px;

height:300px;

background-color:red;

/\*border-radius:50px;\*/

}