# 选择器 #

**选择器**分为基础选择器和复合选择器两个大类，我们这里先讲解一下基础选择器。

基础选择器是由单个选择器组成的

基础选择器又包括：标签选择器、类选择器、id 选择器和通配符选择器  


**2.1 块元素**

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。

**2.2 行内元素**

常见的行内元素有 <a>、<strong>、<b>、<em>、<i>、<del>、<s>、<ins>、<u>、<span>等，其中

<span> 标签是最典型的行内元素。有的地方也将行内元素称为内联元素。

行内元素的特点

**2.3 行内块元素**

在行内元素中有几个特殊的标签 —— <img />、<input />、<td>，它们同时具有块元素和行内元素的特点。 有些资料称它们为行内块元素



### ⑴替换元素：

替换元素是浏览器根据其标签的元素与属性来判断显示具体的内容。

比如：<input /> type="text" 的是，这是一个文本输入框，换一个其他的时候，浏览器显示就不一样

XHTML中的<img>、<input>、<textarea>、<select>、<object>都是替换元素，这些元素都没有实际的内容。

### ⑵非替换元素：

XHTML 的大多数元素是不可替换元素，他们将内容直接告诉浏览器，将其显示出来。

比如<p>wanmei.com</p>，浏览器将把这段内容直接显示出来。

Css三大特性

* 层叠性
* 继承性
* 优先级
  + 权重计算方法（特异性）

层叠性：

* 层叠性是指当一个标签被设置了多个重复的样式的时候，一个属性会覆盖另外一个属性。
* 比如：先给div设置背景颜色为红色，然后设置背景颜色为粉红色，那么最终颜色可能为红色，也可能为粉红色（明显的两种颜色设置不能并立，所以只为其一，这里没给出确定答案是因为还要考虑“优先级”）

层叠性主要遵循的原则是就近原则（**或者理解为覆盖**），在不考虑优先级的情况下，在多个样式中最终生效的样式是离标签最近的样式。【这里要注意只覆盖同样的不可重复的样式属性，不会覆盖不重复的属性，不重复的属性会合并下来（包括那些允许设置多个值的属性也会合并，比如text-shadow）。

继承性：

* 继承性是指当子标签没有设置样式时，会继承父标签的样式。
* 比如一个子标签没有设置背景颜色，那么子标签会继承父标签的背景颜色。

**可继承的属性**比如有字体类属性（**字体颜色、字体大小**之类的）、文本类属性（**行高**之类的）、背景类属性（**背景颜色**之类的）（要注意子标签不会继承父标签的宽度、高度、边距属性。）

优先级：

* 优先级：!important>行内样式>id选择器>类选择器>标签选择器>通配符>继承 【优先级高的规则会忽视层叠性，忽视就近原则来定义样式。比如定义了!important，即使行内样式离得近，最终生效的还是!important标注的属性】
  + !important是css的一种语法，定义在样式属性后面，代表这个属性不会被覆盖，最终生效的属性一定是!important标注的属性。

权重计算方法【可以称为是css的特异性】：

* 几个规则：
  + 权重使用四个数字来衡量(x,x,x,x)【CSS2使用三位】
  + 继承的权重为(0,0,0,0)
  + 标签选择器的权重为(0,0,0,1)
  + 类、伪类选择器的权重为(0,0,1,0)
  + id选择器选择器的权重为(0,1,0,0)
  + 行内样式的权重为(1,0,0,0)
  + !important的权重无限大
* **计算方法**：
  + 多个选择器混杂时，权重之和也是用四个数字来组成，每一位的值为多个选择器四个数字的每位之和，比如div:first-child的权重为(0,0,0,1)+(0,0,1,0)=(0,0,1,1)，比如div ul li 为(0,0,0,3),比如a:hover为(0,0,1,1)
  + 权重之和的数制是不会进位的，再多的标签选择器权重加和也抵不过一个类选择器。
  + 多个选择器之间的顺序是无关的，不影响权重之和。
  + 注意：对于并集选择器来说不是权重的加和，因为理论上它还是独立的，它只相当于将多个选择器的相同内容归于一个并集选择器中，理论上每一个选择器还是独立的

**权重相同时，考虑就近原则**

**1.4 表格的细线边框**

border-collapse 属性控制浏览器绘制表格边框的方式。它控制相邻单元格的边框。

语法：

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

**1.7 外边距典型应用**

外边距可以让块级盒子**水平居中**，但是必须满足两个条件：

① 盒子必须指定了宽度（width）。

② 盒子左右的外边距都设置为 auto 。

.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;

### Auto自适应

auto是自适应的意思，auto是很多尺寸值的默认值，也就是由浏览器自动计算。

块级元素中margin、border、padding以及content宽度之和构成父元素width。

使用auto属性后，父元素宽度发生变化，该元素的宽度也会随之变化。

**1.8 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。

主要有两种情况:

**1. 相邻块元素垂直外边距的合并**

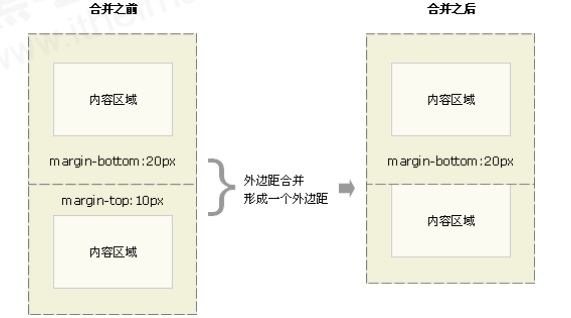
**2. 嵌套块元素垂直外边距的塌陷**

**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（兄弟关系）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**。

**解决方案：**

尽量只给一个盒子添加 margin 值。



**2. 嵌套块元素垂直外边距的塌陷**

对于两个嵌套关系（父子关系）的块元素，父元素有上外边距同时子元素也有上外边距，此时父元素会塌陷较大的外边距值。

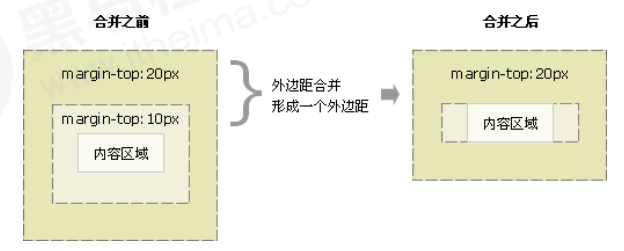
**解决方案：**

① 可以为父元素定义上边框。

② 可以为父元素定义上内边距。

③ 可以为父元素添加 overflow:hidden。

还有其他方法，比如浮动、固定，绝对定位的盒子不会有塌陷问题，后面咱们再总结。



**1.9 清除内外边距**

网页元素很多都带有默认的内外边距，而且不同浏览器默认的也不一致。因此我们在布局前，首先要清除下网页元素的内外边距。

\* {

padding:0; /\* 清除内边距 \*/

margin:0; /\* 清除外边距 \*/

}

**注意：**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。但是转换为块级和行内块元素就可以了

**4. 圆角边框**

在 CSS3 中，新增了圆角边框样式，这样我们的盒子就可以变圆角了。

border-radius 属性用于设置元素的外边框圆角。

语法：

border-radius:length;

参数值可以为数值或百分比的形式

如果是正方形，想要设置为一个圆，把数值修改为高度或者宽度的一半即可，或者直接写为 50%

该属性是一个简写属性，可以跟四个值，分别代表左上角、右上角、右下角、左下角

分开写：border-top-left-radius、border-top-right-radius、border-bottom-right-radius 和

border-bottom-left-radius

### Box-shadow

CSS3 中新增了盒子阴影，我们可以使用 box-shadow 属性为盒子添加阴影。

语法：

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



**注意**：

1. 默认的是外阴影(outset), 但是不可以写这个单词,否则造成阴影无效

2.盒子阴影不占用空间，不会影响其他盒子排列。

**6. 文字阴影**

在 CSS3 中，我们可以使用 text-shadow 属性将阴影应用于文本。

**1. 浮动（float)**

**1.1 传统网页布局的三种方式**

CSS 提供了三种传统布局方式(简单说,就是盒子如何进行排列顺序)：

普通流（标准流）

浮动

定位

**1.2 标准流（普通流/文档流）**

所谓的标准流: 就是标签按照规定好默认方式排列.

1. 块级元素会独占一行，从上向下顺序排列。

2. 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。

### 浮动用法

谁浮动给谁加

  <div class="box">

    <div>aa</div>

    <div>bb</div>

  </div>

    .box {

    }

    .box div:nth-child(1){

      background-color: rebeccapurple;

      float: left;

    }

    .box div:nth-child(2){

      background-color: aqua;

    }



**1.5 浮动特性（重难点）**

加了浮动之后的元素,会具有很多特性,需要我们掌握的.

1. 浮动元素会脱离标准流(脱标)

2. 浮动的元素会一行内显示并且元素顶部对齐

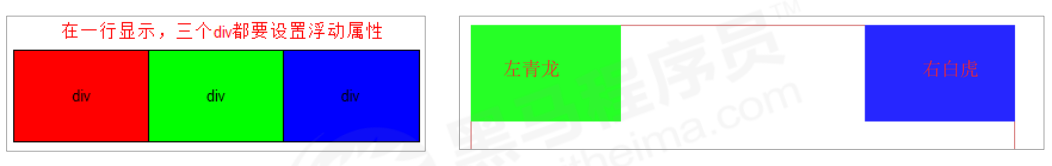
3. 浮动的元素会具有行内块元素的特性

设置了浮动（float）的元素最重要特性：

1. 脱离标准普通流的控制（浮） 移动到指定位置（动）, （俗称脱标）

2. 浮动的盒子不再保留原先的位置

如果多个盒子都设置了浮动，则它们会按照属性值一行内显示并且顶端对齐排列。



**注意：** 浮动的元素是互相贴靠在一起的（不会有缝隙），如果父级宽度装不下这些浮动的盒子， 多出的盒子 会另起一行对齐。

3. 浮动元素会具有行内块元素特性。

任何元素都可以浮动。不管原先是什么模式的元素，添加浮动之后具有**行内块元素**相似的特性。

如果块级盒子没有设置宽度，默认宽度和父级一样宽，但是添加浮动后，它的大小根据内容来决定

**2.2 浮动布局注意点**

**1. 浮动和标准流的父盒子搭配。**

先用标准流的父元素排列上下位置, 之后内部子元素采取浮动排列左右位置

**2. 一个元素浮动了，理论上其余的兄弟元素也要浮动。**

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

浮动的盒子只会影响浮动盒子后面的标准流,不会影响前面的标准流.

**3. 清除浮动**

**3.1 为什么需要清除浮动？**

由于父级盒子很多情况下，不方便给高度，但是子盒子浮动又不占有位置，最后父级盒子高度为 0 时，就会 影响下面的标准流盒子。

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

**3.2 清除浮动本质**

**清除浮动的本质是清除浮动元素造成的影响**

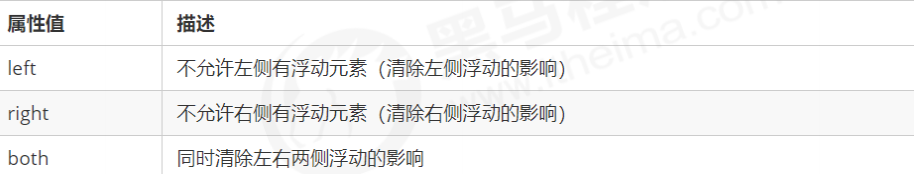
如果父盒子本身有高度，则不需要清除浮动

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

**3.3 清除浮动**

语法：

选择器{clear:属性值;}



我们实际工作中， 几乎只用 clear: both;

清除浮动的策略是: 闭合浮动.

  <div class="box">

    <div>aa</div>

    <div>bb</div>

  </div>

  <div class="shadow">

    .box {

      width: 800px;

    }

    .box div {

      height: 100px;

      width: 200px;

    }

    .box div:nth-child(1) {

      background-color: rebeccapurple;

      float: left;

    }

    .box div:nth-child(2) {

      background-color: aqua;

      float: left;

    }

    .shadow {

      background-color: black;

      height: 300px;

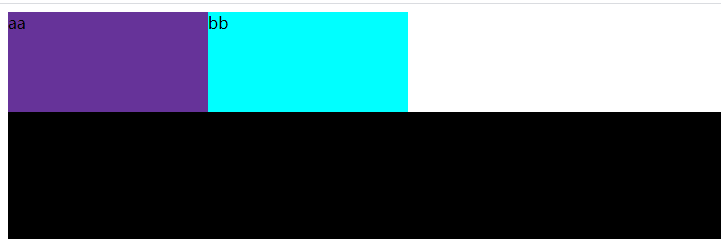
      clear: both;

    }

**不清除浮动**



**清除浮动**



**3.3 清除浮动方法**

**1. 额外标签法**也称为隔墙法，是 W3C 推荐的做法。

2. 父级添加 overflow 属性

3. 父级添加after伪元素

4. 父级添加双伪元素

**3.3 清除浮动 —— 额外标签法**

**额外标签法**也称为隔墙法，是 W3C 推荐的做法。

额外标签法会在浮动元素末尾添加一个空的标签。例如 <div style=”clear:both”></div>，或者其他标签

（如<br />等）。

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

注意： 要求这个新的空标签必须是块级元素。

**3.3 清除浮动 —— 父级添加 overflow**

可以给父级添加 overflow 属性，将其属性值设置为 hidden、 auto 或 scroll 。

子不教,父之过,注意是给父元素添加代码

优点：代码简洁

缺点：无法显示溢出的部分

**3.3 清除浮动 —— :after 伪元素法**

:after 方式是额外标签法的升级版。也是给父元素添加

.clearfix:after {

content: "";

display: block;

height: 0;

clear: both;

visibility: hidden;

}

.clearfix { /\* IE6、7 专有 \*/

\*zoom: 1;

}

优点：没有增加标签，结构更简单

缺点：照顾低版本浏览器

代表网站： 百度、淘宝网、网易等

**3.3 清除浮动 —— 双伪元素清除浮动**

也是给给父元素添加

.clearfix:before,.clearfix:after {

content:"";

display:table;

}

.clearfix:after {

clear:both;

}

.clearfix {

\*zoom:1;

}

优点：代码更简洁

缺点：照顾低版本浏览器

代表网站：小米、腾讯等

**1.11 定位的拓展**

**1. 绝对定位的盒子居中**

加了绝对定位的盒子不能通过 margin:0 auto 水平居中，但是可以通过以下计算方法实现水平和垂直居中。

① left: 50%;：让盒子的左侧移动到父级元素的水平中心位置。

② margin-left: -100px;：让盒子向左移动自身宽度的一半

**2. 定位特殊特性**

绝对定位和固定定位也和浮动类似。

1. 行内元素添加绝对或者固定定位，可以直接设置高度和宽度。

2. 块级元素添加绝对或者固定定位，如果不给宽度或者高度，默认大小是内容的大小。

**3. 脱标的盒子不会触发外边距塌陷**

浮动元素、绝对定位(固定定位）元素的都不会触发外边距合并的问题

**1.11 定位的拓展**

**4. 绝对定位（固定定位）会完全压住盒子**

浮动元素不同，只会压住它下面标准流的盒子，但是不会压住下面标准流盒子里面的文字（图片）

但是绝对定位（固定定位） 会压住下面标准流所有的内容。

浮动之所以不会压住文字，因为浮动产生的目的最初是为了做文字环绕效果的。 文字会围绕浮动元素

**4. 元素的显示与隐藏**

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！

本质：让一个元素在页面中隐藏或者显示出来。

1. display 显示隐藏

2. visibility 显示隐藏

3. overflow 溢出显示隐藏

**4.1 display 属性**

display 属性用于设置一个元素应如何显示。

display: none ；隐藏对象

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

display 隐藏元素后，不再占有原来的位置。

**4.2 visibility 可见性**

visibility 属性用于指定一个元素应可见还是隐藏。

visibility：visible ; 元素可视

visibility：hidden; 元素隐藏

visibility 隐藏元素后，继续占有原来的位置。

如果隐藏元素想要原来位置， 就用 visibility：hidden

如果隐藏元素不想要原来位置， 就用 display：none (用处更多 重点）

**CSS 三角**

网页中常见一些三角形，使用 CSS 直接画出来就可以，不必做成图片或者字体图标。

一张图， 你就知道 CSS 三角是怎么来的了, 做法如下：

div {

width: 0;

height: 0;

line-height: 0;

font-size: 0;

border: 50px solid transparent;

border-left-color: pink;

}

**代码：**

width: 0;

height: 0;

border-color: transparent red transparent transparent;

border-style: solid;

border-width: 22px 8px 0 0;

**4.2 轮廓线 outline**

给表单添加 outline: 0; 或者 outline: none; 样式之后，就可以去掉默认的蓝色边框。

input {outline: none; }

**4.3 防止拖拽文本域 resize**

实际开发中，我们文本域右下角是不可以拖拽的。

textarea{ resize: none;}

**5.2 解决图片底部默认空白缝隙问题**

bug：图片底侧会有一个空白缝隙，原因是行内块元素会和文字的基线对齐。

主要解决方法有两种：

1. 给图片添加 vertical-align:middle | top| bottom 等。 （提倡使用的）

2. 把图片转换为块级元素 display: block

**6. 溢出的文字省略号显示**

**1. 单行文本溢出显示省略号--必须满足三个条件**

/\*1. 先强制一行内显示文本\*/

white-space: nowrap; （ 默认 normal 自动换行）

/\*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;

**2. 多行文本溢出显示省略号**

多行文本溢出显示省略号，有较大兼容性问题， 适合于webKit浏览器或移动端（移动端大部分是webkit内 核）

overflow: hidden;

text-overflow: ellipsis;

/\* 弹性伸缩盒子模型显示 \*/

display: -webkit-box;

/\* 限制在一个块元素显示的文本的行数 \*/

-webkit-line-clamp: 2;

/\* 设置或检索伸缩盒对象的子元素的排列方式 \*/

-webkit-box-orient: vertical;

新增CSS属性（如果以上代码写了之后还不能实现效果，加上下面这句）

word-break: break-all;

**1. margin负值运用**

1.让每个盒子margin 往左侧移动 -1px 正好压住相邻盒子边框

2.鼠标经过某个盒子的时候，提高当前盒子的层级即可（如果没有有定位，则加相对定位（保留位置），如 果有定位，则加z-index）

**2. 文字围绕浮动元素**

巧妙运用浮动元素不会压住文字的特性



**3. 行内块巧妙运用**

页码在页面中间显示:

1. 把这些链接盒子转换为行内块， 之后给**父级**指定 text-align:center;

2. 利用行内块元素中间有缝隙，并且给**父级**添加 text-align:center; 行内块元素会水平会居中

**1. HTML5 的新特性**

<header>：头部标签

<nav>：导航标签

<article>：内容标签

<section>：定义文档某个区域

<aside>：侧边栏标签

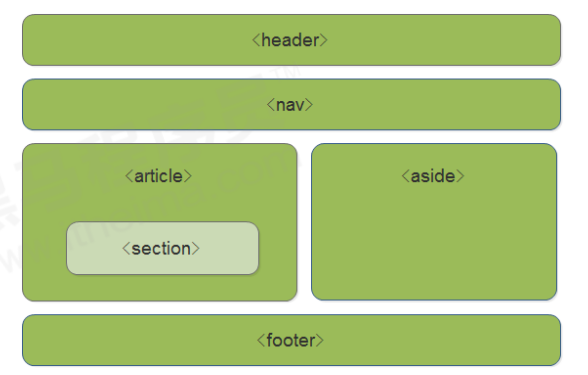
<footer>：尾部标签

**注意：**

这种语义化标准主要是针对搜索引擎的

这些新标签页面中可以使用多次

在 IE9 中，需要把这些元素转换为块级元素



**1.2 HTML5 新增的多媒体标签**

**1. 视频<video>**

当前 <video> 元素支持三种视频格式： MP4、webm、ogg，尽量使用 mp4格式

**语法**

<video src="文件地址" controls="controls"></video>

<video controls="controls" width="300">

<source src="move.ogg" type="video/ogg" >

<source src="move.mp4" type="video/mp4" >

您的浏览器暂不支持 <video> 标签播放视频

</ video >



**2. 音频<audio>**

当前 <audio> 元素支持三种音频格式：MP3、wav、ogg

**语法**

<audio src="文件地址" controls="controls"></audio>

< audio controls="controls" >

<source src="happy.mp3" type="audio/mpeg" >

<source src="happy.ogg" type="audio/ogg" >

您的浏览器暂不支持 <audio> 标签。

</ audio>

**3. 多媒体标签总结**

音频标签和视频标签使用方式基本一致

浏览器支持情况不同

谷歌浏览器把音频和视频自动播放禁止了

我们可以给视频标签添加 muted 属性来静音播放视频，音频不可以（可以通过JavaScript解决）

视频标签是重点，我们经常设置自动播放，不使用 controls 控件，循环和设置大小属性

**1.3 HTML5 新增的 input 类型**

重点记住： number tel search 这三个

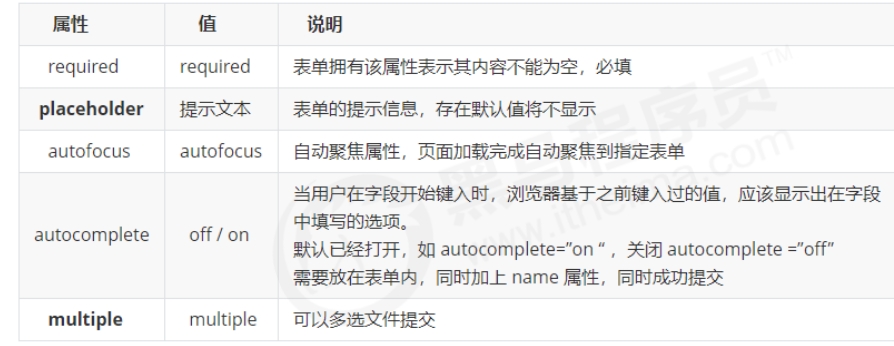
**1.4 HTML5 新增的表单属性**

**可以通过以下设置方式修改placeholder里面的字体颜色：**

input::placeholder {

color: pink;

}



**2.2 属性选择器**

属性选择器可以根据元素特定属性的来选择元素。 这样就可以不用借助于类或者id选择器。

**注意：**类选择器、属性选择器、伪类选择器，权重为 10。



        /\* 必须是input 但是同时具有 value这个属性 选择这个元素  [] \*/

        /\* input[value] {

            color:pink;

        } \*/

        /\* 只选择 type =text 文本框的input 选取出来 \*/

        input[type=text] {

            color: pink;

        }

        /\* 选择首先是div 然后 具有class属性 并且属性值 必须是 icon开头的这些元素 \*/

        div[class^=icon] {

            color: red;

        }

**2.3 结构伪类选择器**

结构伪类选择器主要根据文档结构来选择器元素， 常用于根据父级选择器里面的子元素

**注意：**类选择器、属性选择器、伪类选择器，权重为 10。



区别：

1. nth-child 对父元素里面所有孩子排序选择（序号是固定的） 先找到第n个孩子，然后看看是否和E匹配

2. nth-of-type 对父元素里面指定子元素进行排序选择。 先去匹配E ，然后再根据E 找第n个孩子

**nth-child（n）** 选择某个父元素的一个或多个特定的子元素（重点）

n 可以是数字，关键字和公式

n 如果是数字，就是选择第 n 个子元素， 里面数字从1开始…

n 可以是关键字：even 偶数，odd 奇数

n 可以是公式：常见的公式如下 ( 如果n是公式，则从0开始计算，但是第 0 个元素或者超出了元素的个数会被忽略 )



**小结**

关于 nth-child（n） 我们要知道 n 是从 0 开始计算的，要记住常用的公式

**2.4 伪元素选择器（重点）**

伪元素选择器可以帮助我们利用CSS创建新标签元素，而不需要HTML标签，从而简化HTML结构。

**注意：**

before 和 after 创建一个元素，但是属于行内元素

新创建的这个元素在文档树中是找不到的，所以我们称为伪元素

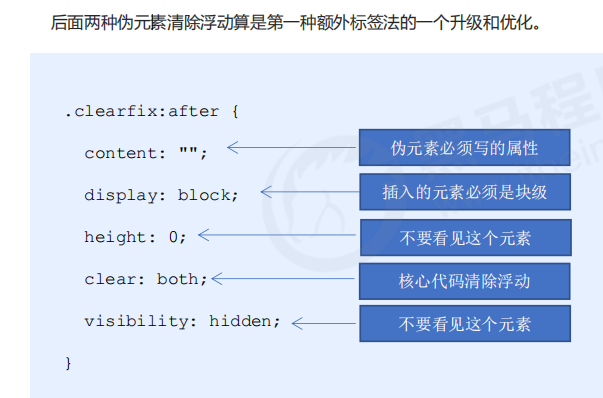
语法： element::before {}

before 和 after 必须有 content 属性

before 在父元素内容的前面创建元素，after 在父元素内容的后面插入元素

伪元素选择器和标签选择器一样，权重为 1





**2.5 CSS3 盒子模型**

CSS3 中可以通过 box-sizing 来指定盒模型，有2个值：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 （以前默认的）

2. box-sizing: border-box 盒子大小为 width

如果盒子模型我们改为了box-sizing: border-box ， 那padding和border就不会撑大盒子了（前提padding 和border不会超过width宽度）

**CSS3滤镜filter:**

filter CSS属性将模糊或颜色偏移等图形效果应用于元素。

filter: 函数(); 例如： filter: blur(5px); blur模糊处理 数值越大越模糊

**CSS3 calc 函数:**

calc() 此CSS函数让你在声明CSS属性值时执行一些计算。

width: calc(100% - 80px);

括号里面可以使用 + - \* / 来进行计算

**2.7 CSS3 过渡（重点）**

过渡（transition)是CSS3中具有颠覆性的特征之一，我们可以在不使用 Flash 动画或 JavaScript 的情况下，当元素从一种样式变换为另一种样式时为元素添加效果。

过渡动画： 是从一个状态 渐渐的过渡到另外一个状态

可以让我们页面更好看，更动感十足，虽然 低版本浏览器不支持（ie9以下版本） 但是不会影响 页面布局。

我们现在经常和 :hover 一起 搭配使用。（鼠标悬浮也会发生过渡效果）

transition: 要过渡的属性 花费时间 运动曲线 何时开始;

**1.属性** transition-property： 想要变化的 css 属性， 宽度高度 背景颜色 内外边距都可以 。如果想要所有的属性都 变化过渡， 写一个all 就可以。

**2. 花费时间**transition-duration**：** 单位是 秒（必须写单位） 比如 0.5s

**3. 运动曲线**transition-timing-function**：** 默认是 ease （可以省略）

**4.何时开始**transition-delay **：**单位是 秒（必须写单位）可以设置延迟触发时间 默认是 0s （可以省略）

记住过渡的使用口诀： 谁做过渡给谁加

        .box {

            background-color: royalblue;

            width: 100px;

            height: 50px;

            transition: all 3s ease 0s;

        }

        .box:hover {

            width: 300px;

            border-radius: 15px;

        }

## propertyName 事件属性

当发生**transitionend**事件时，**propertyName**属性返回与转换关联的CSS属性的名称。此属性是只读的。

**4.2 2D 转换之移动** **translate**

2D移动是2D转换里面的一种功能，可以改变元素在页面中的位置，类似**定位。**

**1. 语法**

transform: translate(x,y); 或者分开写

transform: translateX(n);

transform: translateY(n);

**2. 重点**

定义 2D 转换中的移动，沿着 X 和 Y 轴移动元素

translate最大的优点：不会影响到其他元素的位置

translate中的百分比单位是相对于自身元素的 translate:(50%,50%);

对行内标签没有效果

**4.3 2D 转换之旋转 rotate**

2D旋转指的是让元素在2维平面内顺时针旋转或者逆时针旋转。

**1. 语法**

transform:rotate(度数)

**2. 重点**

rotate里面跟度数， 单位是 deg 比如 rotate(45deg)

角度为正时，顺时针，负时，为逆时针

默认旋转的中心点是元素的中心点

**4.4 2D 转换中心点 transform-origin**

我们可以设置元素转换的中心点

**1. 语法**

transform-origin: x y;

**2. 重点**

注意后面的参数 x 和 y 用空格隔开

x y 默认转换的中心点是元素的中心点 (50% 50%)

还可以给x y 设置 像素 或者 方位名词 （top bottom left right center）

**4.5 2D 转换之缩放scale**

缩放，顾名思义，可以放大和缩小。 只要给元素添加上了这个属性就能控制它放大还是缩小。

**1. 语法**

transform:scale(x,y);

**2. 注意**

注意其中的x和y用逗号分隔

transform:scale(1,1) ：宽和高都放大一倍，相对于没有放大

transform:scale(2,2) ：宽和高都放大了2倍

transform:scale(2) ：只写一个参数，第二个参数则和第一个参数一样，相当于 scale(2,2)

transform:scale(0.5,0.5)：缩小

sacle缩放最大的优势：可以设置转换中心点缩放，默认以中心点缩放的，而且不影响其他盒子

**4.6 2D 转换综合写法**

**注意：**

1. 同时使用多个转换，其格式为：transform: translate() rotate() scale() ...等，

2. 其顺序会影转换的效果。（先旋转会改变坐标轴方向）

**3. 当我们同时有位移和其他属性的时候，记得要将位移放到最前**

**5.1 动画的基本使用**

制作动画分为两步：

1. 先定义动画

2. 再使用（调用）动画

**1. 用keyframes 定义动画（类似定义类选择器）**

@keyframes 动画名称 {

0%{

width:100px;

}

100%{

width:200px;

}

}

**动画序列**

0% 是动画的开始，100% 是动画的完成。这样的规则就是动画序列。

在 @keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。

动画是使元素从一种样式逐渐变化为另一种样式的效果。您可以改变任意多的样式任意多的次数。

请用百分比来规定变化发生的时间，或用关键词 "from" 和 "to"，等同于 0% 和 100%

**2. 元素使用动画**

div {

width: 200px;

height: 200px;

background-color: aqua;

margin: 100px auto;

/\* 调用动画 \*/

animation-name: 动画名称;

/\* 持续时间 \*/

animation-duration: 持续时间;

}



**5.3 动画简写属性**

animation：动画名称 持续时间 运动曲线 何时开始 播放次数 是否反方向 动画起始或者结束的状态;

animation: myfirst 5s linear 2s infinite alternate;

简写属性里面不包含 animation-play-state

暂停动画：animation-play-state: puased; 经常和鼠标经过等其他配合使用

想要动画走回来 ，而不是直接跳回来：animation-direction ： alternate

盒子动画结束后，停在结束位置： animation-fill-mode ： forwards

**5.4 速度曲线细节**

animation-timing-function：规定动画的速度曲线，默认是“ease”

**值**



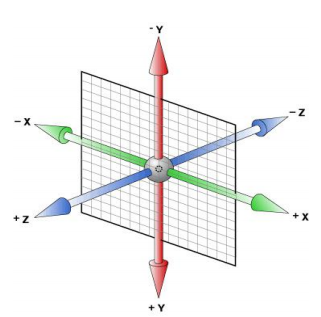
**6.1 三维坐标系**

三维坐标系其实就是指立体空间，立体空间是由3个轴共同组成的。

x轴：水平向右 注意： x 右边是正值，左边是负值

y轴：垂直向下 注意： y 下面是正值，上面是负值

z轴：垂直屏幕 注意： 往外面是正值，往里面是负值



**6.2 3D移动 translate3d**

3D移动在2D移动的基础上多加了一个可以移动的方向，就是z轴方向。

translform:translateX(100px)：仅仅是在x轴上移动

translform:translateY(100px)：仅仅是在Y轴上移动

translform:translateZ(100px)：仅仅是在Z轴上移动（注意：translateZ一般用px单位）

transform:translate3d(x,y,z)：其中 x、y、z 分别指要移动的轴的方向的距离

因为z轴是垂直屏幕，由里指向外面，所以默认是看不到元素在z轴的方向上移动

**6.3 透视 perspective**

在2D平面产生近大远小视觉立体，但是只是效果二维的

如果想要在网页产生3D效果需要透视（理解成3D物体投影在2D平面内）。

模拟人类的视觉位置，可认为安排一只眼睛去看

透视我们也称为视距：视距就是人的眼睛到屏幕的距离

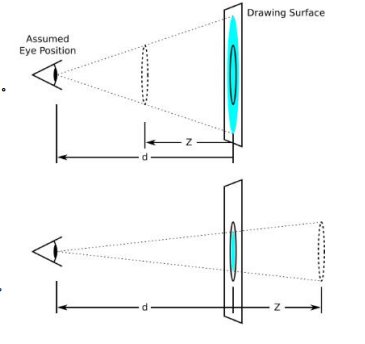
距离视觉点越近的在电脑平面成像越大，越远成像越小

透视的单位是像素

**透视写在被观察元素的父盒子上面的**

**d**：就是视距，视距就是一个距离人的眼睛到屏幕的距离。

**z**：就是 z轴，物体距离屏幕的距离，z轴越大（正值） 我们看到的物体就越大。



**6.5 3D旋转 rotate3d**

3D旋转指可以让元素在三维平面内沿着 x轴，y轴，z轴或者自定义轴进行旋转。

**语法**

transform:rotateX(45deg)：沿着x轴正方向旋转 45度

transform:rotateY(45deg) ：沿着y轴正方向旋转 45deg

transform:rotateZ(45deg) ：沿着Z轴正方向旋转 45deg

transform:rotate3d(x,y,z,deg)： 沿着自定义轴旋转 deg为角度（了解即可）

**6.5 3D旋转 rotate3d**

transform:rotate3d(x,y,z,deg)： 沿着自定义轴旋转 deg为角度（了解即可）

xyz是表示旋转轴的矢量，是标示你是否希望沿着该轴旋转，最后一个标示旋转的角度。

transform:rotate3d(1,0,0,45deg) 就是沿着x轴旋转 45deg

transform:rotate3d(1,1,0,45deg) 就是沿着对角线旋转 45deg

**6.6 3D呈现 transfrom-style**

控制子元素是否开启三维立体环境。。

transform-style: flat 子元素不开启3d立体空间 默认的

transform-style: preserve-3d; 子元素开启立体空间

代码写给父级，但是影响的是子盒子

这个属性很重要，后面必用

**7. 浏览器私有前缀**

浏览器私有前缀是为了兼容老版本的写法，比较新版本的浏览器无须添加。

**1. 私有前缀**

-moz-：代表 firefox 浏览器私有属性

-ms-：代表 ie 浏览器私有属性

-webkit-：代表 safari、chrome 私有属性

-o-：代表 Opera 私有属性

**2. 提倡的写法**

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

-o-border-radius: 10px;

border-radius: 10px;

## Css3新样式

### 伪类选择器 :focus-within

它表示一个元素获得焦点，或，该元素的后代元素获得焦点。划重点，它或它的后代获得焦点。

这也就意味着，它或它的后代获得焦点，都可以触发 :focus-within。

.g-container:focus-within {

transform: translateY(-4px);

box-shadow: 0 0 10px #ddd;

border-color: hsl(199, 98%, 48%);

}

比如g-container类或者该类下的元素获得了焦点，那么这个:focus-within 里写的样式都会触发

### 伪类选择器 :placeholder-shown

当 input 类型标签使用了 placeholder 属性有了默认占位的文字，会触发此伪类样式。就是说输入框placeholder的内容出现，那么该: placeholder-shown 里面写的样式就会触发

### 伪类选择器 :not()

语法： **E:not()**{ sRules }

**匹配不含有s选择符的元素E。**

**比如p:not(.abc){color:pink;} ，p标签中不含有abc类的的元素将被选中。**

### 伪类选择器 :frist-letter

**对段落添加首字下沉效果。**

**p**:first-letter{

*font-size*: 250**%**;

    }

### css中的一些符号

**1、~波浪号**

**A ~ B    选择A标签后的所有B标签，【A和B必须有相同的父元素】**

**2、+加号（兄弟选择器）**

**A+B     选择紧邻在A后面的B元素，【A和B必须拥相同的父元素】所选到的仅为一个B元素标签。**

**3、>大于号**

**A>B      选择A元素里面的B元素【B元素是A元素下的第一代，只第一后代】**

**案例：**

**输入框placeholder存在的时候，**: placeholder-shown是触发的，此时按钮透明度为0，即不显示，当placeholder不存在的时候。not伪类选择器触发了，此时按钮显示。

  .g\_input\_search {

*padding*: 0 15**px**;

*height*: 100**%**;

*width*: 100**%**;

*border*: 1**px** solid #ddd;

*font-size*: 18**px**;

*box-sizing*: border-box;

      &:**not**(:**placeholder-shown**) {

*border*: 1**px** solid #03a9f4;

**+** .g\_button\_search {

*opacity*: 1;

          }

      }

      &:**placeholder-shown** {

**+** .g\_button\_search {

*opacity*: 0;

          }

      }

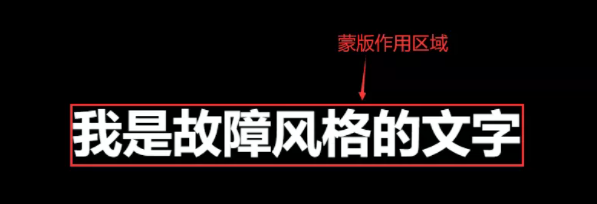
}

### clip-path 蒙版

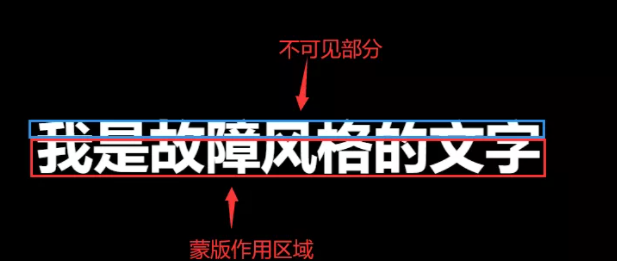
clip-path 是干什么用的呢？这个是css3的一个新属性，叫做「蒙版」，而其中的 inset() 值表示的是蒙版形状为矩形。

首先 inset() 接收四个长度参数，分别表示蒙版距离元素标签的**「上侧」** 、**「右侧」** 、**「下侧」** 、**「左侧」**的距离，从而决定了蒙版的大小

当我们设置为 inset(0 0 0 0)时，表示蒙版作用区域大小跟元素标签一样大，如下图所示（红色边框表示蒙版的作用区域）



然后我们再来看一下，如果我们的设置为 inset(30px 0 0 0) ，则测试表示，蒙版的作用区域距离标签元素的上侧 30px，距离其它的边 0px，如图所示



### 流光效果

 <div class="shadow">文字向右闪过效果</div>

        <style>

            .shadow {

                text-align: center;

                /\* 背景颜色线性渐变 \*/

                    /\* 老式写法 \*/

                        /\* linear为线性渐变，也可以用下面的那种写法。left top，right top指的是渐变方向，左上到右上 \*/

                        /\* color-stop函数，第一个表示渐变的位置，0为起点，0.5为中点，1为结束点；第二个表示该点的颜色。所以本次渐变为两边灰色，中间渐白色 \*/

                background: -webkit-gradient(linear, left top, right top, color-stop(0, #4d4d4d), color-stop(.4, #4d4d4d), color-stop(.5, white), color-stop(.6, #4d4d4d), color-stop(1, #4d4d4d));

                    /\* 新式写法 \*/

                /\* background: -webkit-linear-gradient(left top, right top, color-stop(0, #4d4d4d), color-stop(.4, #4d4d4d), color-stop(.5, white), color-stop(.6, #4d4d4d), color-stop(1, #4d4d4d)); \*/

                /\* 设置为text，意思是把文本内容之外的背景给裁剪掉 \*/

                -webkit-background-clip: text;

                /\* 设置对象中的文字填充颜色 这里设置为透明 \*/

                -webkit-text-fill-color: transparent;

                /\* 每隔2秒调用下面的CSS3动画 infinite属性为循环执行animate \*/

                -webkit-animation: animate 5s infinite;

            }

            /\* 兼容写法，要放在@keyframes前面 \*/

                /\* 背景从-100px的水平位置，移动到+100px的水平位置。如果要移动Y轴的，设置第二个数值 \*/

            @-webkit-keyframes animate {

                from {background-position: -100px;}

                to {background-position: 100px;}

            }

            @keyframes animate {

                from {background-position: -100px;}

                to {background-position: 100px;}

            }

        </style>