# 选择器

### 属性选择器

1、E[attr] 表示存在attr属性即可；

2、E[attr=val] 表示属性值完全等于val

3、E[attr~=val] 表示的一个单独的属性值 这个属性值是以空格分隔的

4、E[attr|=val] 表示的要么一个单独的属性值 要么这个属性值是以“-”分隔的

5、E[attr\*=val] 表示的属性值里包含val字符并且在“任意”位置

6、E[attr^=val] 表示的属性值里包含val字符并且在“开始”位置

7、E[attr$=val] 表示的属性值里包含val字符并且在“结束”位置

### 伪类选择器

E:nth-child(n) 第n个子元素，计算方法是E元素的全部兄弟元素

E:nth-of-type(n) 第n个子元素，计算方法只是E元素，会忽略其子元素

E:nth-last-child(n)同E:nth-child(n) 计算顺序相反。从父级看最后一个是否匹配，不匹配不作用，解决这种问题用nth-last-of-type(1) last-of-type first-of-type

E:nth-last-of-type(n) 同E:nth-of-type(n) 计算顺序相反。

n遵循线性变化，其取值1、2、3、4、...

n取值为n>=12、当n做一个系数时，n取值为n>=0者n<0，例如nth-child(2n+1)

-n+5做为一个整体不能小于1；

E:only-child 父元素的独生子

E:only-of-type只有一个同类型的子元素，允许不同类型元素

E:target 结合锚点进行使用，处于当前锚点的元素会被选中；显示target设置的样式

E:empty E没有内容时选定

### 伪元素选择器

E::selection 选中文本的样式

E::placeholder 改变placeholder默认样式，这个存在明显的兼容问题，比如::-webkit-input-placeholder

E:after、E:before 在旧版本里是伪类，在新版本里是伪元素，新版本下E:after、E:before会被自动识别为E::after、E::before，按伪元素来对待。

":" 与 "::" 区别在于区分伪类和伪元素

### 其它选择器

.box::first-letter

.box::first-line

# 颜色

Hue、Saturation、Lightness、Alpha即HSLA

R、G、B 取值范围0~255

H 取值范围0~360，0/360表示黑色、120表示绿色、240表示蓝色

S 取值范围0%~100%

L 取值范围0%~100%

A 取值范围0~1

关于透明度：

1、opacity子元素会继承父元素的透明度，在实际开发中会带来干扰；

2 、transparent 设置透明度时完全类似于“玻璃”一样的透明；

# 文本

文字阴影：左偏移量、上偏移量、模糊度、颜色（可设透明度）。单位(px)

文本溢出：

white-space

多行文本溢出处理可参照下面的方法，但是有比较严重的兼容性，需要慎重选择，比较完备的多行溢出需要JS辅助完成。多行文本文字溢出处理，非标准属性，可应用于移动端

word-break: break-all; /\*不做区分 全部换行 \*/

## /\* 单行...隐藏文本 \*/

## text-overflow: ellipsis;

## overflow: hidden;

## white-space: nowrap;

## /\* 多行...隐藏文本 设置从哪一行隐藏 \*/

## text-overflow: ellipsis;

## overflow: hidden;

## /\* white-space: nowrap; \*/

## display: -webkit-box;

## -webkit-line-clamp: 1;

## -webkit-box-orient: vertical;

word-wrap: break-word; /\* 英文中文换行 \*/

white-space: pre; /\* 保留空格 不会自动换行 \*/

white-space: pre-wrap; /\* 保留空格 自动换行 \*/

white-space: pre-line; /\* 空格合并 换行 \*/

white-space: nowrap; /\* 空格合并 强制一行显示 \*/

# 边框

### 边框圆角

理解before after

可分别设置长、短半径，以“/”进行分隔，遵循“1，2，3，4”规则。

border-radius: 45px 0 0 45px / 90px 0 0 90px; border-radius: 45px / 90px;

表格运用圆角需要要 border-collapse: separate;

当圆角半径小于或等于边框宽度时，元素内角是直角

border:50px solid transparent;

border-image-source: url(border.png);

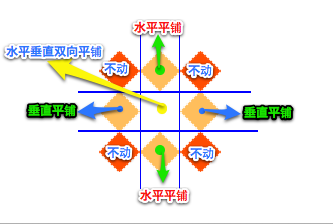
border-image-width: 20px;

border-image-slice: 27;

border-image-repeat: round;

50%可绘制圆和椭圆。

### 边框图片

slice 上右下左设置切割

round 会自动调整尺寸，完整显示边框图片。

repeat 单纯平铺多余部分，会被“裁切”而不显示。

box-shadow: 10px 10px 0px 0px pink,

-10px -10px 0px 0px skyblue,

inset 10px 10px 2px 0px #ccc;

### 边框阴影

设置边框阴影不会影响盒子的布局

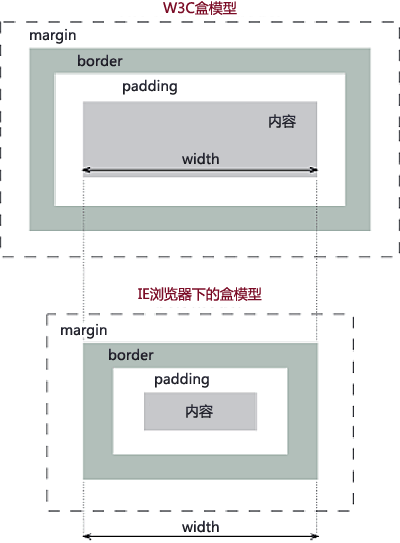
spread可以与blur、h-shadow、v-shadow相互抵消，blur不可为负值

模糊度不能为负。

border:50px solid transparent; Chrome将此属性写在前面。

# 盒模型

关于盒模型存在两种形式，分别是W3C标准盒模型和IE盒模型，如下图所示，其区别主要在于宽度和高度的计算方式，对盒模型做出了新的定义，即允许开发人员指定盒子宽度和高度的计算方式。



IE盒模型只会出现在IE5版本和IE6+的**怪异模式**中。

box-sizing: border-box; 类似IE模型 按padding+border计算总宽

box-sizing: content-box; 默认width为内容

## 背景

background-size会以background-clip设定的盒模型计算。

cover 会使“最大”边，进行缩放，另一边同比缩放，铺满容器，超出部分会溢出。

contain 会使“最小”边，进行缩放，另一边同比缩放，不一定铺满容器，会完整显示图片。

background-size:100% 100%; 铺满背景图片的默认位置。

background-clip: padding-box; //背景不绘制边框底部 从padding位置开始绘制

:content-box; //背景不绘制padding 只绘制内部

background-origin背景图片的起始位置，实际上改变的是background-position的原点值。

background-origin:border-box; //背景图片从边框的左上角开始绘制

:padding-box; //padding的左上角

:content-box; //content的左上角

同时设置origin和clip, clip设置从content开始绘制，origin设置背景图起点为padding左上角。此时背景图只显示一部分。不写origin默认也是padding开始，显示背景图部分。

background组合设置 属性的两个位置是从padding左上开始计算的。

background-size设置的cover/contain属性与background-clip无关，缩放不以它设置的背景绘制区域为参考点，而以background-origin设置的为主。

## 渐变

线性渐变：linear-gradient(to right bottom, #fff, red);

Linear-gradient(135deg, #fff, red);

让坐标轴顺时针旋转角度，渐变方向为Y轴正方向。180deg为默认方向

background-image: linear-gradient(90deg, #fff 30%, red 30%); 

background-image: linear-gradient(90deg, #fff 30%, red 60%); 

径向渐变：radial-gradient(circle at center center, red 30%, yellow 60%);

(ellipse at 50px 50px, red 30%, yellow 60%);

(50px 120px at center center, red 30%, yellow 60%);

重复渐变：repeating-linear-gradient()

repeating-radial-gradient() 具体参数试着设置

# 伸缩布局

旧版：display: -webkit-box;

过渡：display: -webkit-flex;

新版：display: flex;

http://caniuse.com/

指定一个盒子为伸缩盒子 display:flex;

设置属性调整此盒的子元素的布局方式

（，不算文字等内容）

-webkit-box-flex: 1; 均分剩余宽度

flex: 1; 均分宽度

(对齐方式)-webkit-box-align:center; (start左 | end右 | justify分散对其)

(垂直对齐)-webkit-box-pack: center; (垂直居中 )

（反转）-webkit-box-direction: normal; (reverse)

(排列方式) -webkit-box-orient: horizontal; (vertical)

(主轴)justify-content:flex-start; (flex-end center space-between space-around) 水平

(侧轴)align-items:flex-start; (flex-end center **baseline stretch**) 垂直

(子元素侧轴)align-self: center; 同align-items属性值，覆盖父盒子的align-items。

(收缩比例)flex-shrink:1; （默认收缩 | 0不参与收缩 | 值越大收缩比例越大）

(扩展比例)flex-grow: 1; （默认不收缩为0 | ）

(预先分配) flex-basis: 300px;

(方向)flex-direction:column; (column-reverse row row-column)

(换行)flex-wrap:wrap; (默认no-wrap，会缩放)

(行间距)align-content:flex-start;

(flex-end center space-between space-around stretch)

(子元素顺序)order: 2; 按order顺序从小到大展示。

flex-basis在分配空间之前已经分得的像素，也可直接写具体的w/h大小。

注：属性设置成stretch时，height必须设置成auto才能显示效果。

设置了align-items 再设置align-content属性，不显示效果。

设置align-content必须要设置flex-wrap:wrap;

flex的缩写：flex: 1 0 600px; （flex-grow拓展 flex-shrink收缩 flex-basis）

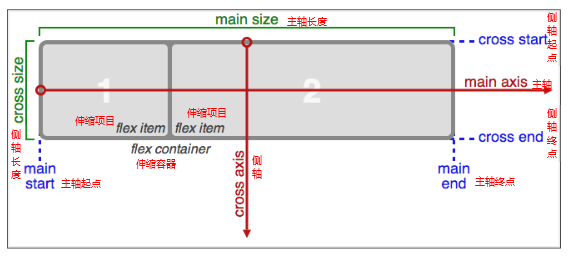
当一个元素过大容不下时，其他元素又不允许参与收缩，此时会换行，不再是全部收缩。

### **新版伸缩布局**

主轴：Flex容器的主轴主要用来配置Flex项目。

侧轴：与主轴垂直的轴称作侧轴，是侧轴方向的延伸。

主轴和侧轴并不是固定不变的，通过flex-direction可以调整。



**兼容性**

/\* 子元素-平均分栏 \*/

.flex1 {

-webkit-box-flex: 1; /\* OLD - iOS 6-, Safari 3.1-6 \*/

-moz-box-flex: 1; /\* OLD - Firefox 19- \*/

width: 20%; /\* For old syntax, otherwise collapses. \*/

-webkit-flex: 1; /\* Chrome \*/

-ms-flex: 1; /\* IE 10 \*/

flex: 1; /\* NEW, Spec - Opera 12.1, Firefox 20+ \*/

}/\* 父元素-横向排列（主轴） \*/

.flex-h {

display: box; /\* OLD - Android 4.4- \*/

display: -webkit-box; /\* OLD - iOS 6-, Safari 3.1-6 \*/

display: -moz-box; /\* OLD - Firefox 19- (buggy but mostly works) \*/

display: -ms-flexbox; /\* TWEENER - IE 10 \*/

display: -webkit-flex; /\* NEW - Chrome \*/

display: flex; /\* NEW, Spec - Opera 12.1, Firefox 20+ \*/

/\* 09版 \*/

-webkit-box-orient: horizontal;

/\* 12版 \*/

-webkit-flex-direction: row;

-moz-flex-direction: row;

-ms-flex-direction: row;

-o-flex-direction: row;

flex-direction: row;

}/\* 父元素-横向换行 \*/

.flex-hw {

/\* 09版 \*/

/\*-webkit-box-lines: multiple;\*/

/\* 12版 \*/

-webkit-flex-wrap: wrap;

-moz-flex-wrap: wrap;

-ms-flex-wrap: wrap;

-o-flex-wrap: wrap;

flex-wrap: wrap;

}/\* 父元素-水平居中（主轴是横向才生效） \*/

.flex-hc {

/\* 09版 \*/

-webkit-box-pack: center;

/\* 12版 \*/

-webkit-justify-content: center;

-moz-justify-content: center;

-ms-justify-content: center;

-o-justify-content: center;

justify-content: center;

/\* 其它取值如下：

align-items 主轴原点方向对齐

flex-end 主轴延伸方向对齐

space-between 等间距排列，首尾不留白

space-around 等间距排列，首尾留白

\*/

}/\* 父元素-纵向排列（主轴） \*/

.flex-v {

display: box; /\* OLD - Android 4.4- \*/

display: -webkit-box; /\* OLD - iOS 6-, Safari 3.1-6 \*/

display: -moz-box; /\* OLD - Firefox 19- (buggy but mostly works) \*/

display: -ms-flexbox; /\* TWEENER - IE 10 \*/

display: -webkit-flex; /\* NEW - Chrome \*/

display: flex; /\* NEW, Spec - Opera 12.1, Firefox 20+ \*/

/\* 09版 \*/

-webkit-box-orient: vertical;

/\* 12版 \*/

-webkit-flex-direction: column;

-moz-flex-direction: column;

-ms-flex-direction: column;

-o-flex-direction: column;

flex-direction: column;

}/\* 父元素-纵向换行 \*/

.flex-vw {

/\* 09版 \*/

/\*-webkit-box-lines: multiple;\*/

/\* 12版 \*/

-webkit-flex-wrap: wrap;

-moz-flex-wrap: wrap;

-ms-flex-wrap: wrap;

-o-flex-wrap: wrap;

flex-wrap: wrap;

}/\* 父元素-竖直居中（主轴是横向才生效） \*/

.flex-vc {

/\* 09版 \*/

-webkit-box-align: center;

/\* 12版 \*/

-webkit-align-items: center;

-moz-align-items: center;

-ms-align-items: center;

-o-align-items: center;

align-items: center;

}/\* 子元素-显示在从左向右（从上向下）第1个位置，用于改变源文档顺序显示 \*/

.flex-1 {

-webkit-box-ordinal-group: 1; /\* OLD - iOS 6-, Safari 3.1-6 \*/

-moz-box-ordinal-group: 1; /\* OLD - Firefox 19- \*/

-ms-flex-order: 1; /\* TWEENER - IE 10 \*/

-webkit-order: 1; /\* NEW - Chrome \*/

order: 1; /\* NEW, Spec - Opera 12.1, Firefox 20+ \*/

}/\* 子元素-显示在从左向右（从上向下）第2个位置，用于改变源文档顺序显示 \*/

.flex-2 {

-webkit-box-ordinal-group: 2; /\* OLD - iOS 6-, Safari 3.1-6 \*/

-moz-box-ordinal-group: 2; /\* OLD - Firefox 19- \*/

-ms-flex-order: 2; /\* TWEENER - IE 10 \*/

-webkit-order: 2; /\* NEW - Chrome \*/

order: 2; /\* NEW, Spec - Opera 12.1, Firefox 20+ \*/

}

为了更好的兼容性，我们需要给容器声明flex-h/flex-v，而不是一般的flex：

/\* 父元素-flex容器 \*/

.flex {

display: box; /\* OLD - Android 4.4- \*/

display: -webkit-box; /\* OLD - iOS 6-, Safari 3.1-6 \*/

display: -moz-box; /\* OLD - Firefox 19- (buggy but mostly works) \*/

display: -ms-flexbox; /\* TWEENER - IE 10 \*/

display: -webkit-flex; /\* NEW - Chrome \*/

display: flex; /\* NEW, Spec - Opera 12.1, Firefox 20+ \*/

}

## 新属性

1. <div contenteditable="true"></div>可用CSS3新属性替代：

-webkit-user-modify:read-write;（read-only | read-write-plaintext-only 富文本粘贴会丢失）

## 多列布局

-webkit-column-count: 4; /\* 列数 \*/

-webkit-column-gap: 5px; /\* 列间隔的宽 \*/

-webkit-column-width: 100px; /\* 列宽设置不合理会自动调整列数 \*/

-webkit-column-rule: 2px dashed #ccc; /\* 间隔竖线 \*/

h4 {

-webkit-column-span: all;

}/\*该属性设置后，后面两行不在h4的行上显示\*/

# 过渡

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

transition组合写法：属性 | 时间 | 延迟时间 | 运动的类型

# 动画

多次可重复动画

animation-name: change;

animation-duration: 2s;

animation-fill-mode: forwards; /\*保持状态\*/ none | backwards(起始点) | both

animation-iteration-count: infinite; /\*无限循环\*/ number

animation-delay: 1s; /\*延时\*/

animation-play-state: paused; /\*暂停\*/ running

animation-direction: alternate; /\*倒方向\*/ 反向运动计次

animation-timing-function: linear; /\*速度\*/

@keyframe change { ...... }

可以hover暂停动画。

# 阴影

-webkit-box-reflect: below 10px -webkit-linear-gradient(t,t 50%, rgba(255,255,255,.6));

# 图片filter



# 2D/3D旋转

**2D**:

（1）平移

transform: translate(40px, 30px); x,y方向平移组合写法，传一个值默认X

transform: translateX(40px);

transform: translateY(30px); 这两个属性不能同时起作用，后面覆盖前面的。

（2）缩放

transform: scale(1.2);

transform: scaleX(0.5);

transform: scaleY(0.5);

1. 旋转 正值顺时针，负值逆时针。

transform: rotate(36deg);

transform: rotateX(180deg);

transform: rotateY(180deg);

1. 斜切

transform: skew(45deg);

transform: skewX(45deg); 横不变

transform: skewY(45deg); 竖不变

1. 变换的轴心点 transform-origin: top left; 也可用像素

**3D:**

必须先加景深perspective: 200px; （值越小离物体越近，推荐500~800）

再开启3D transform-style: preserve-3d;

然后才能看到Z轴变换的各种效果

## pointer-events: none; 屏蔽鼠标事件。

# 移动端布局

**2.1**