**CSS 3**

**不同浏览器可能需要不同的前缀**



1. **CSS3能做什么**
2. **选择器**
3. **圆角效果**
4. **块阴影与文字阴影**
5. **色彩**
6. **渐变效果**
7. **个性化字体**
8. **多背景图**
9. **边框背景图**
10. **元素等的变形处理**
11. **多栏布局（省去多个div构造布局）**
12. **媒体查询（针对不同屏幕分辨率应用不同的样式显示）**
13. **……**
14. **边框**

**2-1. CSS3边框 圆角效果 border-radius**

border-radius:10px; //四个角都是半径为10px的圆角

border-radius:5px 4px 3px 2px; //分别代表 左上 右上 右下 左下 角；（顺时针）

示例：实心上半圆；实心圆

border-radius:50px 50px 0px 0px; width:100px;height:50px;

border-radius:100px/10px; 水平方向上的圆角/垂直方向上的圆角

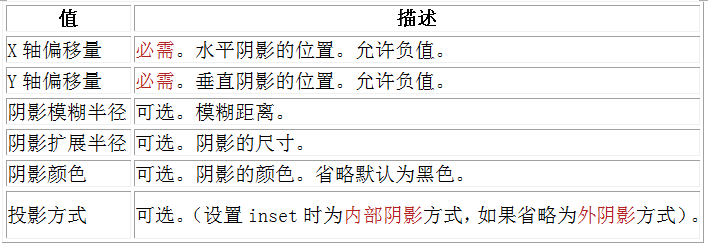
border-radius:100px 100px 10px 10px/10px 10px 100px 100px;

**2-2. CSS3边框 阴影 box-shadow（一）**

box-shadow是向盒子添加阴影。支持添加一个或者多个。（添加多个阴影用逗号隔开）

box-shadow: X轴偏移量 Y轴偏移量 [阴影模糊半径] [阴影扩展半径] [阴影颜色] [投影方式];

参数介绍：



**2-3. CSS3边框 阴影 box-shadow（二）**

1. 阴影模糊半径 与 阴影扩展半径的区别

阴影模糊半径：此参数可选，其值只能是为正值，如果其值为0时，表示阴影不具有模糊效果，其值越 大阴影的边缘就越模糊；

阴影扩展半径：此参数可选，其值可以是正负值，如果值为正，则整个阴影都延展扩大，反之值为负值时，则缩小；

1. X轴偏移量和Y轴偏移量值可以设置为负数

**2-4. CSS3边框 为边框应用图片 border-image**

border-image:url(边框图片链接) 边框宽度（四个边（不加单位）） 延伸方式

延伸方式：round（平铺） repeat（重复） stretch（拉伸）

1. **颜色相关**

**3-1. CSS3颜色 颜色之RGBA**

RGB是一种色彩标准，是由红(R)、绿(G)、蓝(B)的变化以及相互叠加来得到各式各样的颜色。RGBA是在RGB的基础上增加了控制alpha透明度的参数。

color: rgba(R,G,B,A) A为透明度参数，取值在0~1之间，不可为负值。

background-color: rgba(100,120,60,0.5);

**3-1. CSS3颜色 渐变色彩**

CSS3 Gradient 分为线性渐变(linear)和径向渐变(radial)。由于不同的渲染引擎实现渐变的语法不同，这里我们只针对线性渐变的 W3C 标准语法来分析其用法，其余大家可以查阅相关资料。W3C 语法已经得到了 IE10+、Firefox19.0+、Chrome26.0+ 和 Opera12.1+等浏览器的支持。

linear-gradient(to bottom,#fff,#999)

线性渐变 渐变方向 颜色起始和结束点

参数：第一个参数:指定渐变方向，可以用“角度”的关键词或“英文”来



第一个参数省略时，默认为“180deg”，等同于“to bottom”。

第二个和第三个参数，表示颜色的起始点和结束点，可以有多个颜色值。

background-image:linear-gradient(to left,red,orange,yellow,green,blue,indigo,violet);

1. **文字与字体**

**4-1. CSS3文字与字体 text-overflow与word-wrap**

text-overflow用来设置是否使用一个省略标记（...）标示对象内文本的溢出。

text-overflow:clip | ellipsis clip 表示剪切 ellipsis表示显示省略标记

但是text-overflow只是用来说明文字溢出时用什么方式显示，要实现溢出时产生省略号的效果，还须定义强制文本在一行内显示（white-space:nowrap）及溢出内容为隐藏（overflow:hidden），只有这样才能实现溢出文本显示省略号的效果

text-overflow:ellipsis;

overflow:hidden;

white-space:nowrap;

同时，word-wrap也可以用来设置文本行为，当前行超过指定容器的边界时是否断开转行。

word-wrap:normal | break-word normal 表示控制连续文本换行 break-word 内容将在边界内换行

normal为浏览器默认值，break-word设置在长单词或 URL地址内部进行换行，此属性不常用，用浏览器默认值即可。

**4-2. CSS3文字与字体 嵌入字体@font-face**

@font-face能够加载服务器端的字体文件，让浏览器端可以显示用户电脑里没有安装的字体。

@font-face{

font-family: 字体名称;

src: 字体文件在服务器上的相对或绝对路径;

}

这样设置之后，就可以像使用普通字体一样在（font-\*）中设置字体样式。

如：p{

font-size:12px;

font-family:”My Font”;

/\*必须项，设置@font-face中font-family同样的值\*/

}

**4-3. CSS3文字与字体 文本阴影text-shadow**

text-shadow可以用来设置文本的阴影效果

text-shadow: X-offset Y-offset blur color;

blur：是指阴影的模糊程度，其值不能是负值，如果值越大，阴影越模糊，反之阴影越清晰，如果 不需要阴影模糊可以将blur值设置为0；

color可以使用rgba色

如：text-shadow: 0 1px 1px #000;

1. **与背景相关的样式**

**5-1. CSS3背景 background-origin**

设置元素背景图片的原始起始位置

**background-origin: border-box | padding-box | content-box;**

参数分别表示背景图片是从边框，还是**内边距（默认值**），或者是内容区域开始显示。



需要注意的是，如果背景不是no-repeat，这个属性无效，它会从边框开始显示。

**5.2. CSS3背景 background-clip**

用来将背景图片做适当的裁剪以适应实际需要

background-clip: border-box | padding-box | content-box | no-clip;

参数分别表示从边框、或内填充，或者内容区域**向外裁剪背景**。no-clip表示不裁切，和参数border-box显示同样的效果。background-clip默认值为border-box。



**5.3. CSS3背景 background-size**

设置背景图片的大小，以长度值或百分比显示，还可以通过cover和contain来对图片进行伸缩。

**background-size: auto | <长度值> | <百分比> | cover | contain;**

1、auto：默认值，不改变背景图片的原始高度和宽度；

2、<长度值>：成对出现如200px 50px，将背景图片宽高依次设置为前面两个值，当设置一个值时，将其作为 图片宽度值来等比缩放；

3、<百分比>：0％~100％之间的任何值，将背景图片宽高依次设置为所在元素宽高乘以前面百分比得出的数 值，当设置一个值时同上；

4、cover：顾名思义为覆盖，即将背景图片等比缩放以填满整个容器；

5、contain ：容纳，即将背景图片等比缩放至某一边紧贴容器边缘为止。

**5.4. CSS3背景 multiple backgrounds**

多重背景，也就是CSS2里background的属性外加origin、clip和size组成的新background的多次叠加，缩写时为用逗号隔开的每组值；用分解写法时，如果有多个背景图片，而其他属性只有一个（例如background-repeat只有一个），表明所有背景图片应用该属性值。

background: [background-color] | [background-image] | [background-position][**/**background-size] | [background-repeat] | [background-attachment] | [background-clip] | [background-origin],...

可以把上面的缩写拆解成一下形式：

background-image: url1,url2,...,urlN;

background-repeat: repeat1,repeat2,...,repeatN;

background-position: position1,position2,...,positionN;

background-size: size1,size2,...,sizeN;

background-attachment: attachment1,attachment2,...,attachmentN;

background-clip: clip1,clip2,...,clipN;

background-origin: origin1,origin2,...,originN;

background-color: color;

注意：

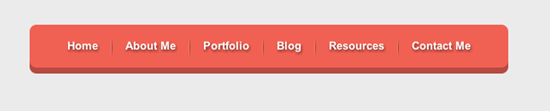
用逗号隔开每组 background 的缩写值；

如果有 size 值，需要紧跟 position 并且用 "/" 隔开；

如果有多个背景图片，而其他属性只有一个（例如 background-repeat 只有一个），表明所有背景图片应用该属性值。

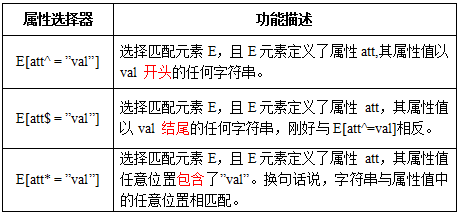
background-color 只能设置一个。

**5.5. CSS3背景 制作导航菜单综合练习题**



1. **CSS3选择器（上）**

**6-1. CSS3选择器 属性选择器**



a[class^=icon]{background:red;} a[href$=pdf]{background:red;} a[title\*=more]{ }

**6-2. CSS3结构性伪类选择器——root**

:root选择器，从字面上我们就可以很清楚的理解是根选择器，他的意思就是匹配元素E所在文档的根元素。在HTML文档中，根元素始终是<html>。

:root{background:red;} 等同于 html{background:red;}

**6-3. CSS3结构性伪类选择器——not**

:not选择器称为否定选择器，和jQuery中的:not选择器一模一样，可以选择除某个元素之外的所有元素。

就拿form元素来说，比如说你想给表单中除submit按钮之外的input元素添加红色边框，CSS代码可以写成：

input:not([type="submit"]){

border:1px solid red;

}

**6-4. CSS3结构性伪类选择器——empty**

:empty选择器表示的就是空。用来选择没有任何内容的元素，这里没有内容指的是一点内容都没有，哪怕是一个空格。

**6-5. CSS3结构性伪类选择器——target**

:target选择器称为目标选择器，用来匹配文档(页面)的url的某个标志符的目标元素。

<h2><a href="#brand">Brand</a></h2>

<div class="menuSection" id="brand">

content for Brand

</div>

css部分:

.menuSection{

display: none;

}

#brand:target{/\*这里的:target就是指id="brand"的div对象\*/

display:block;

}

**6-6. CSS3结构性伪类选择器——first-child**

:first-child选择器表示的是选择父元素的第一个子元素的元素E。简单点理解就是选择元素中的第一个子元素，记住是子元素，而不是后代元素。

ul li:first-child{

color: red;

}

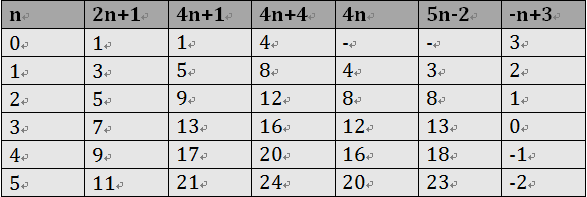
**6-7. CSS3结构性伪类选择器——last-child**

:last-child择器与“:first-child”选择器作用类似，不同的是“:last-child”选择器选择的是元素的最后一个子元素。

**6-8. CSS3结构性伪类选择器——nth-child(n)**

:nth-child(n)选择器用来定位某个父元素的一个或多个特定的子元素。其中“n”是其参数，而且可以是**整数值(1,2,3,4)，也可以是表达式(2n+1、-n+5)和关键词(odd、even)**，但参数n的起始值始终是1，而不是0。也就是说，参数n的值为0时，选择器将选择不到任何匹配的元素。

经验与技巧:当“:nth-child(n)”选择器中的n为一个表达式时，其中n是从0开始计算，当表达式的值为0或小于0的时候，不选择任何匹配的元素。如下表所示：



如：ol>li:nth-child(2n){background:orange;}

ol>li:nth-child(odd){background:orange;} 奇数

ol>li:nth-child(even){background:orange;} 偶数

**6-9. CSS3结构性伪类选择器——nth-last-child(n)**

:nth-last-child(n)选择器和前面的“:nth-child(n)”选择器非常的相似，只是这里多了一个“last”，所起的作用和“:nth-child(n)”选择器有所区别，从某父元素的最后一个子元素开始计算，来选择特定的元素。

选择列表中倒数第五个列表项，将其背景设置为橙色。

ol>li:nth-last-child(5){background:orange;}

**6-10. CSS3 first-of-type选择器**

:first-of-type选择器类似于“:first-child”选择器，不同之处就是指定了元素的类型,其主要用来定位一个父元素下的某个类型的第一个子元素。

通过“:first-of-type”选择器，定位div容器中的第一个p元素（p不一定是容器中的第一个子元素），并设置其背景色为橙色。

.wrapper > p:first-of-type {

background: orange;

}

**6-11. CSS3 nth-of-type(n) 选择器**

:nth-of-type(n)选择器和“:nth-child(n)”选择器非常类似，不同的是它只计算父元素中指定的某种类型的子元素。当某个元素中的子元素不单单是同一种类型的子元素时，使用“:nth-of-type(n)”选择器来定位于父元素中某种类型的子元素是非常方便和有用的。在“:nth-of-type(n)”选择器中的“n”和“:nth-child(n)”选择器中的“n”参数也一样，可以是具体的整数，也可以是表达式，还可以是关键词。

通过“:nth-of-type(2n)”选择器，将容器“div.wrapper”中偶数段数的背景设置为橙色。

.wrapper>p:nth-of-type(2n){background:orange;}

**6-12. CSS3 last-of-type选择器**

:last-of-type选择器和“:first-of-type”选择器功能是一样的，不同的是他选择是父元素下的某个类型的最后一个子元素。

通过“:last-of-type”选择器，将容器“div.wrapper”中最后一个段落元素背景设置为橙色

.wrapper>p:last-of-type{

background:orange;

}

**6-13. CSS3 last-of-type选择器**

:nth-last-of-type(n)选择器和“:nth-of-type(n)”选择器是一样的，选择父元素中指定的某种子元素类型，但它的起始方向是从最后一个子元素开始，而且它的使用方法类似于上节中介绍的“:nth-last-child(n)”选择器一样。

通过“:nth-last-of-type(n)”选择器将容器“div.wrapper”中的倒数第三个段落背景设置为橙色。

.wrapper>p:nth-last-of-type(3){background:orange;}

**6-14. CSS3 only-child选择器**

:only-child选择器选择的是父元素中只有一个子元素，而且只有唯一的一个子元素。也就是说，匹配的元素的父元素中仅有一个子元素，而且是一个唯一的子元素。

选择器选择的是父元素中只有一个子元素，而且只有唯一的一个子元素。也就是说，匹配的元素的父元素中仅有一个子元素，而且是一个唯一的子元素。

<div class="post">

<p>我是一个段落</p>

<p>我是一个段落</p>

</div>

<div class="post">

<p>我是一个段落</p>

</div>

.post p {

background: green;

color: #fff;

padding: 10px;

}

**.post p:only-child** {

background: orange;

}

**6-15. CSS3 only-of-type选择器**

:only-of-type选择器用来选择一个元素是它的父元素的唯一一个相同类型的子元素。这样说或许不太好理解，换一种说法。“:only-of-type”是表示一个元素他有很多个子元素，而其中只有一种类型的子元素是唯一的，使用“:only-of-type”选择器就可以选中这个元素中的唯一一个类型子元素。

<div class="wrapper">

<p>我是一个段落</p>

<p>我是一个段落</p>

<p>我是一个段落</p>

<div>我是一个Div元素</div>

</div>

<div class="wrapper">

<div>我是一个Div</div>

<ul>

<li>我是一个列表项</li>

</ul>

<p>我是一个段落</p>

</div>

.wrapper > div:only-of-type {

background: orange;

}

1. **CSS3选择器（下）**

**7-1. CSS3选择器 :enabled**

在Web的表单中，有些表单元素有可用（“:enabled”）和不可用（“:disabled”）状态，比如输入框，密码框，复选框等。在默认情况之下，这些表单元素都处在可用状态。那么我们可以通过伪选择器“:enabled”对这些表单元素设置样式。

通过“:enabled”选择器，修改文本输入框的边框为2像素的红色边框，并设置它的背景为灰色。

input[type="text"]:enabled {

background: #ccc;

border: 2px solid red;

}

**7-2. CSS3选择器 :disabled**

:disabled选择器刚好与“:enabled”选择器相反，用来选择不可用表单元素。要正常使用“:disabled”选择器，需要在表单元素的HTML中设置“disabled”属性。

**input[type="text"]:disabled {**

background: rgba(0,0,0,.15);

border: 1px solid rgba(0,0,0,.15);

color: rgba(0,0,0,.15);

}

**7-3. CSS3选择器 :checked选择器**

在表单元素中，单选按钮和复选按钮都具有选中和未选中状态。（大家都知道，要覆写这两个按钮默认样式比较困难）。在CSS3中，我们可以通过状态选择器“:checked”配合其他标签实现自定义样式。而“:checked”表示的是选中状态。

通过“:checked”状态来自定义复选框效果。

input[type="checkbox"]:checked + span {

opacity: 1;

}

**7-4. CSS3选择器 ::selection选择器**

::selection伪元素是用来匹配突出显示的文本(用鼠标选择文本时的文本)。浏览器默认情况下，用鼠标选择网页文本是以“深蓝的背景，白色的字体”显示的，效果如下图所示：



有的时候设计要求,不使用上图那种浏览器默认的突出文本效果，需要一个与众不同的效果，此时“::selection”伪元素就非常的实用。不过在Firefox浏览器还需要添加前缀。

::-moz-selection {

background: red;

color: green;

}

::selection {

background: red;

color: green;

}

1、IE9+、Opera、Google Chrome 以及 Safari 中支持 ::selection 选择器。

2、Firefox 支持替代的 ::-moz-selection。

**7-5. CSS3选择器 :read-only选择器**

:read-only伪类选择器用来指定处于只读状态元素的样式。简单点理解就是，元素中设置了“readonly=’readonly’”

input[type=”text”]:read-only{background:orange;}

input[type=”text”]:-moz-read-only{background:orange;}

**7-6. CSS3选择器 :read-write选择器**

:read-write选择器刚好与“:read-only”选择器相反，主要用来指定当元素处于非只读状态时的样式。

**7-7. CSS3选择器 ::before和::after**

::before和::after这两个主要用来给元素的前面或后面插入内容，**这两个常和"content"**配合使用，使用的场景最多的就是清除浮动。

.clearfix::before,

.clearfix::after {

content: ".";

display: block;

height: 0;

visibility: hidden;

}

.clearfix:after {clear: both;}

.clearfix {zoom: 1;}

**7-8. 切换背景图像综合练习题**

使用CSS3知识，实现背景图片的切换效果（不依赖js，点击小图切换大的背景图）

1. **CSS3中的变形与动画（上）**

**8-1. CSS3变形——旋转rotate()**

旋转rotate()函数通过指定的角度参数使元素相对原点进行旋转。它主要在二维空间内进行操作，设置一个角度值，用来指定旋转的幅度。如果这个值为正值，元素相对原点中心顺时针旋转；如果这个值为负值，元素相对原点中心逆时针旋转。

-webkit-transform:rotate(20deg); (-20deg)

-moz-transform:rotate(20deg);

transform:rotate(20deg);

**8-2. CSS3变形——扭曲skew()**

扭曲skew()函数能够让元素倾斜显示。它可以将一个对象以其中心位置围绕着X轴和Y轴按照一定的角度倾斜。这与rotate()函数的旋转不同，rotate()函数只是旋转，而不会改变元素的形状。skew()函数不会旋转，而只会改变元素的形状。

Skew()具有三种情况：

1. skew(x,y)使元素在水平和垂直方向同时扭曲（X轴和Y轴同时按一定的角度值进行扭曲变形）

第一个参数对应X轴，第二个参数对应Y轴。如果第二个参数未提供，则值为0，也就是Y轴方向上无斜切。

1. skewX(x)仅使元素在水平方向扭曲变形（X轴扭曲变形）
2. skewY(y)仅使元素在垂直方向扭曲变形（Y轴扭曲变形）

-webkit-transform:skew(45deg,45deg); x,y轴方向斜切

-moz-transform:skew(45deg); x轴方向斜切

transform:skewX(45deg); x轴方向斜切

transform:skewY(45deg); y轴方向斜切

**8-3. CSS3变形——缩放scale()**

缩放scale函数让元素根据中心原点对对象进行缩放。

缩放 scale 具有三种情况：

1. scale(X,Y)使元素水平方向和垂直方向同时缩放（也就是X轴和Y轴同时缩放）

transform:scale(1.5,0.5) Y是可选参数，若不设置，则x，y**缩放倍数**相同。

2、scaleX(x)元素仅水平方向缩放（X轴缩放）

3、scaleY(y)元素仅垂直方向缩放（Y轴缩放）

**div:hover{**

**transform:scale(1.5);**

**}**

注意： scale()的取值默认的值为1，当值设置为0.01到0.99之间的任何值，作用使一个元素缩小；而任何大于或等于1.01的值，作用是让元素放大。

transform:scale(0.5);

**8-4. CSS3变形——位移translate()**

translate()函数可以将元素向指定的方向移动，类似于position中的relative。或以简单的理解为，使用translate()函数，可以把元素从原来的位置移动，而不影响在X、Y轴上的任何Web组件。

translate我们分为三种情况：

1. translate(x,y)水平方向和垂直方向同时移动（也就是X轴和Y轴同时移动）
2. translateX(x)仅水平方向移动（X轴移动）
3. translateY(Y)仅垂直方向移动（Y轴移动）

transform:translate(-50%,-50%);

**8-5. CSS3变形——矩阵matrix**

matrix()是一个含六个值的(a,b,c,d,e,f)变换矩阵，用来指定一个2D变换，相当于直接应用一个[a b c d e f]变换矩阵。就是基于水平方向（X轴）和垂直方向（Y轴）重新定位元素,此属性值使用涉及到数学中的矩阵，我在这里只是简单的说一下CSS3中的transform有这么一个属性值，如果需要深入了解，需要对数学矩阵有一定的知识。

div{

-webkit-transform:matrix(1,0,0,1,50,50);

-moz-transform:matrix(1,0,0,1,50,50);

transform:matrix(1,0,0,1,50,50);

}

transform:matrix(a,b,c,d,e,f);

a: X轴水平缩放 scale(x,y)

b: Y轴垂直斜切 skew(x,y)

c: X轴水平斜切 skew(x,y)

d: Y轴垂直缩放 rotate(x,y)

e: X轴水平位移 translate(x,y)

f: Y轴垂直位移 translate(x,y)

transform: matrix(1.2,0.2,-0.2,1.3,100,100);

rotate(x,y)旋转使用a,b,c,d四个值控制

**8-6. CSS3变形——原点transform-origin**

任何一个元素都有一个中心点，默认情况之下，其中心点是居于元素X轴和Y轴的50%处。

transform-origin:left top; 原点到左上角

**8-7. CSS3动画——过渡属性transition-property**

鼠标的单击、获得焦点，被点击或对元素任何改变中触发，并平滑地以动画效果改变CSS的属性值。

在CSS中创建简单的过渡效果可以从以下几个步骤来实现：

第一，在默认样式中声明元素的初始状态样式；

第二，声明过渡元素最终状态样式，比如悬浮状态；

第三，在默认样式中通过添加过渡函数，添加一些不同的样式。

CSS3的过度transition属性是一个复合属性，主要包括以下几个子属性：

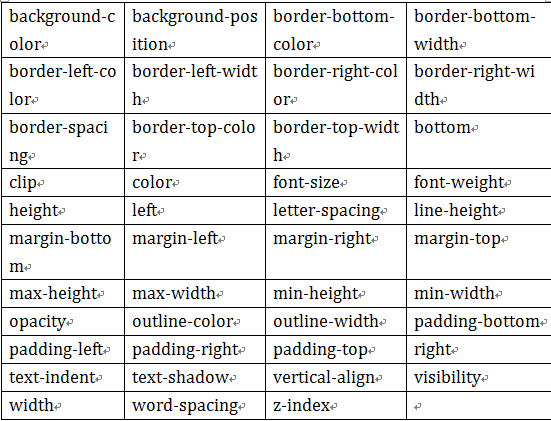
transition-property:指定过渡或动态模拟的CSS属性

transition-duration:指定完成过渡所需的时间

transition-timing-function:指定过渡函数

transition-delay:指定开始出现的延迟时间

transition-property用来指定过渡动画的CSS属性名称，而这个过渡属性只有具备一个中点值的属性（需要产生动画的属性）才能具备过渡效果，其对应具有过渡的CSS属性主要有：



div{

width:200px;height:200px;

**background-color:red;**

margin:20px auto;

-webkit-transition: background-color .5s ease .1s;

transition: background-color .5s ease .1s;

}

div:hover{background-color:orange;}

特别注意：当“transition-property”属性设置为all时，表示的是所有中点值的属性。

假设你的初始状态设置了样式“width”,“height”,“background”,当你在终始状态都改变了这三个属性，那么all代表的就是“width”、“height”和“background”。如果你的终始状态只改变了“width”和“height”时，那么all代表的就是“width”和“height”。

transition-property:width;

**8-8. CSS3动画——过渡所需时间transition-duration**

transition-duration属性主要用来设置一个属性过渡到另一个属性所需的时间，也就是从旧属性过渡到新属性花费的时间长度，俗称持续时间。

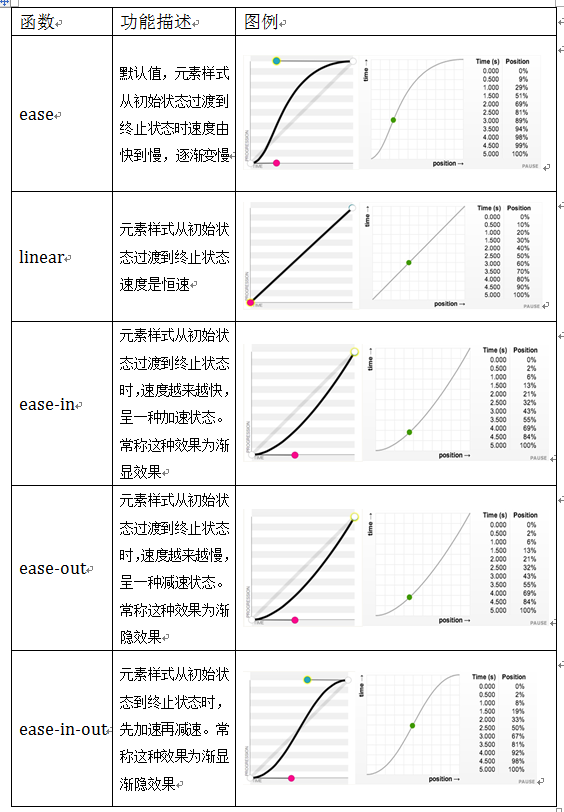
在鼠标悬停（hover）状态下，让容器从直角慢慢过渡到圆角，并让整个动画持续0.5s。

div{transition:border-radius .5s ease-out .2s}

transition-duration: .5s;

**8-9. CSS3动画——过渡函数transition-timing-function**

transition-timing-function属性指的是过渡的“缓动函数”。主要用来指定浏览器的过渡速度，以及过渡期间的操作进展情况，其中要包括以下几种函数：



transition-timing-function:ease-in-out;

transition-timing-function:linear;

**8-9. CSS3动画——过渡函数transition-delay**

transition-delay属性和transition-duration属性极其类似，不同的是transition-duration是用来设置过渡动画的持续时间，而transition-delay主要用来指定一个动画开始执行的时间，也就是说当改变元素属性值后多长时间开始执行。

有时我们想改变两个或者多个css属性的transition效果时，只要把几个transition的声明串在一起，用逗号（“，”）隔开，然后各自可以有各自不同的延续时间和其时间的速率变换方式。但需要值得注意的一点：第一个时间的值为 transition-duration，第二个为transition-delay。

如：a{

transition: background 0.8s ease-in 0.3 ,color 0.6s ease-out 0.3;

}

1. **CSS3中的变形与动画（下）**

**9-1. CSS3 keyframes介绍 （关键帧）**

keyframes 被称为关键帧，其类似于Flash中的关键帧。在CSS3中其主要以“@keyframes”开头，后面紧跟着是动画名称加上一对花括号“{…}”，括号中就是一些不同时间段样式规则。

@keyframes changecolor{

0%{ background:red; } //from

...

100%{ background:green; } //to

}

在一个“@keyframes”中的样式规则可以由多个百分比构成的，如在“0%”到“100%”之间创建更多个百分比，分别给每个百分比中给需要有动画效果的元素加上不同的样式，从而达到一种在不断变化的效果。

**经验与技巧：**在@keyframes中定义动画名称时，其中0%和100%还可以使用关键词from和to来代表，其中0%对应的是from，100%对应的是to。

**9-2. CSS3中调用动画**

animation-name属性主要是用来调用 @keyframes 定义好的动画。需要特别注意: animation-name 调用的动画名需要和“@keyframes”定义的动画名称完全一致（区分大小写），如果不一致将不具有任何动画效果。

animation-name:none | IDENT[,none|DENT]\*;

1、IDENT是由 @keyframes 创建的动画名，上面已经讲过了（animation-name 调用的动画名需要和“@keyframes”定义的动画名称完全一致）；

2、none为默认值，当值为 none 时，将没有任何动画效果,这可以用于覆盖任何动画。

注意：需要在 Chrome 和 Safari 上面的基础上加上-webkit-前缀，Firefox加上-moz-。

-webkit-animation-name:around;

-moz-animation-name:around;

animation-name:around;

**9-3. CSS3中设置动画播放时间**

animation-duration主要用来设置CSS3动画播放时间，其使用方法和transition-duration类似，是用来指定元素播放动画所持续的时间长，也就是完成从0%到100%一次动画所需时间。单位：S秒

animation-duration: <time>[,<time>]\*

取值<time>为数值，单位为秒，其默认值为“0”，这意味着动画周期为“0”，也就是没有动画效果（如果值为负值会被视为“0”）。

-webkit-animation-duration:5s;

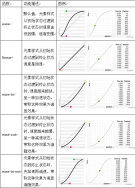
-moz-animation-duration:5s;

animation-duration:5s;

**9-4. CSS3中设置动画播放方式**

animation-timing-function属性主要用来设置动画播放方式。主要让元素根据时间的推进来改变属性值的变换速率，简单点说就是动画的播放方式。

它和transition中的transition-timing-function一样，具有以下几种变换方式：ease,ease-in,ease-in-out,ease-out,linear和cubic-bezier。对应功如下：



transition-timing-function: ease-in-out;

**9-5. CSS3中设置动画开始播放的时间**

animation-delay属性用来定义动画开始播放的时间，用来触发动画播放的时间点。和transition-delay属性一样，用于定义在浏览器开始执行动画之前等待的时间。

animation-delay:<time>[,<time>]\*

animation-delay:3s;

**9-6. CSS3中设置动画播放次数**

animation-iteration-count属性主要用来定义动画的播放次数。

animation-iteration-count: infinite | <number> [,infinite | <number>]\*

1、其值通常为整数，但也可以使用带有小数的数字，其默认值为1，这意味着动画将从开始到结束只播放一次。

2、如果取值为infinite，动画将会无限次的播放。

animation-iteration-count: infinite;

animation-iteration-count: 5;

**9-7. CSS3中设置动画播放方向**

animation-direction属性主要用来设置动画播放方向

animation-direction:normal | [,normal | alternate]\*

1、normal是默认值，如果设置为normal时，动画的每次循环都是向前播放；

2、另一个值是alternate，他的作用是，动画播放在第偶数次向前播放，第奇数次向反方向播放。

animation-direction:alternate; animation-direction:normal;

**9-8. CSS3中设置动画的播放状态**

animation-play-state属性主要用来控制元素动画的播放状态。

animation-play-state:running | paused;

其中running是其默认值，主要作用就是类似于音乐播放器一样，可以通过paused将正在播放的动画停下来，也可以通过running将暂停的动画重新播放，这里的重新播放不一定是从元素动画的开始播放，而是从暂停的那个位置开始播放。另外如果暂停了动画的播放，元素的样式将回到最原始设置状态。

**9-9. CSS3中设置动画时间外属性 （开始前和结束后）**

animation-fill-mode属性定义在动画开始之前和结束之后发生的操作。主要具有四个属性值：**none、forwards、backwords和both。**其四个属性值对应效果如下：

1. none 默认值，表示表示动画将按预期进行和结束，在动画完成其最后一帧时，动画会反转到初始帧处

2. forwards 表示动画在结束后继续应用最后的关键帧的位置

3. backwards 会在向元素应用动画样式时迅速应用动画的初始帧

4. both 元素动画同时具有forwards和backwards效果

在默认情况之下，动画不会影响它的关键帧之外的属性，使用animation-fill-mode属性可以修改动画的默认行为。简单的说就是告诉动画在第一关键帧上等待动画开始，或者在动画结束时停在最后一个关键帧上而不回到动画的第一帧上。或者同时具有这两个效果。

animation-fill-mode:none;

animation-fill-mode:forwards;

animation-fill-mode:backwards;

animation-fill-mode:both;

**9-10. 制作3D旋转导航综合练习题**

sdasd

1. **布局样式相关**

**10-1. CSS3多列布局——Columns**

为了能在Web页面中方便实现类似报纸、杂志那种多列排版的布局，W3C特意给CSS3增加了一个多列布局模块（CSS Multi Column Layout Module）。它主要应用在文本的多列布局方面，这种布局在报纸和杂志上都使用了几十年了，但要在Web页面上实现这样的效果还是有相当大的难度，庆幸的是，CSS3的多列布局可以轻松实现。

columns:<column-width> || <column-count> //列宽和列数

<column-width> 主要用来定义多列中每列的宽度

<column-count> 主要用来定义多列中的列数

columns:200px 2; // 2栏显示，每栏宽度为200px;

**10-2. CSS3多列布局——column-width**

column-width的使用和CSS中的width属性一样，不过不同的是，column-width属性在定义元素列宽的时候，既可以单独使用，也可以和多列属性中其他属性配合使用。

column-width: auto | <length>; column-width:200px;

1. auto 如果column-width设置值为auto或者没有显式的设置值时，元素多列的列宽将由其他属性 来决定，比如前面的示例就是由列数column-count来决定。
2. <length> 使用固定值来设置元素列的宽度，其主要是由数值和长度单位组成，不过其值只能是正 值，不能为负值。

**10-3. CSS3多列布局——column-count**

column-count属性主要用来给元素指定想要的列数和允许的最大列数

column-count: auto | <integer>

1. auto 此值为column-count的默认值，表示元素只有一列，其主要依靠浏览器计算自动设置
2. <integer> 此值为正整数值，主要用来定义元素的列数，取值为大于0的整数，负值无效

column-count:4;

**10-4. CSS3列间距column-gap (列与列的间距)**

column-gap主要用来设置列与列之间的间距

column-gap: normal || <length>

1. normal 默认值，默值为1em（如果你的字号是px，其默认值为你的font-size值）。
2. <length> 此值用来设置列与列之间的距离，其可以使用px,em单位的任何整数值，但不能是负值

column-gap:2em;

**10-5. CSS3列表边框column-rule**

column-rule:主要是用来定义列与列之间的边框宽度、边框样式和边框颜色。简单点说，就有点类似于常用的border属性。但column-rule是不占用任何空间位置的，在列与列之间改变其宽度不会改变任何列的位置。

column-rule:<column-rule-width> | <column-rule-style> | <column-rule-color>

1. column-rule-width 类似于border-width属性，主要用来定义列边框的宽度，其默认值为“medium”， column-rule-width属性接受任意浮点数，但不接收负值。但也像 border-width属性一样，可以使用关键词：medium、thick和thin。
2. column-rule-style 类似于border-style属性，主要用来定义列边框样式，其默认值为“none”。 column-rule-style属性值与border-style属值相同，包括none、hidden、 dotted、dashed、solid、double、groove、ridge、inset、outset。
3. column-rule-color 类似于border-color属性，主要用来定义列边框颜色，其默认值为前景色 color的值，使用时相当于border-color。column-rule-color接受所有的颜 色。如果不希望显示颜色，也可以将其设置为transparent(透明色)

column-rule: 2px dotted green;

**10-6. CSS3跨列设置column-span**

column-span主要用来定义一个分列元素中的子元素能跨列多少。column-width、column-count等属性能让一元素分成多列，不管里面元素如何排放顺序，他们都是从左向右的放置内容，但有时我们需要基中一段内容或一个标题不进行分列，也就是横跨所有列，此时column-span就可以轻松实现，此属性的语法如下。

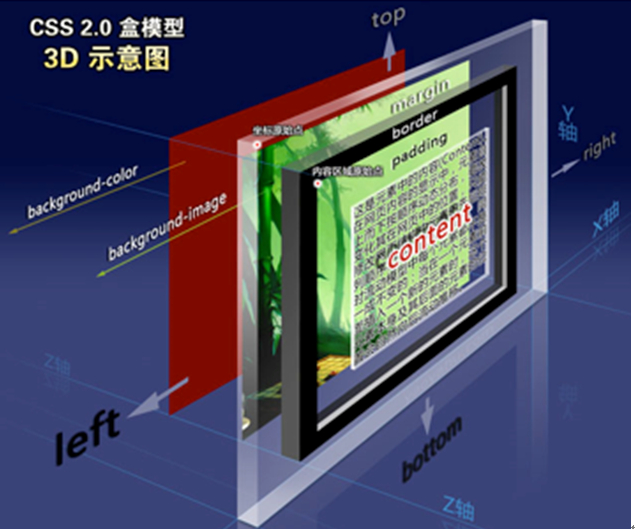
column-span: none || all;

1. none 此值为column-span的默认值，表示不跨越任何列。
2. all 这个值跟none值刚好相反，表示的是元素跨越所有列，并定位在列的Ｚ轴之上。

如：将标题横跨越所有列

column-span:all;

**10-7. CSS3盒子模型 (改变其计算方式)**



从图中可以看出padding属性和content属性层叠background-image属性，层叠background-color属性，这个是存在的，它们四者之间构成了Ｚ轴（垂直屏幕的坐标）多重层叠关系。但是border属性与margin属性、padding属性三者之间应该是平面上的并级关系，并不能构成Ｚ轴的层叠关系。

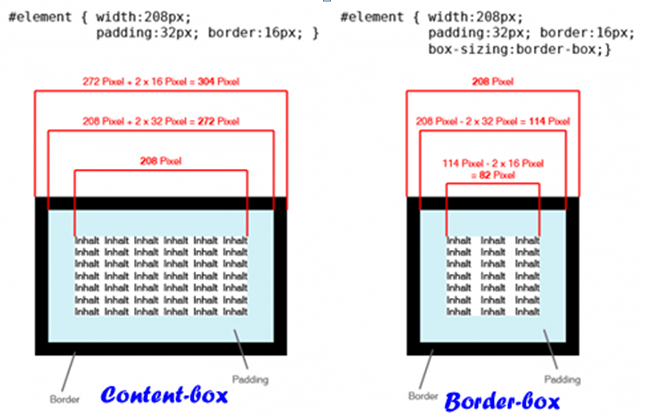
box-sizing：

在CSS3中新增加了box-sizing属性，能够事先定义盒模型的尺寸解析方式，其语法规则如下：

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

1. content-box 默认值，其让元素维持W3C的标准盒模型，也就是说元素的宽度和高度（width/height） 等于元素边框宽度（border）加上元素内距（padding）加上元素内容宽度或高度（content width/ height），也就是element width/height = border + padding + content width / height
2. border-box 重新定义CSS2.1中盒模型组成的模式，让元素维持IE传统的盒模型（IE6以下版本和IE6-7 怪异模式），也就是说元素的宽度或高度等于元素内容的宽度或高度。从上面盒模型介绍 可知，这里的内容宽度或高度包含了元素的border、padding、内容的宽度或高度（此处的 内容宽度或高度＝盒子的宽度或高度—边框—内距）。
3. inherit 使元素继承父元素的盒模型模式

其中最为关键的是box-sizing中content-box和border-box两者的区别，他们之间的区别可以通过下图来展示，其对盒模型的不同解析：



box-sizing: content-box; ||

box-sizing: border-box;

**10-8. CSS3伸缩布局（一）**

CSS3引入了一种新的布局模式——Flexbox布局，即伸缩布局盒模型（Flexible Box），用来提供一个更加有效的方式制定、调整和分布一个容器里项目布局，即使它们的大小是未知或者动态的，这里简称为Flex。

Flexbox布局常用于设计比较复杂的页面，可以轻松的实现屏幕和浏览器窗口大小发生变化时保持元素的相对位置和大小不变，同时减少了依赖于浮动布局实现元素位置的定义以及重置元素的大小。

Flexbox布局在定义伸缩项目大小时伸缩容器会预留一些可用空间，让你可以调节伸缩项目的相对大小和位置。例如，你可以确保伸缩容器中的多余空间平均分配多个伸缩项目，当然，如果你的伸缩容器没有足够大的空间放置伸缩项目时，浏览器会根据一定的比例减少伸缩项目的大小，使其不溢出伸缩容器。综合而言，Flexbox布局功能主要具有以下几点：

第一，屏幕和浏览器窗口大小发生改变也可以灵活调整布局；

第二，可以指定伸缩项目沿着主轴或侧轴按比例分配额外空间（伸缩容器额外空间），从而调整伸缩项目的大小；

第三，可以指定伸缩项目沿着主轴或侧轴将伸缩容器额外空间，分配到伸缩项目之前、之后或之间；

第四，可以指定如何将垂直于元素布局轴的额外空间分布到该元素的周围；

第五，可以控制元素在页面上的布局方向；

第六，可以按照不同于文档对象模型（DOM）所指定排序方式对屏幕上的元素重新排序。也就是说可以在浏览器渲染中不按照文档流先后顺序重排伸缩项目顺序。

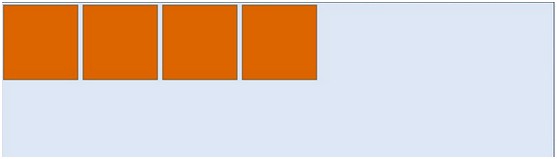
1. **创建一个flex容器**

任何一个flexbox布局的第一步是需要创建一个flex容器。为此给元素设置display属性的值为flex。在Safari浏览器中，你依然需要添加前缀-webkit，

.flexcontainer{ dispaly: -webkit-flex;display:flex; }

1. **Flex项目显示**

Flex项目是Flex容器的子元素。他们沿着主要轴和横轴定位。默认的是沿着水平轴排列一行。你可以通过flex-direction来改变主轴方向修改为column，其默认值是row。



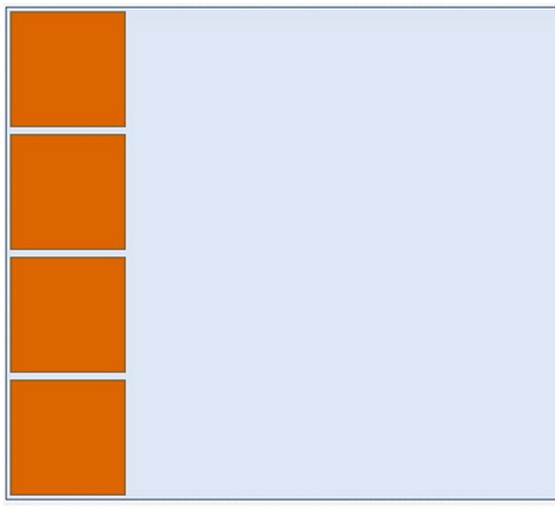
1. **Flex项目列显示**

.flexcontainer{

dispaly:-webkit-flex; display:flex;

-webkit-flex-direction: column;flex-direction:column;

}



1. **Flex项目移动到顶部**

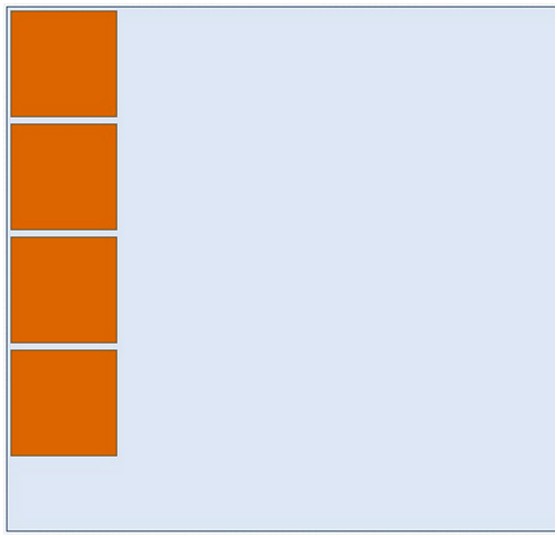
如何将flex项目移动到顶部，取决于主轴的方向。如果它是垂直的方向通过align-items设置；如果它是水平的方向通过justify-content设置。

.flexcontainer{

-webkit-flex-direction:column;flex-direction:column;

-webkit-justify-content:flex-start;justify-content:flex-start;

}



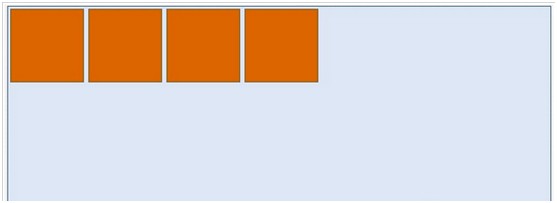
.flexcontainer{

display:-webkit-flex;display:flex;

-webkit-flex-direction:row;flex-direction:row;

-webkit-align-items:flex-start;align-items:flex-start;

}



1. **Flex项目移动到左边**

flex项目称动到左边或右边也取决于主轴的方向。如果flex-direction设置为row，设置justify-content控制方向；如果设置为column，设置align-items控制方向。

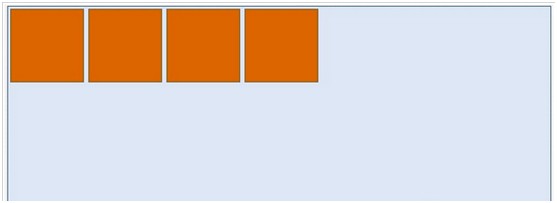
.flexcontainer{

display:-webkit-flex;display:flex;

-webkit-flex-direction:row;flex-direction:row;

-webkit-justify-content:flex-start;justify-content:flex-start;

}



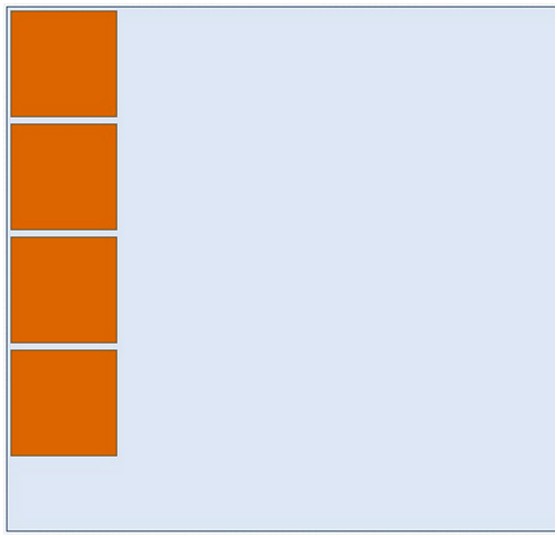
.flexcontainer{

display:-webkit-flex;display:flex;

-webkit-flex-direction:column;flex-direction:column;

-webkit-align-items:flex-start;align-items:flex-start;

}



1. **Flex项目移动到右边**

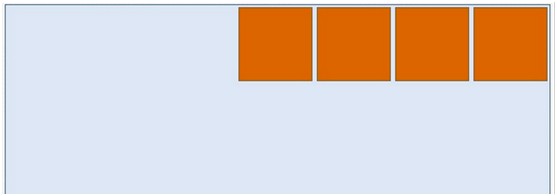
.flexcontainer{

display:-webkit-flex;display:flex;

-webkit-flex-direction:row;flex-direction:row;

-webkit-justify-content:flex-end;justify-content:flex-end;

}



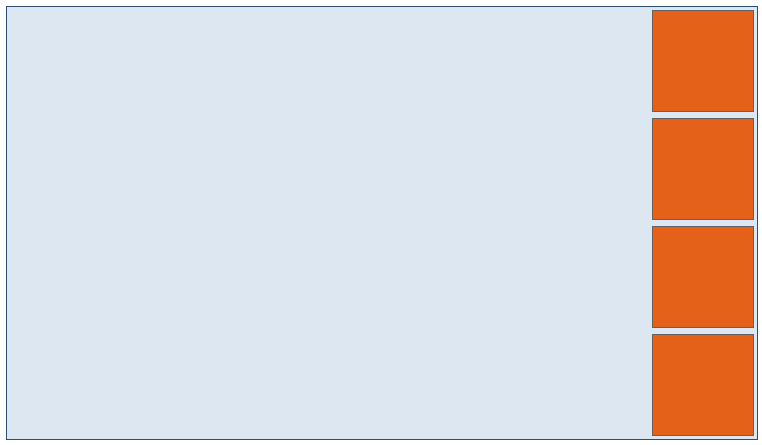
.flexcontainer{

display:-webkit-flex;display:flex;

-webkit-flex-direction:column;flex-direction:column;

-webkit-align-items:flex-end;align-items:flex-end;

}



1. **水平垂直居中**

在Flexbox容器中制作水平垂直居中是微不足道的。设置justify-content或者align-items为center。另外根据主轴的方向设置flex-direction为row或column。

.flexcontainer{

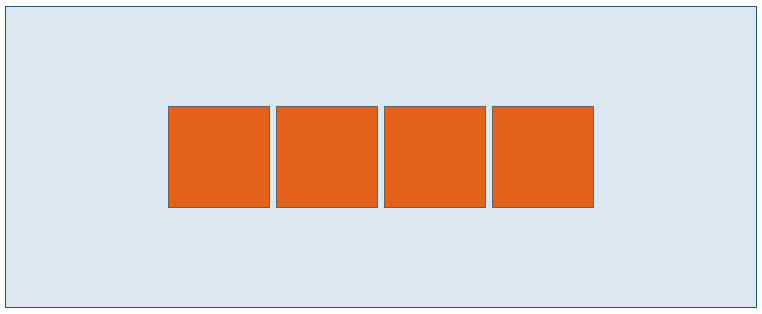
display:-webkit-flex;display:flex;

-webkit-flex-direction:row;flex-direction:row;

-webkit-align-items:center;align-items:center;

-webkit-justify-content:center;justify-content:center;

}



.flexcontainer{

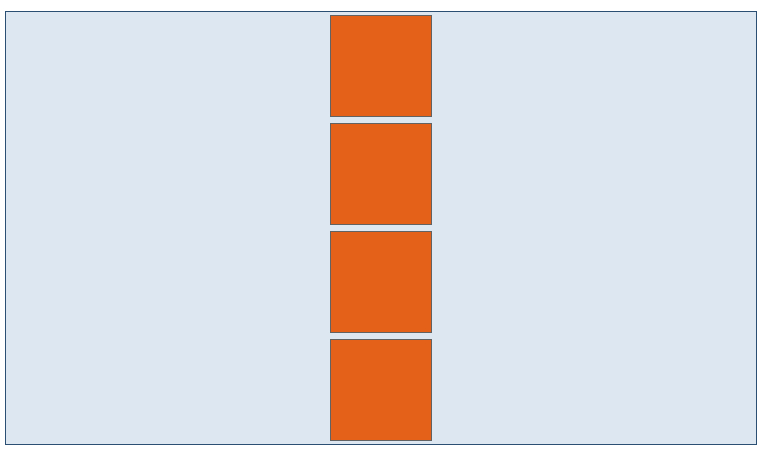
display:-webkit-flex;display:flex;

-webkit-flex-direction:column;flex-direction:column;

-webkit-align-items:center;align-items:center;

-webkit-justify-content:center;justify-content:center;

}



1. **Flex项目实现自动伸缩**

您可以定义一个flex项目，如何相对于flex容器实现自动的伸缩。需要给每个flex项目设置flex属性设置需要伸缩的值。

.bigitem{

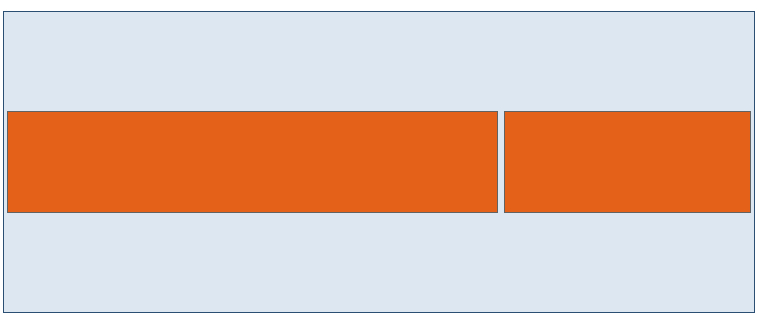
-webkit-flex:200;flex:200;

}

.smallitem{

-webkit-flex:100,flex:100;

}



1. **Media Queries与Responsive设计**
2. **用户界面与其他重要属性**