

CSS

一 CSS概述

CSS是Cascading Style Sheets的简称，中文称为层叠样式表，用来控制网页数据的表现，可以使网页的表现与数据内容分离。

一 CSS的四种引入方式

1. 行内式

行内式是在标记的style属性中设定CSS样式。这种方式没有体现出CSS的优势，不推荐使用。

```
1 <p style="background-color: rebeccapurple">hello yuan</p>
```

2. 嵌入式

嵌入式是将CSS样式集中写在网页的<head></head>标签对的<style></style>标签对中。格式如下：

```
1 <head>
2   <meta charset="UTF-8">
3   <title>Title</title>
4   <style>
5     p{
6       background-color: #2b99ff;
7     }
8   </style>
9 </head>
```

3 链接式

将一个.css文件引入到HTML文件中

```
1 <link href="mystyle.css" rel="stylesheet" type="text/css"/>
```

4. 导入式

将一个独立的.css文件引入HTML文件中，导入式使用CSS规则引入外部CSS文件，<style>标记也是写在<head>标记中，使用的语法如下：

```
1 <style type="text/css">
2
3   @import "mystyle.css"; 此处要注意.css文件的路径
4
5 </style>
```


注意：

导入式会在整个网页装载完后再装载CSS文件，因此这就导致了一个问题，如果网页比较大则会出现先显示无样式的页面，闪烁一下之后，再出现网页的样式。这是导入式固有的一个缺陷。使用链接式时与导入式不同的是它会以网页文件主体装载前装载CSS文件，因此显示出来的网页从一开始就是带样式的效果的，它不会象导入式那样先显示无样式的网页，然后再显示有样式的网页，这是链接式的优点。

二 CSS的选择器 (Selector)

“选择器”指明了{}中的“样式”的作用对象，也就是“样式”作用于网页中的哪些元素

1 基础选择器

		
* :	通用元素选择器，匹配任何元素	* { margin:0; padding:0; }
E :	标签选择器，匹配所有使用E标签的元素	p { color:green; }
.info 和 E.info:	class选择器，匹配所有class属性中包含info的元素	.info { background:#ff0; } p.info {

#info和E#info id选择器, 匹配所有id属性等于footer的元素

#info { background:#ff0; } p#info {

2 组合选择器

E,F	多元素选择器, 同时匹配所有E元素或F元素, E和F之间用逗号分隔	div,p { color:#f00; }
E F	后代元素选择器, 匹配所有属于E元素后代的F元素, E和F之间用空格分隔	li a { font-weight:bold; }
E > F	子元素选择器, 匹配所有E元素的子元素F	div > p { color:#f00; }
E + F	毗邻元素选择器, 匹配所有紧随E元素之后的同级元素F	div + p { color:#f00; }

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>

    .div1>p{
      background-color: aqua;
      color: deeppink;
    }

    .main2>div{
      background-color: blueviolet;
      color: chartreuse;
    }
  </style>
</head>
<body>

  <div class="div1">hello1
    <div class="div2">hello2
      <div>hello4</div>
      <p>hello5</p>
    </div>
    <p>hello3</p>
  </div>
  <p>hello6</p>

<hr>

  <div class="main2">1
    <div>2
      <div>
        4
      </div>
    </div>
    <div>
      3
    </div>
  </div>
</body>
</html>
```

注意嵌套规则：

- 1. 块级元素可以包含内联元素或某些块级元素，但内联元素不能包含块级元素，它只能包含其它内联元素。
- 2. 有几个特殊的块级元素只能包含内联元素，不能包含块级元素。如h1,h2,h3,h4,h5,h6,p,dt
- 3. li内可以包含div
- 4. 块级元素与块级元素并列、内联元素与内联元素并列。



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    [suoning]{
      color: blueviolet;
    }
    .he>div{
      color: bisque;
    }
  </style>
</head>
<body>

<div class="he">111
  <p class="fr">222
    <div>333</div>
  </p>
  <div>444</div>
</div>

*****

<div suoning="sb">ddd
  <p>pppp</p>
</div>
<p suoning="sb2">ddd2
  <p>pppp2</p>
</p>
</body>
</html>
```


3 属性选择器

	E[att]	匹配所有具有att属性的E元素，不考虑它的值。（注意：E在此处可以省略，比如"[checked]"。以下同。）	
	E[att=val]	匹配所有att属性等于“val”的E元素	div[class="error"] {
	E[att~=val]	匹配所有att属性具有多个空格分隔的值、其中一个值等于“val”的E元素	td[class~="name"] { c
	E[attr^=val]	匹配属性值以指定值开头的每个元素	div[class^="test"] {background:#f
	E[attr\$=val]	匹配属性值以指定值结尾的每个元素	div[class\$="test"] {background:#f
	E[attr*=val]	匹配属性值中包含指定值的每个元素	div[class*="test"] {background:#f
			

4 伪类(Pseudo-classes)

CSS伪类是用来给选择器添加一些特殊效果。

anchor伪类：专用于控制链接的显示效果

	
a:link	(没有接触过的链接)，用于定义了链接的常规状态。
a:hover	(鼠标放在链接上的状态)，用于产生视觉效果。
a:visited	(访问过的链接)，用于阅读文章，能清楚的判断已经访问过的链接。
a:active	(在链接上按下鼠标时的状态)，用于表现鼠标按下时的链接状态。
伪类选择器	：伪类指的是标签的不同状态：

```
a ==> 点过状态 没有点过的状态 鼠标悬浮状态 激活状态

a:link {color: #FF0000} /* 未访问的链接 */

a:visited {color: #00FF00} /* 已访问的链接 */

a:hover {color: #FF00FF} /* 鼠标移动到链接上 */

a:active {color: #0000FF} /* 选定的链接 */ 格式: 标签:伪类名称{ css代码; }
```

View Code

补充:

```
1 | .outer:hover .right{color: red}
```

before after伪类：

:before	p:before	在每个<p>元素之前插入内容
:after	p:after	在每个<p>元素之后插入内容

p:before	在每个 <p> 元素的内容之前插入内容	p:before{content:"hello";color:red}
p:after	在每个 <p> 元素的内容之后插入内容	p:after{ content:"hello"; color:red}

5 css优先级和继承

CSS优先级:

所谓CSS优先级，即是指CSS样式在浏览器中被解析的先后顺序。

样式表中的特殊性描述了不同规则的相对权重，它的基本规则是：

1	内联样式表的权重最高	style="-----1000;
2	统计选择符中的ID属性个数。	#id - - - - -100
3	统计选择符中的CLASS属性个数。	.class - - - - -10
4	统计选择符中的HTML标签名个数。	p - - - - -1

按这些规则将数字串逐位相加，就得到最终的权重，然后在比较取舍时按照从左到右的顺序逐位比较。

```
<style>
  #p{
    color: rebeccapurple;
  }
  .p{
    color: #2459a2;
  }
  p{
    color: yellow;
  }
</style>
<p id="p" class="p" style="color: deeppink">hello yuan</p>
```

CSS的继承性:

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定义了的颜色值也会应用到段落的文本中。

```
1 | body{color:red;}      <p>helloyuan</p>
```

这段文字都继承了由body {color:red;}样式定义的颜色。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0。

```
1 | p{color:green}
```

发现只需要给加个颜色值就能覆盖掉它继承的样式颜色。由此可见：任何显示申明的规则都可以覆盖其继承样式。

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。

```


{
border:1px solid #222
}

<div>hello <p>yuan</p> </div>


```

附加说明:

1、文内的样式优先级为1,0,0,0, 所以始终高于外部定义。这里文内样式指形如<div style="color:red">blah</div>的样式

2、有!important声明的规则高于一切。

3、如果!important声明冲突,则比较优先权。

4、如果优先权一样,则按照在源码中出现的顺序决定,后来者居上。

5、由继承而得到的样式没有specificity的计算,它低于一切其它规则(比如全局选择符*定义的规则)。

三 CSS的常用属性

1 颜色属性

```

1 <div style="color:blueviolet">pppp</div>
2
3 <div style="color:#fee33">pppp</div>
4
5 <div style="color:rgb(255,0,0)">pppp</div>
6
7 <div style="color:rgba(255,0,0,0.5)">pppp</div>

```

2 字体属性

```

1 font-size: 20px/50%/larger
2
3 font-family:'Lucida Bright'
4
5 font-weight: lighter/bold/bolder/
6
7 <h1 style="font-style: oblique">老男孩</h1>

```

3 背景属性

```

background-color: cornflowerblue

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

background-repeat: no-repeat; (repeat:平铺满)

background-position: right top (20px 20px) ; (横向: left center right) (纵向: top center bottom)

简写: <body style="background: 20px 20px no-repeat #ff4 url('1.jpg')">

<div style="width: 300px;height: 300px;background: 20px 20px no-repeat #ff4 url('1.

```

注意: 如果将背景属性加在body上,要记得给body加上一个height,否则结果异常,这是因为body为空,无法撑起背景图片;另外,如果此时要设置一个width = 100px,你也看不出效果,除非你设置出html。

```

<!DOCTYPE html>
<html lang="en">
<head>

```

```

<meta charset="UTF-8">
<title>Title</title>
<style>
    html{
        background-color: antiquewhite;

    }
    body{
        width: 100px;
        height: 600px;
        background-color: deeppink;
        background-image: url(1.jpg);
        background-repeat: no-repeat;
        background-position: center center;
    }
</style>
</head>
<body>

</body>
</html>

```



eg:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>

    <style>

        span{
            display: inline-block;
            width: 18px;
            height: 20px;
            background-image: url("http://dig.chouti.com/images/icon_18_118.png?v=2.13");
            background-position: 0 -100px;
        }
    </style>
</head>
<body>

    <span></span>

</body>
</html>

```



4 文本属性



font-size: 10px;

text-align: center; 横向排列

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 50%:基于字体大小的百分比

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-indent: 150px; 首行缩进

letter-spacing: 10px;

word-spacing: 20px;

text-transform: capitalize;



思考:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>

    .outer .item {
      width: 300px;
      height: 200px;
      background-color: chartreuse;
      display: inline-block;
    }

  </style>
</head>
<body>
  <div class="outer">
    <div class="item" style="vertical-align: top">11
    </div>
    <div class="item">
    </div>
  </div>

  <script>

  </script>
</body>
</html>

```

5 边框属性

```

1 border-style: solid;
2
3 border-color: chartreuse;
4
5 border-width: 20px;
6
7 简写: border: 30px rebeccapurple solid;

```

6 列表属性

```

1 ul,ol{ list-style: decimal-leading-zero;
2         list-style: none; <br>         list-style: circle;
3         list-style: upper-alpha;
4         list-style: disc; }

```

7 display属性

```

1 none
2 block
3 inline

```

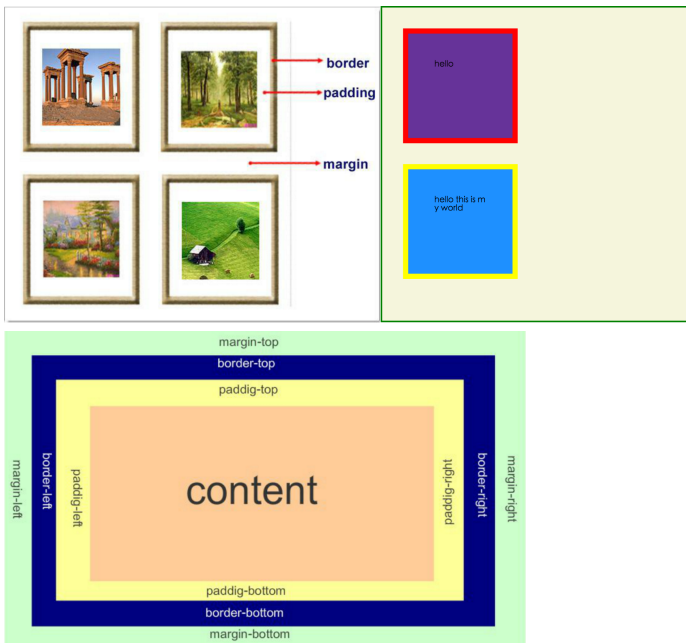
display:inline-block可做列表布局，其中的类似于图片间的间隙小bug可以通过如下设置解决：

```

1 #outer{
2         border: 3px dashed;
3         word-spacing: -5px;
4     }

```

8 外边距和内边



- **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。
- **padding:** 用于控制内容与边框之间的距离；
- **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。

元素的宽度和高度:

💡**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，完全大小的元素，你还必须添加填充，边框和边距。.

```
margin:10px 5px 15px 20px;-----上 右 下 左
margin:10px 5px 15px;-----上 右左 下
margin:10px 5px;-----上下 右左
margin:10px; -----上下左右
```

下面的例子中的元素的总宽度为300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

练习: 300px*300px的盒子装着100px*100px的盒子，分别通过margin和padding设置将小盒子 移到大盒子的中间

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>

    .div1{
      background-color: aqua;
      width: 300px;
      height: 300px;
    }
    .div2{
      background-color: blueviolet;
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>

```



```

    <div class="div1">
      <div class="div2"></div>
      <div class="div2"></div>
    </div>
  </body>
</html>

```



思考1:

边框在默认情况下会定位于浏览器窗口的左上角，但是并没有紧贴着浏览器的窗口的边框，这是因为body本身也是一个盒子（外层还有html），在默认情况下，body距离html会有若干像素的margin，具体数值因各个浏览器不尽相同，所以body中的盒子不会紧贴浏览器窗口的边框了，为了验证这一点，加上：

```

1  body{
2    border: 1px solid;
3    background-color: cadetblue;
4  }

```

>>>>解决方法：

```

1  body{
2    margin: 0;
3  }

```

思考2:

margin collapse（边界塌陷或者说边界重叠）

外边距的重叠只产生在普通流文档的上下外边距之间，这个看起来有点奇怪的规则，其实有其现实意义。设想，当我们上下排列一系列规则的块级元素（如段落P）时，那么块元素之间因为外边距重叠的存在，段落之间就不会产生双倍的距离。又比如停车场

1兄弟div：上面div的margin-bottom和下面div的margin-top会塌陷，也就是会取上下两者margin里最大值作为显示值

2父子div

if 父级div中没有 border, padding, inline content, 子级div的margin会一直向上找，直到找到某个标签包括border, padding, inline content 中的其中一个，然后按此div 进行margin；

```

<!DOCTYPE html>
<html lang="en" style="padding: 0px">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>

    body{
      margin: 0px;
    }

    .div1{
      background-color: aqua;
      width: 300px;
      height: 300px;
    }
    .div2{
      background-color: blueviolet;
      width: 100px;
      height: 100px;
      margin: 20px;
    }
  </style>
</head>
<body>

  <div style="background-color: cadetblue;width: 300px;height: 300px"></div>

  <div class="div1">
    <div class="div2"></div>
  </div>

```

```
<div class="div2"></div>
</div>

</body>
</html>
```



解决方法:

```
1: border:1px solid transparent
2: padding:1px
3: over-flow:hidden;
```

9 float属性

先来了解一下block元素和inline元素在文档流中的排列方式。

block元素通常被现实为独立的一块，独占一行，多个block元素会各自新起一行，默认block元素宽度自动填满其父元素宽度。block元素可以设置width、height、margin、padding属性；

inline元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化。inline元素设置width、height属性无效。inline元素的margin和padding属性。水平方向的padding-left, padding-right, margin-left, margin-right都产生边距效果；但垂直方向的padding-top, padding-bottom, margin-top, margin-bottom不会产生边距效果。

- 常见的块级元素有 div、form、table、p、pre、h1~h5、dl、ol、ul 等。
- 常见的内联元素有span、a、strong、em、label、input、select、textarea、img、br等

所谓的文档流，指的是元素排版布局过程中，元素会自动从左往右，从上往下的流式排列。

脱离文档流，也就是将元素从普通的布局排版中拿走，其他盒子在定位的时候，会当做脱离文档流的元素不存在而进行定位。

只有绝对定位absolute和浮动float才会脱离文档流。

---部分无视和完全无视的区别？需要注意的是，使用float脱离文档流时，其他盒子会无视这个元素，但其他盒子内的文本依然会在这个元素让出位置，环绕在周围(可以说是部分无视)。而对于使用absolute position脱离文档流的元素，其他盒子与其他盒子内的文本都会无视它。(可以说是完全无视)

浮动的表现

定义：浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，所以文档的普通流中的浮动框之后的块框表现得就像浮动框不存在一样。（注意这里是块框而不是内联元素；浮动框只对它后面的元素造成影响）

注意 当初float被设计的时候就是用来完成文本环绕的效果，所以文本不会被挡住，这是float的特性，即float是一种不彻底的脱离文档流方式。无论多么复杂的布局，其基本出发点均是：“如何在一行显示多个div元素”。

现象1:

假如某个div元素A是浮动的，如果A元素上一个元素也是浮动的，那么A元素会跟随在上一个元素的后边(如果一行放不下这两个元素，那么A元素会被挤到下一行)；如果A元素上一个元素是标准流中的元素，那么A的相对垂直位置不会改变，也就是说A的顶部总是和上一个元素的底部对齐。此外，浮动的框之后的block元素元素会认为这个框不存在，但其中的文本依然会在这个元素让出位置。浮动的框之后的inline元素，会为此框空出位置，然后按顺序排列。

现象2:

(1)左右结构div盒子重叠现象，一般是由于相邻两个DIV一个使用浮动一个没有使用浮动。如上面的例1：相邻的两个盒子box2向左浮动、box3未浮动。一个使用浮动一个没有导致DIV不是在同个“平面”上，但内容不会照成覆盖现象，只有DIV形成覆盖现象。

解决方法：要么都不使用浮动；要么都使用float浮动；要么对没有使用float浮动的DIV设置margin样式。

(2)上下结构div盒子重叠现象



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

<title>Title</title>
<style type="text/css">
    * {
        margin:0;padding:0;
    }
    .container{
        border:1px solid red;width:300px;
    }
    #box1{
        background-color:green;float:left;width:100px;height:100px;
    }
    #box2{
        background-color:deeppink; float:right;width:100px;height:100px;
    }
    #box3{
        background-color:pink;height:40px;
    }
</style>
</head>
<body>

    <div class="container">
        <div id="box1">box1 向左浮动</div>
        <div id="box2">box2 向右浮动</div>
    </div>
    <div id="box3">box3</div>
</body>
</body>
</html>

```

例子上如：.container和box3的布局是上下结构，上图发现box3跑到了上面，与.container产生了重叠，但文本内容没有发生覆盖，只有div发生覆盖现象。这个原因是因为第一个大盒子里的子元素使用了浮动，脱离了文档流，导致.container没有被撑开。box3认为.container没有高度（未被撑开），因此跑上去了。

解决方法：

- 1、要么给.container设置固定高度，一般情况下文字内容不确定多少就不能设置固定高度，所以一般不能设置“.container”高度(当然能确定内容多高，这种情况下“.container是可以设置一个高度即可解决覆盖问题。
- 2、要么清除浮动。

清除浮动：

在非IE浏览器（如Firefox）下，当容器的高度为auto，且容器的内容中有浮动（float为left或right）的元素，在这种情况下，容器的高度不能自动伸长以适应内容的高度，使得内容溢出到容器外面而影响（甚至破坏）布局的现象。这个现象叫浮动溢出，为了防止这个现象的出现而进行的CSS处理，就叫CSS清除浮动。



clear语法：

clear : none | left | right | both

取值：

none : 默认值。允许两边都可以有浮动对象

left : 不允许左边有浮动对象

right : 不允许右边有浮动对象

both : 不允许有浮动对象

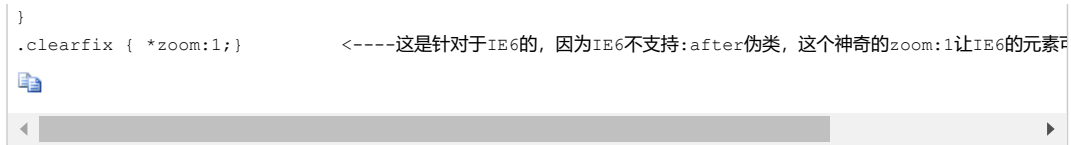
但是需要注意的是：clear属性只会对自身起作用，而不会影响其他元素。如果一个元素的右侧有一浮动对象，而这个元素设置了不允许右边有浮动对象，即clear: right，则这个元素会自动下移一格，达到本元素右边没有浮动对象的目的。



方式1(推荐):



<pre> .clearfix:after { content: "."; display: block; clear: both; visibility: hidden; line-height: 0; height: 0; font-size: 0; </pre>	<pre> <----在类名为“clearfix”的元素内最后面加入内容； <----内容为“.”就是一个英文的句号而已。也可以不写。 <----加入的这个元素转换为块级元素。 <----清除左右两边浮动。 <----可见度设为隐藏。注意它和display:none;是有区别的。visibility:hidden; <----行高为0； <----高度为0； <----字体大小为0； </pre>
--	--



整段代码就相当于在浮动元素后面跟了个宽高为0的空div，然后设定它clear:both来达到清除浮动的效果。之所以用它，是因为，你不必在html文件中写入大量无意义的空标签，又能清除浮动。

话说回来，你这段代码真是累赘啊，这样写不利于维护。

只要写一个.clearfix就行了，然后在需要清浮动的元素中 添加clearfix类名就好了。

如：

```
1 <div class="head clearfix"></div>
```

方式2:

```
1 overflow:hidden;
```

overflow: hidden的含义是超出的部分要裁切隐藏，float的元素虽然不在普通流中，但是他是浮动在普通流之上的，可以把普通流元素+浮动元素想象成一个立方体。如果没有明确设定包含容器高度的情况下，它要计算内容的全部高度才能确定在什么位置hidden，这样浮动元素的高度就要被计算进去。这样包含容器就会被撑开，清除浮动。

10 position(定位)

1 static

static 默认值，无定位，不能当作绝对定位的参照物，并且设置标签对象的left、top等值是不起作用的。

2 position: relative / absolute

relative 相对定位。相对定位是相对于该元素在文档流中的原始位置，即以自己原始位置为参照物。有趣的是，即使设定了元素的相对定位以及偏移值，元素还占有着原来的位置，即占据文档流空间。**对象遵循正常文档流**，但将依据top, right, bottom, left等属性在正常文档流中偏移位置。而其层叠通过z-index属性定义。

注意: position: relative的一个主要用法: 方便绝对定位元素找到参照物。

absolute 绝对定位。

定义：设置为绝对定位的元素框从文档流完全删除，并相对于最近的已定位祖先元素定位，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块（即body元素）。元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

重点：如果父级设置了position属性，例如position:relative;，那么子元素就会以父级的左上角为原始点进行定位。这样能很好的解决自适应网站的标签偏离问题，即父级为自适应的，那我子元素就设置position:absolute;父元素设置position:relative;，然后Top、Right、Bottom、Left用百分比宽度表示。

另外，对象脱离正常文档流，使用top, right, bottom, left等属性进行绝对定位。而其层叠通过z-index属性定义。

总结：参照物用相对定位，子元素用绝对定位，并且保证相对定位参照物不会偏移即可。

3 position:fixed

fixed：对象脱离正常文档流，使用top, right, bottom, left等属性以窗口为参考点进行定位，当出现滚动条时，对象不会随着滚动。而其层叠通过z-index属性 定义。 注意点： 一个元素若设置了 position:absolute | fixed; 则该元素就不能设置float。这 是一个常识性的知识点，因为这是两个不同的流，一个是浮动流，另一个是“定位流”。但是 relative 却可以。因为它原本所占的空间仍然占据文档流。

在理论上，被设置为fixed的元素会被定位于浏览器窗口的一个指定坐标，不论窗口是否滚动，它都会固定在这个位置。

4 仅使用margin属性布局绝对定位元素

此情况，margin-bottom 和margin-right的值不再对文档流中的元素产生影响，因为该元素已经脱离了文档流。

另外，不管它的祖先元素有没有定位，都是以文档流中原来所在的位置上偏移参照物。

图9中，使用margin属性布局绝对定位元素。

层级关系为：

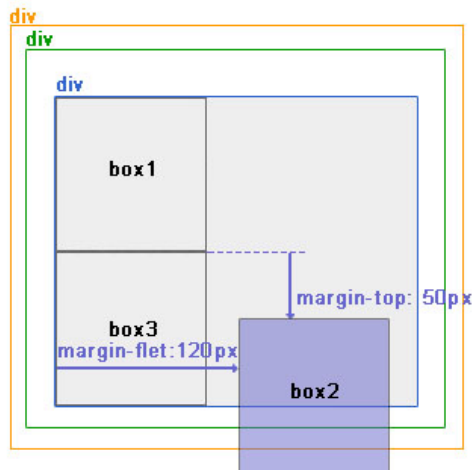
```

<div ————— position:relative;
<div ————— 没有设置为定位元素，不是参照物
<div ————— 没有设置为定位元素，不是参照物
<div box1
<div box2 ——— position:absolute; margin-top:50px; margin-left:120px;

```

<div box3

效果图:



11 关于css的拾遗

11.1 inline-block 的间隙

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    a{
      display: inline-block;
      background-color: #2459a2;
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>

<a>111</a>
<a>222</a>
<a>333</a>

</body>
</html>

```

inline-block默认的空格符就是标签与标签之间的空隙造成的。

(1) 我们可以通过margin:-3px来解决, 但是

- 1.我们布局肯定很多元素, 不可能每个都添加margin负这样维护成本太大了
- 2.我们线上代码如果压缩, 那么我们就存在哪个4px的问题了, 那么我们的margin负就回造成布局混乱!

(2)我们可以给几个标签加一个父级div, 然后:

```
1 | div{word-spacing: -5px;}
```

11.2 word-wrap & word-break:

word-wrap:

the word-wrap CSS property is used to specify whether or not the browser is allowed to break lines within words in order to prevent overflow when an otherwise unbreakable string is too long to fit.

The word-wrap property was invented by Microsoft and added to CSS3. It allows long words to be able to be broken and wrap onto the next line. It takes in two values; normal or break-word.

word-break:

The word-break CSS property is used to specify how (or if) to break lines within words

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>

    .p1{
      width: 100px;
      background-color: #84a42b;
      word-wrap: break-word;
      word-break: break-all;
    }
    .p2{
      width: 100px;
      background-color: darkgoldenrod;
      word-wrap: break-word;
      word-break: break-all;
    }
  </style>
</head>
<body>

<p class="p1">hello yuan hello yuan hello yuan hello yuan hello yuan</p>

<p class="p2">helloyuanhelloyuanhelloyuanhelloyuanhelloyuanhelloyuan</p>

</body>
</html>
```

1.3

一旦给元素加上absolute或float就相当于给元素加上了display:block;。什么意思呢? 比如内联元素span默认宽度是自适应的, 你给其加上width是不起作用的。要想width定宽, 你需要将span设成display:block。但如果你给span加上absolute或float, 那span的display属性自动就变成block, 就可以指定width了。

1.4 快写

```

1 | div    tab
2 | a      tab
3 | div.main>ul>li.c*4    tab
```

1.5 display:flex

1.6 响应式布局

参考文献:

<http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html>

<http://www.jianshu.com/p/a3da5e27d22b>

好文要顶

已关注

收藏该文



Yuan先生

关注 - 1

粉丝 - 3966

我在关注他 取消关注

8

0

Post Comment

#1楼 2018-03-16 09:47 | 闹世闲人

mark

支持(3) 反对(0)

刷新评论 刷新页面 返回顶部

- 【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】了不起的开发者，挡不住的华为，园子里的品牌专区
- 【推荐】九大训练营同开 2020阿里云大数据独门绝学
- 【推荐】精品问答: Java 技术 1000 问

相关博文:

- CSS: CSS学习总结
- CSS float和CSS clear
- [CSS] CSS实践--CSS Reset
- 【css】动态的 css——less
- CSS学习: 编写CSS文件
- » 更多推荐...

最新 IT 新闻:

- 天猫超市上线鲜花品类: 全程0°C冷链配送 七夕预订每支玫瑰不到5元
- 微软《飞行模拟》有惊喜地图: 探索神秘51区、切尔诺贝利废墟等
- 腾讯在成都设立新总部: 投资50亿元 主攻游戏、动漫、视频
- 拿到腾讯字节快手offer后, 他的LeetCode刷题经验在GitHub上收获1.3k星
- 阿里招募“看海官”: 工作0小时 年薪一片海
- » 更多新闻...