# 1.选择器

## 1）标签选择器

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>样式表基础-id选择器</title>        <style>          div{              background-color: green;              font-size: 30px;          }          /\*其实是那个近就那个先其主要 \*/        </style>    </head>  <body>      <!-- <div style="background-color: red;">欢迎来到HTML课堂</div> -->      <!-- 内嵌样式优先级比内联样式高 -->      <!-- 在实际开发中一般使用外部样式表，这样子比较方便省事 -->      <div >欢迎来到HTML课堂</div>    </body> |

## 2）id选择器

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>样式表基础-id选择器</title>        <style>          #box{              background-color: green;              font-size: 30px;          }          /\*其实是那个近就那个先其主要 \*/        </style>      <link rel="stylesheet" href="css/main.css">  </head>  <body>      <!-- <div style="background-color: red;">欢迎来到HTML课堂</div> -->      <!-- 内嵌样式优先级比内联样式高 -->      <!-- 在实际开发中一般使用外部样式表，这样子比较方便省事 -->      <div >欢迎来到HTML课堂</div>      <div id="box">欢迎来到CSS课堂</div>      <!-- id选择器的优先级比标签选择器高 -->      <!-- 建议一般不要使用id把它留给js，因为js使用id来找标签比较方便 -->      <!-- id是不允许重复的 -->  </body>  </html> |

## 3）.类选择器

|  |
| --- |
| Html |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>样式表基础-class选择器</title>        <style>          div{              background-color: green;              font-size: 30px;          }          /\*其实是那个近就那个先其主要 \*/        </style>      <link rel="stylesheet" href="css/main.css">  </head>  <body>      <!-- <div style="background-color: red;">欢迎来到HTML课堂</div> -->      <!-- 内嵌样式优先级比内联样式高 -->      <!-- 在实际开发中一般使用外部样式表，这样子比较方便省事 -->      <div >欢迎来到HTML课堂</div>      <div class="box">欢迎来到CSS课堂</div>      <!-- class选择器的优先级比标签选择器高 -->      <!-- class是允许重复的 -->      <h1 class="box biaoti">我是h1</h1>      <!-- 可以在同一个class里面写两个名字，用空格隔开，使用任何一个名字都可以找到这个标签 -->      <!-- 类名不能包含特色字符如#，@，&、.等等 因为他们是有特殊用途的-->      <!-- 类名建议使用见文知义的写法，就是起一些有意义的类名 -->  </body>  </html> |
| main.css |
| .box{      background-color: aqua;      text-align: center;      color: yellow;  }  .biaoti{      font-weight: 500;      font-size: 60px;  } |

# 2.css常用语法(有一些比较简单的语法这里没有记录如宽度高度等等)

## 1）边框语法

|  |
| --- |
| border: 边框宽度 边框元素 边框颜色;  如：  **border : 2px solid red** |

## 2）使div居中显示

|  |
| --- |
| **div{**  **margin: 0 auto; /\***居中显示，0表示上下的外边距为0，auto表示左右的外边距自动均分**\*/**  **}** |

# 3.盒子模型（参考菜鸟教程，不过做了适当扩充）

## CSS 盒子模型(Box Model)

所有HTML元素可以看作盒子，在CSS中，"box model"这一术语是用来设计和布局时使用。

CSS盒模型本质上是一个盒子，封装周围的HTML元素，它包括：边距，边框，填充，和实际内容。

盒模型允许我们在其它元素和周围元素边框之间的空间放置元素。

下面的图片说明了盒子模型(Box Model)：



## 不同部分的说明：

* **Margin(外边距)** - 清除边框外的区域，外边距是透明的。
* **可以一次设置上右下左的边距也可以分别设置**
* **margin-top：上边距**
* **margin-right：右边距**
* **margin-bottom：下边距**
* **margin-left：左边距**

## 用margin设置盒子水平居中

* **还可以设置两个值，分别表示上下边距和左右边距**
* **如margin: 0 auto; 表示上下边距为0，左右边距均分**
* **需要注意的是这种方法只能适当盒子水平居中，垂直方向是不行的，即使我们使用margin: auto auto;它也只是水平居中，因为html的高度是由内容来撑开的。**
* **还可以设置3个值，分别表示上边距 左右边距 下边距**
* **如：margin : 30px 100px 50px; 表示上边距30px 左右边距各100px 下边距50px**
* **还可以设置4个值，是按照上右下左的顺序设置的**

## 用margin实现垂直居中的方法，使用margin-top和calc()函数来计算

|  |
| --- |
| margin-top : calc((父容器的高度 - 子容器的高度) / 2) |

#### 注意：1.计算的数值必须带单位。2.每一个数值和运算符之间必须要右一个空格

## margin为负数的用法

使得盒子网相反方向移动

## margin的影响

#### 子元素的margin-top属性会传递给没有设置边框的属性的父元素

#### 解决办法：

#### 1给父元素设置边框属性，但是这个会改变宽高

#### 2.给父元素设置overflow属性为hidden，这个可以很好的解决上面的问题

* **Border(边框)** - 围绕在内边距和内容外的边框。
* **Padding(内边距)** - 清除内容周围的区域，内边距是透明的。
* **可以一次设置上右下左的边距也可以分别设置**
* padding-top:  上内边距;
* padding-right:  右内边距;
* padding-bottom: 下内边距;
* padding-left:  左面内边距;
* 和margin一样，padding属性也是会撑大盒子的大小
* 和margin类似，padding也可以设置1，2，3，4个值，也是按照顺时针顺序的
* **Content(内容)** - 盒子的内容，显示文本和图像。

为了正确设置元素在所有浏览器中的宽度和高度，你需要知道的盒模型是如何工作的。

## 元素的宽度和高度

Remark**重要:** 当您指定一个 CSS 元素的宽度和高度属性时，你只是设置内容区域的宽度和高度。要知道，完整大小的元素，你还必须添加内边距，边框和外边距。

下面的例子中的元素的总宽度为 450px：

## 实例

div { width: 300px; border: 25px solid green; padding: 25px; margin: 25px; }

[**尝试一下 »**](https://www.runoob.com/try/try.php?filename=trycss_boxmodel)

让我们自己算算：  
300px (宽)  
+ 50px (左 + 右填充)  
+ 50px (左 + 右边框)  
+ 50px (左 + 右边距)  
= 450px

试想一下，你只有 250 像素的空间。让我们设置总宽度为 250 像素的元素:

## 实例

div { width: 220px; padding: 10px; border: 5px solid gray; margin: 0; }

[**尝试一下 »**](https://www.runoob.com/try/try.php?filename=trycss_boxmodel_width)

最终元素的总宽度计算公式是这样的：

总元素的宽度=宽度+左填充+右填充+左边框+右边框+左边距+右边距

元素的总高度最终计算公式是这样的：

总元素的高度=高度+顶部填充+底部填充+上边框+下边框+上边距+下边距

## 浏览器的兼容性问题

一旦为页面设置了恰当的 DTD，大多数浏览器都会按照上面的图示来呈现内容。然而 IE 5 和 6 的呈现却是不正确的。根据 W3C 的规范，元素内容占据的空间是由 width 属性设置的，而内容周围的 padding 和 border 值是另外计算的。不幸的是，IE5.X 和 6 在怪异模式中使用自己的非标准模型。这些浏览器的 width 属性不是内容的宽度，而是内容、内边距和边框的宽度的总和。

虽然有方法解决这个问题。但是目前最好的解决方案是回避这个问题。也就是，不要给元素添加具有指定宽度的内边距，而是尝试将内边距或外边距添加到元素的父元素和子元素。

IE8 及更早IE版本不支持设置填充的宽度和边框的宽度属性。

解决IE8及更早版本不兼容问题可以在HTML页面声明 <!DOCTYPE html>即可。

## 盒子模型的大小的计算

盒子模型的宽 = width（content） + 边框宽度\*2 +padding\*2+margin\*2（假如只设置一个值）

盒子模型的高 = height（content） + 边框宽度\*2 +padding\*2+margin\*2（假如只设置一个值）

# 4.标准盒子模型和怪异盒子模型

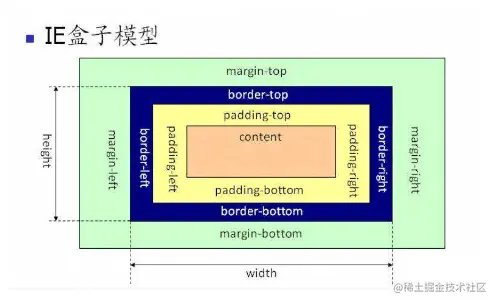
CSS中Box model是分为两种:: W3C标准 和 IE标准盒子模型。

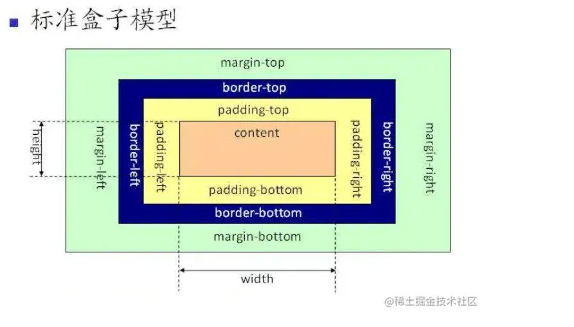
大多数浏览器采用W3C标准模型，而IE中则采用Microsoft自己的标准。

怪异模式是“部分浏览器在支持W3C标准的同时还保留了原来的解析模式”，怪异模式主要表现在IE内核的浏览器。

当不对doctype进行定义时，会触发怪异模式。

* **在标准模式下，一个块的总宽度= width + margin(左右) + padding(左右) + border(左右)**
* **在怪异模式下，一个块的总宽度= width + margin(左右)（即width已经包含了padding和border值）**



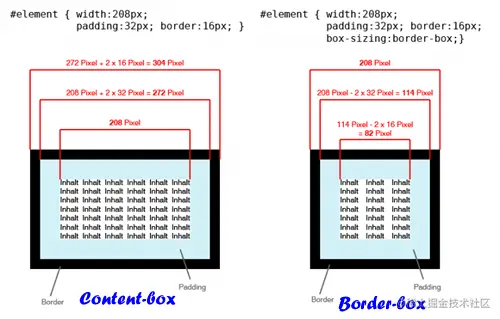


## ****CSS3的box-sizing****

box-sizing语法：

box-sizing ： content-box || border-box || inherit;

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



### 怪异盒子模型的padding和边框属性不会撑大盒子的大小