

**CSS3**

目录

[目录 2](#_Toc444804676)

[第1章 CSS3简介 4](#_Toc444804677)

[1.1 CSS3的现状 4](#_Toc444804678)

[1.2 如何对待 4](#_Toc444804679)

[第2章 准备工作 4](#_Toc444804680)

[2.1 统一环境 4](#_Toc444804681)

[2.1 如何使用手册 5](#_Toc444804682)

[第3章 基础知识 5](#_Toc444804683)

[3.1 选择器 5](#_Toc444804684)

[3.1.1 属性选择器 5](#_Toc444804685)

[3.1.2 伪类选择器 6](#_Toc444804686)

[3.1.3 伪元素选择器 7](#_Toc444804687)

[3.2 颜色 7](#_Toc444804688)

[3.3 文本 8](#_Toc444804689)

[3.4 边框 8](#_Toc444804690)

[3.4.1 边框圆角 8](#_Toc444804691)

[3.4.2 边框阴影 9](#_Toc444804692)

[3.4.3 边框图片 9](#_Toc444804693)

[3.5 盒模型 12](#_Toc444804694)

[3.6 背景 12](#_Toc444804695)

[3.7 渐变 13](#_Toc444804696)

[3.7.1 线性渐变 13](#_Toc444804697)

[3.7.2 径向渐变 14](#_Toc444804698)

[3.8 过渡 15](#_Toc444804699)

[3.9 2D转换 16](#_Toc444804700)

[3.10 3D转换 17](#_Toc444804701)

[3.11 动画 19](#_Toc444804702)

[3.12 伸缩布局 20](#_Toc444804703)

[3.13 多列布局 22](#_Toc444804704)

[第4章 Web字体 22](#_Toc444804705)

[4.1 字体格式 22](#_Toc444804706)

[4.2 字体图标 23](#_Toc444804707)

[第5章 兼容性 24](#_Toc444804708)

[第6章 高级应用 24](#_Toc444804709)

# CSS3简介

如同人类的的进化一样，CSS3是CSS2的“进化”版本，在CSS2基础上，增强或新增了许多特性， 弥补了CSS2的众多不足之处，使得Web开发变得更为高效和便捷。

## CSS3的现状

1、浏览器支持程度差，需要添加私有前缀

2、移动端支持优于PC端

3、不断改进中

4、应用相对广泛

## 如何对待

1、坚持渐进增强原则

2、考虑用户群体

3、遵照产品的方案

4、听Boss的

# 准备工作

## 统一环境

由于CSS3兼容性问题的普遍存在，为了避免因兼容性带来的干扰，我们约定统一的环境，以保证学习的效率，在最后会单独说明兼容性的问题。

1、Chrome浏览器 version 46+

2、Firefox浏览器 firefox 42+

3、PhotoShop CS6（建议）

## 如何使用手册

学会使用工具，可以让我们事半功倍。

|  |  |  |
| --- | --- | --- |
| 元字符 | 含义 | 示例 |
| [] | 全部可选项 | padding: [<length> | <percentage>]{1, 4} |
| || | 并列 | border: <line-width> || <line-style> || <color> |
| | | 多选一 | position: static | relative | absolute | fixed |
| ? | 0个或1个 | box-shadow: none | <shadow>[, <shadow>]\*  <shadow>: inset? && <length>{2, 4} && <color>? |
| \* | 0个或多个 |
| {} | 范围 |

学会查看手册，培养自主学习能力。

# 基础知识

## 选择器

CSS3新增了许多灵活查找元素的方法，极大的提高了查找元素的效率和精准度。CSS3选择器与jQuery中所提供的绝大部分选择器兼容。

### 属性

其特点是通过属性来选择元素，具体有以下5种形式：

|  |  |  |
| --- | --- | --- |
| 选择器 | 示例 | 含义 |
| E[attr] |  | 存在attr属性即可 |
| E[attr=val] |  | 属性值完全等于val |
| E[attr\*=val] |  | 属性值里包含val字符并且在“任意”位置 |
| E[attr^=val] |  | 属性值里包含val字符并且在“开始”位置 |
| E[attr$=val] |  | 属性值里包含val字符并且在“结束”位置 |

见代码示例01 选择器-属性.html

### 伪类

除了以前学过的:link、:active、:visited、:hover，CSS3又新增了其它的伪类选择器。

1、结构(位置)伪类

以某元素（E）相对于其父元素或兄弟元素的位置来获取无素；

|  |  |  |
| --- | --- | --- |
| 选择器 | 示例 | 含义 |
| E:first-child |  | 其父元素的第1个子元素 |
| E:last-child |  | 其父元素的最后1个子元素 |
| E:nth-child(n) |  | 其父元素的第n个子元素 |
| E:nth-last-child(n) |  | 其父元素的第n个子元素（倒着数） |

n遵循线性变化，其取值0、1、2、3、4、...

n可是多种形式：nth-child(2n+0)、nth-child(2n+1)、nth-child(-1n+3)等；

注：指E元素的父元素，并对应位置的子元素必须是E

见代码示例02 选择器-伪类.html

2、空伪类

E:empty 选中没有任何子节点的E元素；（使用不是非常广泛）

见代码示例03 选择器-伪类empty.html

3、目标伪类

E:target 结合锚点进行使用，处于当前锚点的元素会被选中；

见代码示例04 选择器-伪类target.html

4、排除伪类

E:not(selector) 除selector（任意选择器）外的元素会被选中；

见代码示例05 选择器-伪类not.html

### 伪元素

1、E::first-letter文本的第一个单词或字（如中文、日文、韩文等）

见代码示例07 选择器-伪元素first-letter.html

2、E::first-line 文本第一行；

见代码示例07 选择器-伪元素first-line.html

3、E::selection 可改变选中文本的样式；

见代码示例08 选择器-伪元素selection.html

4、E::before和E::after

在E元素内部的开始位置和结束位创建一个元素，该元素为行内元素，且必须要结合content属性使用。

见代码示例09 选择器-伪元素before&after.html

E:after、E:before 在旧版本里是伪元素，CSS3的规范里“:”用来表示伪类，“::”用来表示伪元素，但是在高版本浏览器下E:after、E:before会被自动识别为E::after、E::before，这样做的目的是用来做兼容处理。

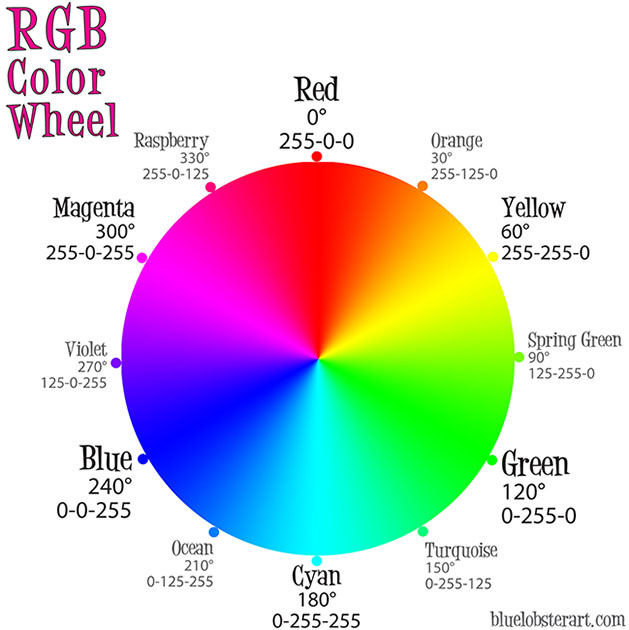
E:after、E:before后面的练习中会反复用到，目前只需要有个大致了解

":" 与 "::" 区别在于区分伪类和伪元素

## 颜色

新增了RGBA、HSLA模式，其中的A 表示透明度通道，即可以设置颜色值的透明度，相较opacity，它们不具有继承性，即不会影响子元素的透明度。

如下图所示为颜色表示方法：



Red、Green、Blue、Alpha即RGBA

Hue、Saturation、Lightness、Alpha即HSLA

不同的颜色表示方法其取值也不相同，具体如下：

R、G、B 取值范围0~255

见代码示例01 颜色-透明rgba.html

H 色调 取值范围0~360，0/360表示红色、120表示绿色、240表示蓝色

S 饱和度 取值范围0%~100%

L 亮度 取值范围0%~100%

A 透明度 取值范围0~1

见代码示例02 颜色-透明hsla.html

RGBA、HSLA可应用于所有使用颜色的地方。

见代码示

关于CSS透明度：

1、opacity只能针对整个盒子设置透明度，子盒子及内容会继承父盒子的透明度；

2 、transparent 不可调节透明度，始终完全透明

## 文本

text-shadow，可分别设置偏移量、模糊度、颜色（可设透明度）。

如：text-shadow: 2px 2px 2px #CCC;

1、水平偏移量 正值向右 负值向左；

2、垂直偏移量 正值向下 负值向上；

3、模糊度是不能为负值；

见代码示例01 文本-阴影text-shadow.html

## 盒模型

CSS3中可以通过box-sizing 来指定盒模型，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: border-box 盒子大小为 width

2、box-sizing: content-box 盒子大小为 width + padding + border

注：上面的标注的**width指的是CSS属性里设置的width: length，content的值是会自动调整的。**

**见代码示例01 盒模型.html**

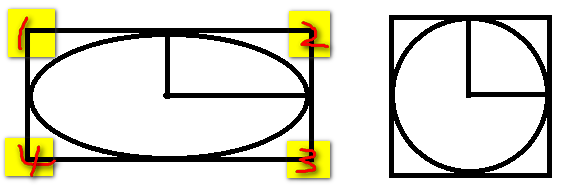
## 边框

其中边框圆角、边框阴影属性，应用十分广泛，兼容性也相对较好，具有符合渐进增强原则的特征，我们需要重点掌握。

### 边框圆角

border-radius

圆角处理时，脑中要形成圆、圆心、横轴、纵轴的概念，正圆是椭圆的一种特殊情况。如下图



为了方便表述，我们将四个角标记成1、2、3、4，如2代表右上角，CSS里提供了border-radius来设置这些角横纵轴半径值。

见代码示例01 边框-圆角border-radius.html

分别设置横纵轴半径，以“/”进行分隔，遵循“1，2，3，4”规则，“/”前面的1~4个用来设置横轴半径（分别对应横轴1、2、3、4位置 ），“/”后面1~4个参数用来设置纵轴半径（分别对应纵轴1、2、3、4位置 ）。

支持简写模式，具体如下：

1、border-radius: 10px; 表示四个角的横纵轴半径都为10px；

2、border-radius: 10px 5px; 表示1和3角横纵轴半径都为10px，2和4角横纵轴半径为5px；

3、border-radius: 10px 5px 8px; 表示1角模纵轴半径都为10px，2和4角横纵轴半径都为8px，3角的横纵轴半径都为8px；

4、border-radius: 10px 8px 6px 4px; 表示1角横纵轴半径都为10px，表示2角横纵轴半径都为8px，表示3角横纵轴半径都为6px，表示4角横纵轴半径都为6px；

见代码示例02 边框-圆角-详解border-radius.html

### 边框阴影

box-shadow

与文字阴影类似，可分别设置盒子阴影偏移量、模糊度、颜色（可设透明度）。

如box-shadow: 5px 5px 5px #CCC

1、水平偏移量 正值向右 负值向左；

2、垂直偏移量 正值向下 负值向上；

3、模糊度是不能为负值；

4、inset可以设置内阴影；

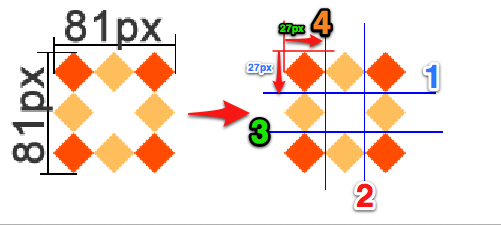
注：设置边框阴影不会改变盒子的大小，即不会影响其兄弟元素的布局。可以设置多重边框阴影，实现更好的效果，增强立体感，符合渐进增强，实际开发中可以大胆使用。

见代码示例04 边框-阴影box-shadow.html

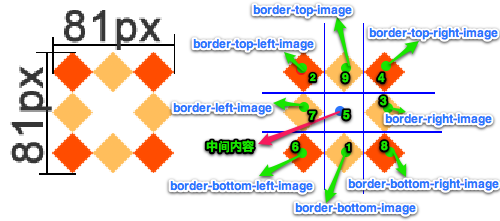
### 边框图片

border-image

设置的图片将会被“切割”成九宫格形式，然后进行设置。如下图

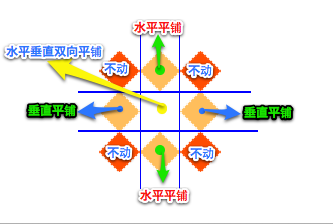


最少“4刀”便可以将一个图片切成9部分，“切割”完成后生成虚拟的9块图形，如下图



这时我们将一个盒子想象是由9部分组成的，分别是左上角、上边框、右上角、右边框、右下角、下边框、左下角、左边框、中间，那么浏览器会将切割好的9张虚拟图片分别对应到盒子的各个部分上。

其中四个角位置、形状保持不变，中心位置水平垂直两个方向平铺或拉伸，如下图



**参数详解**

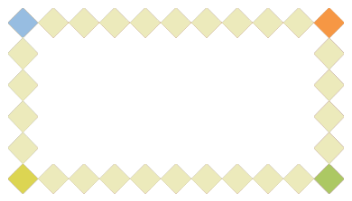
**1、border-image-source**

指定图片路径

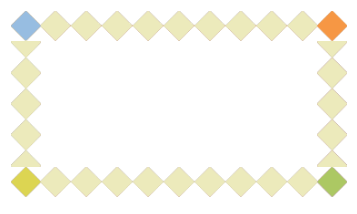
**2、border-image-repeat**

指定裁切好的虚拟图片的平铺方式

a) round会自动调整尺寸，完整显示边框图片



b) repeat 单纯平铺,多余部分，会被“裁切”而不能完整显示。



**3、border-image-slice**

4、border-image-width

设置边框背景区域的大小，这个值的大小不会影响到盒子的大小。

见代码示例06 边框-图片border-image.html

关于边框图片重点理解9宫格的裁切及平铺方式，实际开发中应用不广泛，但是如能灵活动用会给我们带来不少便利。

## 背景

背景在CSS3中也得到很大程度的增强，比如背景图片尺寸、背景裁切区域、背景定位参照点、多重背景等。

1. **background-size**

通过background-size设置背景图片的尺寸，就像我们设置img的尺寸一样，在移动Web开发中做屏幕适配应用非常广泛。

**其参数设置如下：**

a) 可以设置长度单位(px)或百分比（设置百分比时，参照盒子的宽高）

b) 设置为cover时，会自动调整缩放比例，保证图片始终填充满背景区域，如有溢出部分则会被隐藏。

c) 设置为contain会自动调整缩放比例，保证图片始终完整显示在背景区域。

见代码示例01 背景-尺寸background-size.html

**2、background-origin**

通过background-origin可以设置背景图片定位(background-position)的参照原点。

**其参数设置如下：**

border-box以边框做为参考原点；

padding-box以内边距做为参考原点；

content-box以内容区做为参考点；

见代码示例02 背景-原点background-origin.html

**3、background-clip**

通过background-clip，可以设置对背景区域进行裁切，即改变背景区域的大小。

**其参数设置如下：**

border-box裁切边框以内为背景区域；

padding-box裁切内边距以内为背景区域；

content-box裁切内容区做为背景区域；

见代码示例03 背景-裁切background-clip.html

**4、多背景**

以逗号分隔可以设置多背景，可用于自适应布局

见代码示例04 背景-多背景.html

## 渐变

渐变是CSS3当中比较丰富多彩的一个特性，通过渐变我们可以实现许多炫丽的效果，有效的减少图片的使用数量，并且具有很强的适应性和可扩展性。

### 线性渐变

linear-gradient线性渐变指沿着某条直线朝一个方向产生渐变效果，如下图是从黄色渐变到绿色。



**1、必要的元素：**

借助Photoshop总结得出线性渐变的必要元素

a、方向

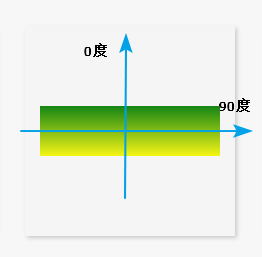
b、起始色

c、终止色

d、渐变距离

**2、关于方向**

设置渐变方向，可以用关键字如to top、to right，也可以用角度（正负值均可）如45deg、-90deg等，当以角度做为参数时，可参照下图来使用，0deg从下往上，90deg从左向右，进而可以推算出180deg从上向下。



见代码示例01 渐变-线性渐变linear-gradient.html

注：我们可以设置渐变的起始点，这个起始点的值可以是百分比形式，这个百分比在没有设置background-size时，是相对于盒子大小的，当设置了background-size时则是相对于background-size的。

### 径向渐变

radial-gradient径向渐变指从一个中心点开始沿着四周产生渐变效果



**1、必要的元素：**

a) 辐射范围即圆半径

b) 中心点 即圆的中心

c) 渐变起始色

d) 渐变终止色

e) 渐变范围

**2、关于中心点**

中心位置参照的是盒子的左上角，例如background-image: radial-gradient(120px at 0 0 yellow green)其圆心点为左上角，background-image: radial-gradient(120px at 0 100% yellow green)其圆心为左下角。

**3、关于辐射范围**

其半径可以不等，即可以是椭圆，如background-image: radial-gradient(120px 100px at 0 0 yellow green)会是一个椭圆形（横轴120px、纵轴100px）的渐变。

见代码示例02 渐变-径向渐变radial-gradient.html

**写在最后**

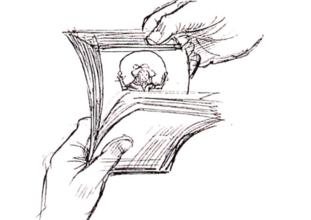
关于渐变不同浏览器有不同的版本，即语法格式不一样，我们以最新语法为准，可自行查找资料了解即可。

http://www.w3cplus.com/css3/new-css3-linear-gradient.html

## 过渡

过渡是CSS3中具有颠覆性的特征之一，可以实现元素不同状态间的平滑过渡（补间动画），经常用来制作动画效果。

帧动画：通过一帧一帧的画面按照固定顺序和速度播放。如电影胶片



见代码示例baidu.html

补间动画：自动完成从起始状态到终止状态的的过渡。

见代码示例01 体验过渡.html

关于补间动画更多学习可查看http://mux.alimama.com/posts/1009

在CSS3里使用transition可以实现补间动画（过渡效果），并且当前元素只要有“属性”发生变化时即存在两种状态(我们用A和B代指），就可以实现平滑的过渡，为了方便演示采用hover切换两种状态，但是并不仅仅局限于hover状态来实现过渡。

可以通过all设置所有属性的过渡效果，也可以分别设置某一属性的过渡效果，见代码示例03 过渡-宽高.html

可以将过渡属性transition设置在A或B状态，但是会有一些细节的差异，见代码示例03 过渡-颜色.html

transition属性拆解如下表：

|  |  |  |
| --- | --- | --- |
| 属性 | 示例 | 含义 |
| transition-property |  | 设置过渡属性 |
| transition-duration |  | 设置过渡时间 |
| transition-timing-function |  | 设置过渡速度 |
| transition-delay |  | 设置过渡延时 |

见代码示例05 过渡-属性详解.html

## 2D转换

转换是CSS3中具有颠覆性的特征之一，可以实现元素的位移、旋转、变形、缩放，甚至支持矩阵方式，配合过渡和即将学习的动画知识，可以取代大量之前只能靠Flash才可以实现的效果。

1、移动 translate(x, y) 可以改变元素的位置，x、y可为负值；

a) 移动位置相当于自身原来位置

b) y轴正方向朝下

c) 除了可以像素值，也可以是百分比，相对于自身的宽度或高度

见代码示例01 2D转换-移动translate.html

2、缩放 scale(x, y) 可以对元素进行水平和垂直方向的缩放，x、y的取值可为小数；

见代码示例02 2D转换-缩放scale.html

3、旋转 rotate(deg) 可以对元素进行旋转，正值为顺时针，负值为逆时针；

见代码示例03 2D转换-旋转rotate.html

a) 当元素旋转以后，坐标轴也跟着发生的转变

b) 调整顺序可以解决，把旋转放到最后

见代码示例05 综合使用例子.html

4、倾斜 skew(deg, deg) 可以使元素按一定的角度进行倾斜，可为负值，第二个参数不写默认为0。

见代码示例04 2D转换-倾斜skew.html

5、矩阵matrix() 把所有的2D转换组合到一起，需要6个参数（了解）。

6、transform-origin可以调整元素转换的原点

见代码示例08 转换-原点transform-origin.html

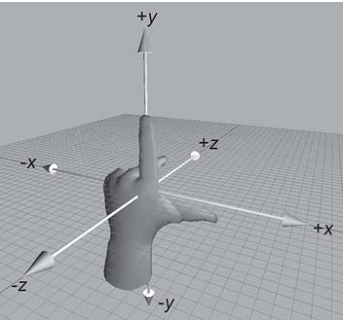
我们可以同时使用多个转换，其格式为：transform: translate() rotate() scale() ...等，其顺序会影响转换的效果。

见代码示例05 综合使用例子.html

## 3D转换

**1、左手坐标系**

伸出左手，让拇指和食指成“L”形，大拇指向右，食指向上，中指指向前方。这样我们就建立了一个左手坐标系，拇指、食指和中指分别代表X、Y、Z轴的正方向。如下图



**2、CSS中的3D坐标系**

CSS3中的3D坐标系与上述的3D坐标系是有一定区别的，相当于其绕着X轴旋转了180度，如下图



借助示例理解3D转换

a) 绕X轴旋转，见代码示例01 3D转换-旋转rotateX.html

b) 绕Y轴旋转，见代码示例02 3D转换-旋转rotateY.html

c) 绕Z轴旋转，见代码示例03 3D转换-旋转rotateZ.html

d) 在X轴移动，见代码示例04 3D转换-移动translateX.html

d) 在Y轴移动，见代码示例05 3D转换-移动translateY.html

d) 在Z轴移动，见代码示例06 3D转换-移动translateZ.html

**3、左手法则**

左手握住旋转轴，竖起拇指指向旋转轴正方向，正向就是其余手指卷曲的方向。

**4、透视（perspective）**

电脑显示屏是一个2D平面，图像之所以具有立体感（3D效果），其实只是一种视觉呈现 ，通过透视可以实现此目的。

透视可以将一个2D平面，在转换的过程当中，呈现3D效果。

**注：并非任何情况下需要透视效果，根据开发需要进行设置。**

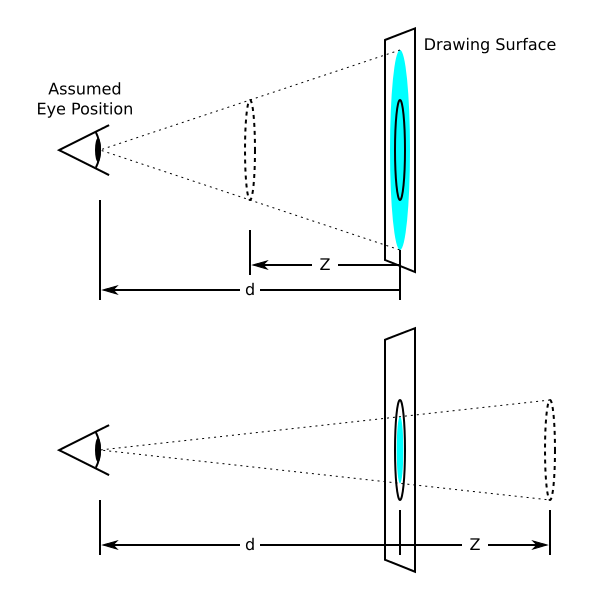
**perspective有两种写法**

a) 作为一个属性，设置给父元素，作用于所有3D转换的子元素

b) 作为transform属性的一个值，做用于元素自身

见代码示例07 3D转换-透视perspective.html

**5、理解透视距离**



透视会产生“近大远小”的效果

**6、3D呈现（transform-style）**

设置内嵌的元素在 3D 空间如何呈现，这些子元素必须为变形原素。

flat：所有子元素在 2D 平面呈现

preserve-3d：保留3D空间

见代码示例08 3D转换-透视transform-style.html

3D元素构建是指某个图形是由多个元素构成的，可以给这些元素的父元素设置transform-style: preserve-3d来使其变成一个真正的3D图形。

见代码示例09 3D转换-练习-立方体.html

**7、backface-visibility**

设置元素背面是否可见

见代码示例01 3D转换-应用-音乐盒.html

[**参考文档**](http://isux.tencent.com/css3/index.html?transform)

**CSS3动画库**

animate.css

## 动画

动画是CSS3中具有颠覆性的特征之一，可通过设置多个节点来精确控制一个或一组动画，常用来实现复杂的动画效果。

**1、必要元素：**

a、通过@keyframes指定动画序列；

b、通过百分比将动画序列分割成多个节点；

c、在各节点中分别定义各属性

d、通过animation将动画应用于相应元素；

**2、关键属性**

a、animation-name设置动画序列名称

b、animation-duration动画持续时间

c、animation-delay动画延时时间

d、animation-timing-function动画执行速度，linear、ease等

e、animation-play-state动画播放状态，running、paused等

f、animation-direction动画逆播，alternate等

g、animation-fill-mode动画执行完毕后状态，forwards、backwards等

h、animation-iteration-count动画执行次数，inifinate等

i、steps(60) 表示动画分成60步完成

**参数值的顺序：**

关于几个值，除了名字，动画时间，延时有严格顺序要求其它随意

## 伸缩布局

CSS3在布局方面做了非常大的改进，使得我们对块级元素的布局排列变得十分灵活，适应性非常强，其强大的伸缩性，在响应式开中可以发挥极大的作用。

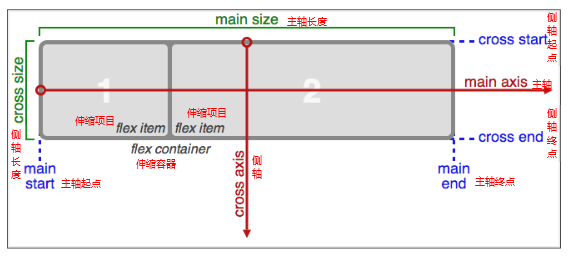
如下图，学习新的概念：

主轴：Flex容器的主轴主要用来配置Flex项目，默认是水平方向

侧轴：与主轴垂直的轴称作侧轴，默认是垂直方向的

方向：默认主轴从左向右，侧轴默认从上到下

主轴和侧轴并不是固定不变的，通过flex-direction可以互换。



**1、必要元素：**

a、指定一个盒子为伸缩盒子 display: flex

b、设置属性来调整此盒的子元素的布局方式 例如 flex-direction

c、明确主侧轴及方向

d、可互换主侧轴，也可改变方向

**2、各属性详解**

a、flex-direction调整主轴方向（默认为水平方向）

b、justify-content调整主轴对齐

c、align-items调整侧轴对齐

d、flex-wrap控制是否换行

e、align-content堆栈（由flex-wrap产生的独立行）对齐

f、flex-flow是flex-direction、flex-wrap的简写形式

g、flex子项目在主轴的缩放比例，不指定flex属性，则不参与伸缩分配

h、order控制子项目的排列顺序，正序方式排序，从小到大

此知识点重在理解，要明确找出主轴、侧轴、方向，各属性对应的属性值可参考示例源码

## 多列布局

类似报纸或杂志中的排版方式，上要用以控制大篇幅文本。

了解即可，实际意义不大。

# Web字体

开发人员可以为自已的网页指定特殊的字体，无需考虑用户电脑上是否安装了此特殊字体，从此把特殊字体处理成图片的时代便成为了过去。

支持程度比较好，甚至IE低版本浏览器也能支持。

## 字体格式

不同浏览器所支持的字体格式是不一样的，我们有必要了解一下有关字体格式的知识。

1、TureType(.ttf)格式

.ttf字体是Windows和Mac的最常见的字体，是一种RAW格式，支持这种字体的浏览器有IE9+、Firefox3.5+、Chrome4+、Safari3+、Opera10+、iOS Mobile、Safari4.2+；

2、OpenType(.otf)格式

.otf字体被认为是一种原始的字体格式，其内置在TureType的基础上，支持这种字体的浏览器有Firefox3.5+、Chrome4.0+、Safari3.1+、Opera10.0+、iOS Mobile、Safari4.2+；

3、Web Open Font Format(.woff)格式

woff字体是Web字体中最佳格式，他是一个开放的TrueType/OpenType的压缩版本，同时也支持元数据包的分离，支持这种字体的浏览器有IE9+、Firefox3.5+、Chrome6+、Safari3.6+、Opera11.1+；

4、Embedded Open Type(.eot)格式

.eot字体是IE专用字体，可以从TrueType创建此格式字体，支持这种字体的浏览器有IE4+；

5、SVG(.svg)格式

.svg字体是基于SVG字体渲染的一种格式，支持这种字体的浏览器有Chrome4+、Safari3.1+、Opera10.0+、iOS Mobile Safari3.2+；

了解了上面的知识后，我们就需要为不同的浏览器准备不同格式的字体，通常我们会通过字体生成工具帮我们生成各种格式的字体，因此无需过于在意字体格式间的区别差异。

推荐http://www.zhaozi.cn/、http://www.youziku.com/ 查找更多中文字体

## 字体图标

其实我们可以把文字理解成是一种特殊形状的图片，反之我们是不是也可以把图片制作成字体呢？

答案是肯定的。

常见的是把网页常用的一些小的图标，借助工具帮我们生成一个字体包，然后就可以像使用文字一样使用图标了。

优点：

1、将所有图标打包成字体库，减少请求；

2、具有矢量性，可保证清晰度；

3、使用灵活，便于维护；

**Font Awesome 使用介绍**

http://fontawesome.dashgame.com/

**定制自已的字体图标库**

http://iconfont.cn/

https://icomoon.io/

**SVG素材**

http://www.iconsvg.com/

# 兼容性

通过http://caniuse.com/ 可查询CSS3各特性的支持程度，一般兼容性处理的常见方法是为属性添加私有前缀，如不能解决，应避免使用，无需刻意去处理CSS3的兼容性问题。

# 高级应用