# CSS hack

CSS hack 简单的说，就是使你的CSS代码兼容不同的浏览器。  
当然，我们也可以反过来利用CSS hack为不同版本的浏览器定制编写不同的CSS效果。  
原因：  
 不同厂商的浏览器，比如Internet Explorer,Safari,Mozilla Firefox,Chrome等，或者同一厂商的浏览器的不同版本，如IE6和IE7，对CSS的解析认识不完全一样，  
结果：  
 导致生成的页面效果不一样，得不到我们所需要的页面效果。  
解决：  
 需要针对不同的浏览器去写不同的CSS，让它能够同时兼容不同的浏览器，能在不同的浏览器中也能得到我们想要的页面效果。  
参考百度百科：http://baike.baidu.com/item/css%20hack

## 判断IE浏览器版本的条件表达式

|  |  |  |
| --- | --- | --- |
| **项目** | **范例** | **说明** |
| ! | [if !IE] | The NOT operator. This is placed immediately in front of the *feature*, *operator*, or *subexpression* to reverse the Boolean meaning of the expression. NOT运算符。这是摆立即在前面的*功能*，*操作员*，或*子表达式*扭转布尔表达式的意义。 |
| lt | [if lt IE 5.5] | The less-than operator. Returns true if the first argument is less than the second argument. 小于运算符。如果第一个参数小于第二个参数，则返回true。 |
| lte | [if lte IE 6] | The less-than or equal operator. Returns true if the first argument is less than or equal to the second argument. 小于或等于运算。如果第一个参数是小于或等于第二个参数，则返回true。 |
| gt | [if gt IE 5] | The greater-than operator. Returns true if the first argument is greater than the second argument. 大于运算符。如果第一个参数大于第二个参数，则返回true。 |
| gte | [if gte IE 7] | The greater-than or equal operator. Returns true if the first argument is greater than or equal to the second argument. 大于或等于运算。如果第一个参数是大于或等于第二个参数，则返回true。 |
| ( ) | [if !(IE 7)] | Subexpression operators. Used in conjunction with boolean operators to create more complex expressions. 子表达式运营商。在与布尔运算符用于创建更复杂的表达式。 |
| & | [if (gt IE 5)&(lt IE 7)] | The AND operator. Returns true if all subexpressions evaluate to true AND运算符。如果所有的子表达式计算结果为true，返回true |
| | | [if (IE 6)|(IE 7)] | The OR operator. Returns true if any of the subexpressions evaluates to true. OR运算符。返回true，如果子表达式计算结果为true。 |

在WebStorm中直接写cc:ie加tab键就可以生成表达式：

<!--[if IE]>

中间写内容

<![endif]-->

上面表达式不完全，如果想要在IE8以下的显示中间的内容，表达式为<!--[if lt IE 8]>，IE与8之间有空格，可以利用上表中的表达式在不同IE版本中添加内容，现在测试只在IE10（不包括）以下可以生效

## CSS3PIE

1、CSS3PIE 官网

http://css3pie.com/

2、CSS3PIE 让IE支持的CSS3的新特性（即CSS新的属性）

http://css3pie.com/documentation/supported-css3-features/

border-radius --- IE 6-8. IE 9

box-shadow --- IE 6-8. IE 9

border-image --- ？

Gradients --- ？

RGBA Color Values --- IE 9

3、集成到WEB项目中，

3.1、在其官网下载压缩文件，解压缩，复制到项目指定目录

3.2、在选择器中引入即可。例如

behavior: url("HACK/PIE-1.0.0/PIE.htc");

具体代码：

*<!--引入PIE的JS库-->*<**script type="text/javascript" src="相对路径/PIE\_IE678\_uncompressed.js"**></**script**>

<**style type="text/css"**>  
 .**box** {  
 **margin**: 200**px** 0 0 200**px**;  
 **width**: 200**px**;  
 **height**: 200**px**;  
 **background-color**: **orange**;  
 **border-radius**: 50%;  
 **box-shadow**: 0 0 50**px** 0 **red**;  
 */\*引入htc文件，让IE678支持圆角、阴影特性\*/* **behavior**: **url**(**"相对路径/PIE.htc"**);  
 }

# 渐变背景色

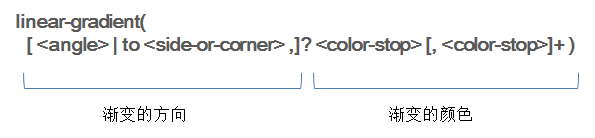
## 线性渐变

* 线性渐变：颜色沿着一条直线轴变换。

渐变函数

### linear-gradient()

* 函数创建了一个呈现线性渐变的颜色的<image>。



* linear-gradient( to right, blue, red );
  + 颜色渐变方向：从左->右。默认是从上向下
* linear-gradient( to left top, blue, red);
  + 颜色渐变方向：从右下角 -> 左上角
* linear-gradient( 45deg, blue, red );

基于0deg顺时针旋转45度。（注意：默认是180deg）当设置0deg时，不设置渐变方向，此时颜色渐变是从下往上

* 颜色渐变距离：设置每个颜色过渡的起始位置

background: linear-gradient(

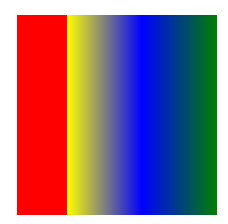
to right,

blue 10px, /\*蓝色从10px（根据元素渐变方向计算）开始有渐变效果\*/

red 50px /\*红色从50px开始有渐变效果\*/

);

**background-image**: **linear-gradient**(**to right**, **red** 50**px**, **yellow** 50**px**, **blue**, **green**);



设置红色从50px开始渐变，50px前面的不渐变，黄色也是从50px开始渐变，两个颜色中间没有位置，所以50px位置红色和黄色没有渐变

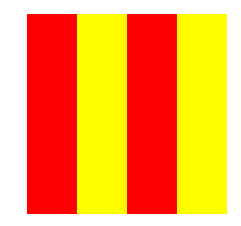
**background-image**: **linear-gradient**(**to right**, **red** 50**px**, **yellow** 50**px**, **yellow** 100**px**, **blue**, **green**);



红色是一条的和上面是一样的，而黄色从50px和100px开始渐变的，50px位置到100px位置都是黄色，所以也是一条。利用这一特性可以设置彩条相间的效果

.**box** {  
 **margin**: 200**px** 0 0 200**px**;  
 **width**: 200**px**;  
 **height**: 200**px**;  
 **background-image**: **linear-gradient**(**to right**, **red** 0,  
 **red** 50**px**, **yellow** 50**px**, **yellow** 100**px**, **red** 100**px**, **red** 150**px**, **yellow** 150**px**, **yellow** 200**px**);

}

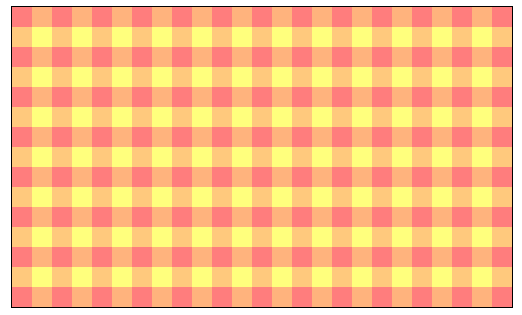


### repeating-linear-gradient

* 重复线性渐变
* 利用重复线性渐变可以简化上面代码
* **background-image**: **repeating-linear-gradient**(  
   **to right**, **red** 0,  
   **red** 50**px**, **yellow** 50**px**, **yellow** 100**px**);

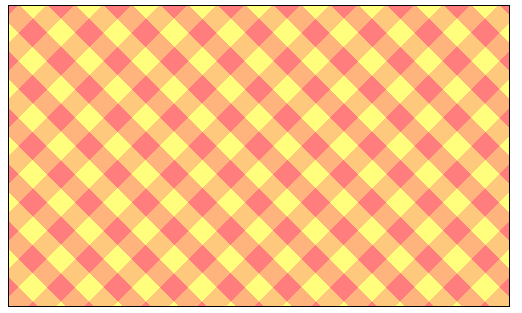
可以写多个线性渐变，造成重叠效果

**background-image**: **repeating-linear-gradient**(  
 **rgba**(255, 0, 0, .3) 0  
 , **rgba**(255, 0, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 40**px** ), **repeating-linear-gradient**( **to right**,  
 **rgba**(255, 0, 0, .3) 0  
 , **rgba**(255, 0, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 40**px** );



也可以选择一个角度

**background-image**: **repeating-linear-gradient**(45**deg**,  
 **rgba**(255, 0, 0, .3) 0  
 , **rgba**(255, 0, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 40**px** ), **repeating-linear-gradient**( -45**deg**,  
 **rgba**(255, 0, 0, .3) 0  
 , **rgba**(255, 0, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 20**px** , **rgba**(255, 255, 0, .3) 40**px** );



### radial-gradient()

* 函数创建了一个呈现径向渐变的颜色的<image>。
* 圆形或椭圆形渐变。颜色不再沿着一条直线轴变换，而是从一个“圆心”向外扩散。
* 径向渐变只有两种形状：椭圆形，圆形

基本用法：

radial-gradient(red, blue);

**设置圆心位置 可以使用px、百分比、关键字**

radial-gradient(

circle at 100px 100px ,

red,

blue

);

**设置为圆形**

radial-gradient(

circle,

red,

blue

);

修改渐变的半径

默认：基于元素宽度的1/2作为其水平及垂直半径

**设置圆形的半径**

background: radial-gradient(

circle **200px** at 100px 50px,

#fff,

#000

);

**设置为椭圆形**

background: radial-gradient(

ellipse,

blue 20px,

red 30px ,

yellow 50px

);

**设置椭圆形的圆心**

background: radial-gradient(

ellipse at 0px 0px,

blue 20px,

red 30px ,

yellow 50px

);

**设置椭圆形X，Y轴半径**

background: radial-gradient(

ellipse 50px 100px at 100px 50px,

#fff,

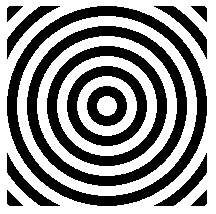
#000

);

### repeating-radial-gradient()

重复径向渐变

background: repeating-radial-gradient(



**circle**,

transparent 0,

transparent 10px,

black 10px,

black 20px

);

# CSS3文本

## 文本换行

* overflow-wrap : 浏览器为了防止文本溢出，是否可以在单词内换行。
  + normal 默认情况，浏览器只在“空格”或者“半角”的位置换行。
  + **break-word** 为防止长文本（例如网页地址、长的单词）溢出，浏览器自行决定在何处“截断”单词。默认情况是禁止浏览器截断单词。当水平方向不足以放下长单词，根据水平宽度截断文本
* word-break: 让文本在任何位置换行
  + normal 根据语言自己的规则确定换行方式，中文到边界上的汉字换行，英文整个单词换行。
  + **break-all** 为了保证每一行的**空间不浪费**，强行“截断”英文单词。在整个一行如果最后一个单词放不下，截断单词，能放多少放多少
  + **keep-all**  不准许“截断”。如果是中文把前后标点符号（全角）后开始换行，英文单词整个换行。 当水平方向不足以放下长单词，不截断单词，使单词溢出父元素
  + IE6及以上浏览器对该属性支持。
* white-space： 处理元素中的空白符（CSS3新增）
  + normal 默认值。空白处会被浏览器忽略（即多余空格被浏览器删除）。只保留正常的空格。
  + **pre**： 文本空白会被浏览器保留。其行为方式类似HTML的<pre>元素效果。
  + **pre-line**: 与normal类似，空白处会被浏览器忽略。不同点是保留换行符，IE7及以下浏览器不支持此属性。
  + **pre-wrap**: 保留空白符序列，换行单独一行显示，IE7及以下浏览器不支持此属性。
  + **nowrap**: 文本不换行，文本在同一行显示，直到遇到<br/>位置。空白处会被浏览器忽略。使用场景： 广告语，文章标题

## 文本阴影

* text-shadow: offset-x | offset-y | blur-radius | color
  + offset-x X轴偏移量
  + offset-y Y轴偏移量
  + blur-radius 模糊半径
  + color 阴影颜色
* IE8及以下浏览器兼容方案。
* fliter: shadow([Color], [Direction], [Strength]);
  + Color 阴影颜色
  + Direction 设定投影的方向。0 阴影在文本上；45度阴影在文本右上角。
  + Strength

.**box**:**hover** {  
 */\*多重阴影\*/* **text-shadow**:  
 0 0 30**px blue**,  
 0 0 30**px #fff**,  
 0 0 80**px blue**,  
 0 0 80**px #fff**,  
 0 0 180**px blue**,  
 0 0 80**px #fff**;  
 **transition**: 2**s**;  
}



## 文本溢出

* **text-overflow: ellipsis**; 当文本溢出时，省略号显示。
* **text-overflow: clip**; 仅仅简单的剪裁，不使用省略号。
* 注意：上述两个属性需要配合 white-space: nowrap; 和 overflow: hidden 一起使用，且容器需要定义宽度。只在块元素内生效。
* IE浏览器对其支持非常好。

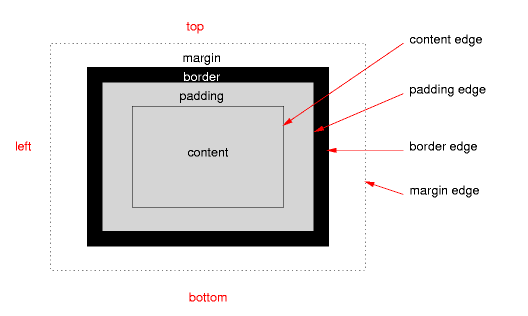
当使用省略号时，如果需要显示完整文本，可以加一个title属性，属性值为完整文本，当鼠标悬浮上去时，可以显示title中的内容

# 布局及盒模型

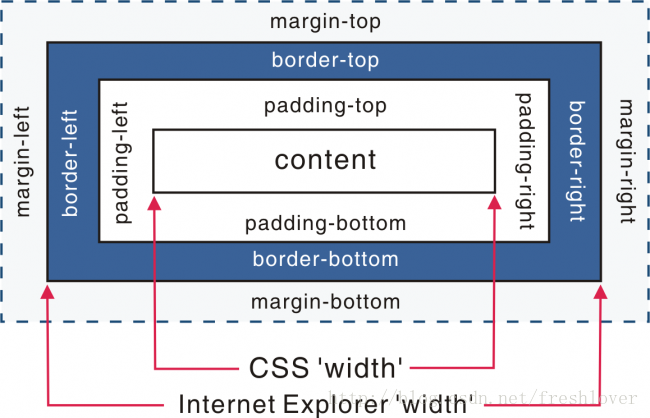
* 根据实际需求，把HTML元素通过CSS样式显示在网页中。

1. PC端布局特点
   * 会为网页设置一个默认宽度，倾向像素单位（px）
2. 移动端布局特点
   * 基于视口作为网页宽度，更倾向使用相对单位（% , rem，vh, wh 等）

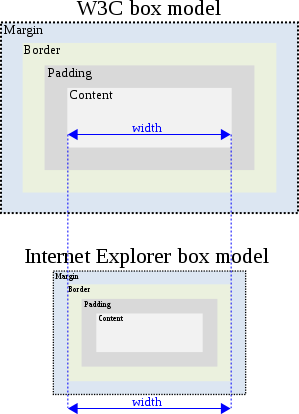
## 标准盒模型



## 怪异盒模型



## 标准盒模型VS怪异盒模型



## box-sizing

* + box-sizing: border-box;元素设置宽度按 border-box 宽度计算
  + box-sizing: content-box; （默认值），元素设置宽度按 content-box宽度计算

## 计算函数calc()

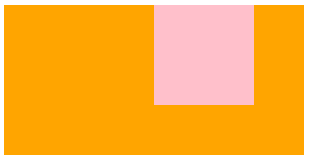
**width**: **calc**(200**px** - 1**px**);  
**width**: **calc**(200**px** + 100**px**);  
**width**: **calc**(200**px** / 2);

运算符左右一定要添加空格符

## 右外边距失效

当父元素水平空间不足，设置右外边距失效，优先保证左外边距

.**box-wrapper** {  
 **width**: 300**px**;  
 **height**: 150**px**;  
 **background-color**: **orange**;  
}  
.**box**{  
 **width**: 100**px**;  
 **height**: 100**px**;  
 **background-color**: **pink**;  
 **margin**: 0 150**px**;  
}

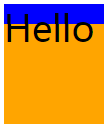


## 负外边距

* 使用负外边会产生一个元素“悬浮”在另一个元素上面的效果。
* 注意：被覆盖的元素文本内容不会被覆盖！

.**box1** {  
 **width**: 100**px**;  
 **height**: 100**px**;  
 **background-color**: **blue**;  
 **font-size**: 36**px**;  
}  
.**box2** {  
 **width**: 100**px**;  
 **height**: 100**px**;  
 **background-color**: **orange**;  
 **margin-top**: 0**px**;  
}





.**box1** {  
 **width**: 100**px**;  
 **height**: 100**px**;  
 **background-color**: **blue**;  
 **font-size**: 36**px**;  
 }  
 .**box2** {  
 **width**: 100**px**;  
 **height**: 100**px**;  
 **background-color**: **orange**;  
 **margin-top**: -80**px**;  
 }

* **使用场景：左侧自适应，右侧固定宽度布局效果**
  1. **为浮动元素（.box）设置右侧外边距为负**
  2. **右侧外边距绝对值等于紧邻兄弟元素的宽度**
  3. **为紧邻兄弟元素设置左浮动**
  4. **注意事项： .box如果有文本，需要添加一个子元素，将文本放入子元素中，子元素外边距为正值，该值等于父元素的负边距的绝对值。同时不要给子元素指定宽度！！！，通过上述处理，就不会出现覆盖内容的问题！**

## BFC

* 设置元素浮动，紧邻兄弟元素“占有”该元素位置，通过开启紧邻兄弟元素BFC（overflow: hidden），防止其被浮动元素覆盖。
* **使用场景：左侧固定，右侧自适应**

1.第一列设置固定的宽度，设置左浮动。

2.兄弟元素使用overflow: hidden 开启BFC，重新计算元素的宽度（切记不要设置固定值，不能设置100%，使用浏览器默认值）

## 三列布局（圣杯布局、双飞翼）

* 综合负右外边距，BFC方式，实现中间自适应，两侧固定布局。

## table-cell

* 为元素设置 display: table-cell 该元素就具有了<td>的特性
  + 可以通过为该院设置verticale-align 轻松实现对齐内容及其子元素在垂直方向的位置
  + **我们推荐在局部布局中使用！不推荐整体布局！**

使用场景：制作多列等宽自适应布局

1.父元素 display: table; width: 100%（table默认没有宽度）

2.布局元素（子元素 ） display: table-cell;

3.子元素之间的空隙，通过一个正常的div分割即可。

4.如果存在多行，需要在包裹一个 display: table-row（<tr>）

<**div class="table-box"**>  
 <**div class="box box1"**></**div**>  
 <**div class="white-space"**></**div**>  
 <**div class="box box2"**></**div**>  
 <**div class="white-space"**></**div**>  
 <**div class="box box3"**></**div**>  
</**div**>

.**table-box**{  
 **display**: **table**;  
 **width**: 100%;  
}  
.**box** {  
 **height**: 200**px**;  
 **width**: 33.3333333%;  
 **display**: **table-cell**;  
}  
.**box1**{  
 **background-color**: **red**;  
}  
.**box2**{  
 **background-color**: **blue**;  
}  
.**box3**{  
 **background-color**: **gold**;  
}  
.**white-space**{  
 **height**: 200**px**;  
 **width**: 5**px**;  
 **background-color**: **#fff**;  
}

