# \*\*\*\*\*\*\*css3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# css3新属性特点

尽管元素的外形或位置发生了变化,但是元素在文档流中的位置是不变的,元素的变化并不会影响其他兄弟元素,也不会影响子元素。

# 1.选择器

## 1.1基本选择器

### 1.1.1子元素选择器

1)定义: 只能选择某元素的子元素

2)语法: 父元素>子元素

### 1.1.2相邻兄弟选择器

1)定义: 可以选择紧接着在另外一个元素后的元素,而且他们具有一个相同的父元素

2)语法: 元素 + 相邻兄弟元素

### 1.1.3通用兄弟选择器

1)定义: 选择某元素后边的所有兄弟元素,而且他们具有一个共同的父亲

2)语法: 元素 ~ 后面所有兄弟元素

### 1.1.4群组选择器

1)定义: 将具有相同样式的元素分组在一起,每个选择器之间用逗号 , 隔开

2) 语法: 元素1,元素2,元素3,.......

### 1.1.5交集选择器(选择后代选择器)

li.box { }

## 1.2.属性选择器

一般给input元素使用

### 1.2.1 element[attribute]

1)定义: 为带有attribute属性的元素设置样式

2)语法: element[attribute]

li[class]

### 1.2.2 element[attribute='value']

1)定义: 为attribute='value'属性的元素设置样式

2)语法: element[attribute='value']

li[class='box']

### 1.2.3 element[attribute~='value']

1) 定义: 选择attribute属性包含单词value的元素,并设置样式

2) 语法: element[attribute~='value']

li[class~='box']

### 1.2.4 element[attribute^='value']

1)定义: 设置attribute属性值,以value开头的所有元素样式

2)语法: element[attribute^='value']

li[class^='box']

### 1.2.5 element[attribute$='value']

1) 定义: 设置attribute属性值,以value结尾的所有元素样式

2) 语法: element[attribute$='value']

li[class$='box']

### 1.2.6 element[attribute\*='value']

1)定义: 设置attribute属性值,包含value的所有元素,并为其设 置样式

2)语法: element[attribute\*='value']

li[class\*='box']

## 1.3.伪类

### 1.3.1锚点伪类

:link

:visited

### 1.3.2用户行为伪类

:hover

:active

:focus

获取焦点 光标闪烁表示获取焦点

/\*outline:none; 设置外轮廓隐藏\*/

### 1.3.3:target目标伪类

#box {  
 margin-top: 2000px;  
 width: 200px;  
 height: 200px;  
 background: deeppink;  
 }  
 /\*:target伪类用来添加额外样式\*/  
 #box:target{  
 border: 10px solid #000000;  
 }

<!--当我们点击锚链接时,对应ID的元素会显示在视口-->

<a href="#box"></a>

<div id="box"></div>

### 1.3.4:checked状态伪类

清除input默认样式

-webkit-appearance: none;

-moz-appearance: none;

appearance: none;

属性加厂商前缀的意义:

1.该属性是浏览器特有的

2.该属性处在测试阶段

浏览器 浏览器内核 厂商前缀

chrome Webkit -webkit-

firefox Gecko -moz-

IE Trident -ms-

Opera Webkit

Safari Webkit

大部分手机 Webkit

### 1.3.5 CSS3结构类

选择器中的element为子元素,既所要选择的元素

1)first-child:

选择属于其父元素的首个子元素的每个element元素,并为其设置样式(element:first-child)

2)last-child:

选择属于其父元素的最后一个子元素的每个element元素,并为其设置样式(element:last-child)

3)nth-child(n):

选择某元素下第number个element元素(n是一个简单的表达式,例如2n+1,不能用其他字母代替,n从0开始计算)

1.nth-child(odd):可用于匹配下标是奇数的元素的关键字

2.nth-child(even):可用于匹配下标是偶数的元素的关键字

4)nth-last-child():

匹配属于其元素的第n个元素的每个元素,从最后一个子元素开始计数(element:nth-last-child(n))

5)nth-of-type():

匹配属于父元素的特定类型的第n个子元素(element:nth-of-type())

6)nth-last-of-type():

选匹配属于父元素的特定类型的第n个子元素,从最后一个开始计数(element:nth-last-of-type())

7)first-of-type():

匹配属于其父元素的特定类型的首个子元素的每个元素(element: first-of-type())

8)last-of-type():

匹配属于其父元素的特定类型的最后一个子元素的每个元素(element: last-of-type())

9):only-child:

匹配属于其父元素的唯一一个子元素的每个元素(element:only-child)

10):only-of-type:

匹配属于其父元素的特定类型的唯一一个子元素的每个元素(element :only-of-type)

11):empty:

匹配没有子元素(包括文本节点)的每个元素(element :empty ---- div:empty)

### 1.3.6否定选择器

匹配非 元素或者选择器 的每个元素

父元素: not(子元素或者选择器)

例: ul:not(span)

## 1.4.伪元素

### 1.4.1 element::first-line

对元素的第一行文本进行设置,只能用于块级元素

### 1.4.2 element::first-letter

用于对文本的首字母设置特殊样式,只能用于块级元素

### 1.4.3 element::before

在元素的内容前面插入新内容,常与content配合使用

### 1.4.4 element::after

在元素的内容后面插入新内容,常与content配合使用

### 1.4.5 element::selection

用于设置浏览器中选中文本后的背景色与前景色

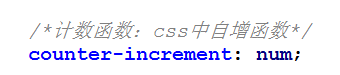
### 1.4.6使用伪元素注意事项

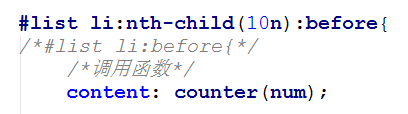
1)使用伪元素before,after必须设置content

2)使用伪元素before,after显示背景图,一定要使用display设置为块元素

3)使用伪元素before,after设置为display:inline\_block,需要再次设置vertcal-align:middle

### 1.4.7css中的函数





注: 实参和形参要一样

## 1.5.权重

!important > 行内样式(1000)> ID选择器(100)>类,属性选择器和伪类选择器(10)> 元素选择器和伪元素选择器(1)>通配符选择器(0)>继承的属性

(注意: 尽量不要使用！important)

# 2.圆角边框

## 2.1 Border-radius

### 2.1.1定义

可以为元素添加圆角边框(块元素,行内块元素,行内元素)

### 2.1.2 语法使用

1)四个值: 左上角 右上角 右下角 左下角

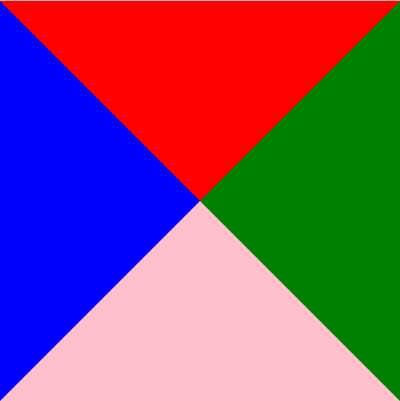
2)三个值: 左上角 右上角和左下角 右下角

3)两个值: 左上角和右下角 右上角和左下角

4)一个值: 4个角都生效

## 2.2案例

1)边框实现三角形



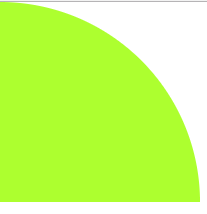
2)搜狐评论框



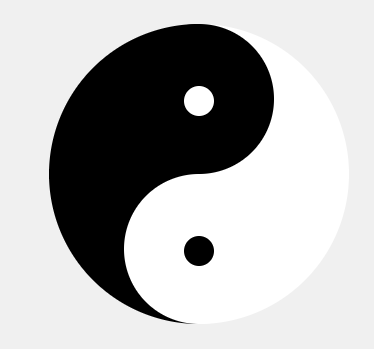
3)半圆



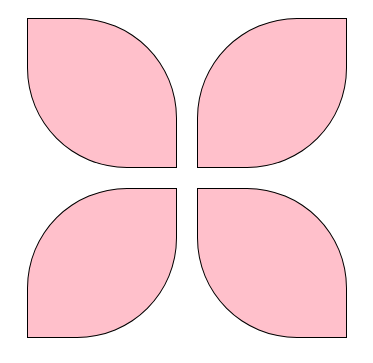
4)扇形



5)太极图



6)风车



# 3.盒子模型

box-sizing:content-box; 标准盒子模型

box-sizing:border-box; 怪异盒子模型

# 4.文字特殊样式

## 4.1文字阴影

text-shadow:水平方向偏移量 垂直方向偏移量 模糊程度 颜色；

/\*设置多个阴影,不同阴影之间用逗号隔开\*/

text-shadow:水平方向偏移量 垂直方向偏移量 模糊程度 颜色  **,** 水平方向偏移量 垂直方向偏移量 模糊程度 颜色 ；



## 4.2浮雕文字

color:#fff;

text-shadow: 2px 2px 8px #000;



## 4.3文字模糊

#box:hover{

/\*文字透明\*/  
 color: transparent;

/\*模糊文字背景\*/  
 text-shadow: 0px 0px 80px rgba(0,0,0,0.4);  
}

过渡

#box{

transition: 2s;

}



## 4.4文字描边

-webkit-text-stroke: 粗细 颜色;



## 4.5文字排版

direction: ltr; /\*默认从右到左 left to right\*/

direction: rtl; /\*从右到左\*/  
 unicode-bidi: bidi-override;/\*文字反序\*/

## 4.6文字背景裁剪

-webkit-background-clip: text;

/\*添加背景图\*/

background: url("###");

background-repeat: no-repeat;  
 background-position: ;

/\*设置文字透明\*/

color: rgba(255,0,0,0);

color:transparent;

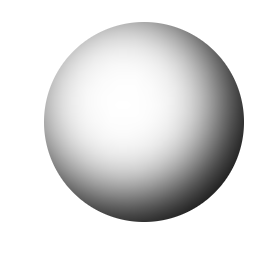


# 5.盒子阴影

box-shadow:水平偏移量 垂直偏移量 模糊程度 扩展程度 颜色 是否是内阴影 (是写上inset)；

扩展程度: 扩展阴影的大小

## 5.1立体球



#wrap {  
 margin: 100px auto;  
 width: 200px;  
 height: 200px;  
 border-radius: 50%;  
 box-shadow: -20px -20px 100px 0px black inset;  
 }

## 5.2重复盒子阴影



#wrap {  
 margin: 100px auto;  
 width: 200px;  
 height: 10px;  
 background: gray;  
 box-shadow: 0px 20px 0px 0px gray,0px 40px 0px 0px gray,0px 60px 0px 0px gray;  
 }

## 5.3模糊盒子阴影



1.元素的宽度或高度要设置成0

2.通过扩展程度来扩展阴影的大小

3.通过模糊程度设置阴影效果

#wrap {  
 margin:100px auto;  
 width: 200px;  
 height: 0px;  
 background: deeppink;  
 box-shadow: 0px 0px 50px 10px deeppink;  
 }

## 5.4盒子倒影(拓展)

-webkit-box-reflect:方向 盒子与倒影的距离 倒影的效果(线性渐变);

方向: left right above below

效果: linear-gradient(90deg,rgba(0,0,0,1),rgba(0,0,0,0))

-webkit-box-reflect: left 10px linear-gradient(90deg,rgba(0,0,0,1),rgba(0,0,0,0));  
 -webkit-box-reflect: right 10px linear-gradient(90deg,rgba(0,0,0,0),rgba(0,0,0,1));  
 -webkit-box-reflect: above 10px linear-gradient(0deg,rgba(0,0,0,0),rgba(0,0,0,1));  
 -webkit-box-reflect: below 10px linear-gradient(0deg,rgba(0,0,0,1),rgba(0,0,0,0));



## 5.5调整盒子大小

overflow:auto;

resize: horizontal; /\*水平调整\*/  
 resize: vertical;/\*垂直调整\*/  
 resize: both;/\*两者都有\*/



# 6.背景

背景颜色、背景图片填充的盒子有content-box、padding-box、border-box

背景图片的起始位置在padding-box的左上角

子元素的起始位置在父元素的content-box的左上角

## 6.1背景裁剪

background-clip: border-box; /\*默认值\*/

padding-box

content-box

背景的起始位置不变,盒子以外的裁剪掉,只留下里面的

## 6.2背景起始位置

background-origin: padding-box;/\*默认值\*/

border-box

content-box

background-position: 0px 0px; 配合使用

## 6.3背景图像的大小

background-size: number / % / cover / contain;

background-size:宽度 高度(如果只写一个数值,第二个数值默认auto)

百分比:0% - 100%之间的任何值,此时的百分比参照于元素div的大小

cover:将背景图片以容器最远边进行缩放(填充)

contain:将背景图片以容器最近边进行缩放(适应)

## 6.4多重背景

background-image: CSS3允许您为元素使用多个背景图片

background-image: url('1.jpg),url('2.jpg') ... 使用逗号把图片分开

元素引入多个背景图片,前面图片会覆盖后面的图片

## 6.5案例: 背景模糊(高斯模糊)

/\*高斯模糊: 图片设置\*/  
 /\*过滤器 filter: blur();\*/  
 filter: blur(5px);

---------------------------------------------------------------

<!--模糊区域-->  
 <div id="mohu">  
 <img src="img/timg.jpg" alt="">  
 </div>  
 <!--内容区域-->  
 <div id="wrap">  
 尚硅谷  
 </div>

注: 图片和文字不在同一个块元素内,图片使用插入图,不能使用背景图

# 7.多列

## 7.1定义

使用CSS3,能够创建多个列来对文本进行布局 ( 就像报纸那样)

## 7.2属性

### 7.2.1 [column-count](http://www.w3school.com.cn/cssref/pr_column-count.asp" \o "CSS3 column-count 属性)

1)定义: 规定元素应该被分隔的列数

2)语法: column-count: number;

### 7.2.2 [column-f](http://www.w3school.com.cn/cssref/pr_column-count.asp" \o "CSS3 column-count 属性)ill

1)定义: 规定如何填充列

2)语法: column-fill: balance | auto;

### 7.2.3 [column-](http://www.w3school.com.cn/cssref/pr_column-count.asp" \o "CSS3 column-count 属性)gap

1)定义: 规定列之间的间隔

2)语法: column-gap: 像素值;

### 7.2.4 [column-](http://www.w3school.com.cn/cssref/pr_column-count.asp" \o "CSS3 column-count 属性)rule

1)定义: 所有 column-rule-\* 属性的简写属性(颜色,样式,宽度)

2)语法: column-rule: 颜色,样式,宽度;

3) 列之间规则的颜色: [column-rule-color](http://www.w3school.com.cn/cssref/pr_column-rule-color.asp" \o "CSS3 column-rule-color 属性)

列之间规则的样式: [column-rule-style](http://www.w3school.com.cn/cssref/pr_column-rule-style.asp" \o "CSS3 column-rule-style 属性)

列之间规则的宽度: [column-rule-width](http://www.w3school.com.cn/cssref/pr_column-rule-width.asp" \o "CSS3 column-rule-width 属性)

### 7.2.5 [column-](http://www.w3school.com.cn/cssref/pr_column-count.asp" \o "CSS3 column-count 属性)span

1)定义: 元素应该横跨的列数

2)语法: column-span:number

### 7.2.6 [column-](http://www.w3school.com.cn/cssref/pr_column-count.asp" \o "CSS3 column-count 属性)width

1)定义: 列的宽度

2)语法: column-width:像素值

## 7.3案例

### 7.3.1文本多列

.article{

/\*列数\*/  
 /\*column-count: 5;\*/  
 /\*列宽\*/  
 column-width: 200px;  
 /\*列间隔\*/  
 column-gap: 5px;  
  
 column-rule: 5px solid red;  
  
 height: 1000px;  
  
 /\*column-fill: balance;!\*默认值\*!\*/  
 column-fill: auto;  
}  
.article h1{  
 /\*column-span: 1;\*/  
 column-span: all;  
 text-align: center;  
}

### 7.3.2图片多列(瀑布流)

#imgs {  
 column-width: 230px;  
 column-gap: 5px;  
  
 column-rule: 2px solid green;  
 /\*font-size: 0;\*/  
 }  
 #imgs img{

width: 100%;  
 /\*float: left;\*/  
 display: block;  
}

-----------------------------------------------------------------------------

<div id="imgs">  
 <img src="img/img10.jpg" alt="">

......

<img src="img/img20.jpg" alt="">  
</div>

# 渐变

## 8.1线性渐变

### 8.1.1渐变本质

渐变的本质是绘制背景图片,可以使用background或background-image:;来引用,同样可以设置background-size,background-position。渐变绘制的背景图大小默认是100%。

### 8.1.2语法

background-image:linear-gradient(方向,开始颜色,结束颜色)

颜色最少写两个

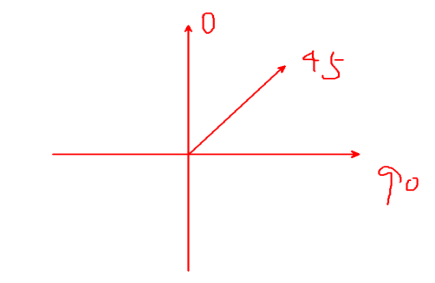
### 8.1.3渐变方向

1) 从上到下(默认): background: linear-gradient(red,blue);

2) 从左到右: background: linear-gradient(to right,red,blue);

3) 对角: background: linear-gradient(to right bottom,red,blue);

4) 角度:



角度单位deg

### 8.1.4 颜色结点

默认每个颜色均匀分布

1) background: linear-gradient(red 10%,blue 20%,green 30%,yellow 40%);

从0%到10%,为红色,从10%到20%为红色到蓝色的渐变,从20%到30%为蓝色到绿色的渐变,从30%到40%,为绿色到黄色的渐变

2) background: linear-gradient(red 10%,blue);

从0%到10%,为红色,从10%到100%为红色到蓝色的渐变,最后如果不写具体数值,默认到100%

3) background: linear-gradient(red,blue 30%);

从0%到30%,为红色到蓝色的渐变,如果第一个不写,默认数值是 0%

4) background: linear-gradient(rgba(255,0,0,0),rgba(255,0,0,1));

由透明色变为不透明色

注: 百分比: 把元素渐变方向的整体长度看成100%

## 8.2径向渐变

### 8.2.1定义

background-image: 从起点到终点,颜色从内向外进行圆形渐变

### 8.2.2语法

background:radial-gradient(形状尺寸,开始颜色,结束颜色)

### 8.2.3形状分类

1) 圆形circle

2) 椭圆形ellipse

3) 注意: 当元素的高和宽一样时,参数无论设置谁,都是圆形

### 8.2.4 颜色结点

background: radial-gradient(circle,red 30% ,blue 70%);

注意: 此时的百分比,指的是圆心到元素最远端(对角线)的距离(角度)

### 8.2.5尺寸大小

1) closest-sid e最近边

background: radial-gradient(closest-side circle,red , blue);

2) farthest-side 最远边

background: radial-gradient(farthest-side circle,red , blue);

3) closest-corner最近角

background: radial-gradient(closest-corner circle,red , blue);

4) farthest-corner最远角

background: radial-gradient(farthest-corner circle,red , blue);

注: 调节圆心的位置,在第一个参数后面加 at 50px 50px

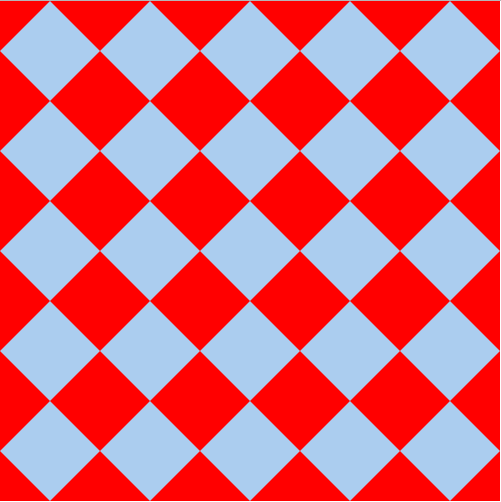
### 8.2.6 重复渐变

background: repeating-radial-gradient(red 0%,blue 20%);

background: repeating-radial-gradient(red 0%,blue 10%,red 20%);

## 8.3案例

8.3.1马赛克地砖



#wrap {  
 width: 500px;  
 height: 500px;  
 background-image: linear-gradient(45deg,red 25%,rgba(255,0,0,0) 25%,rgba(255,0,0,0) 75%,red 75%),linear-gradient(135deg,red 25%,rgba(255,0,0,0) 25%,rgba(255,0,0,0) 75%,red 75%);  
 background-size:20%;  
 background-color: #abcdef;  
 }

8.3.2发廊灯



#box {  
 width: 1000px;  
 height: 50px;  
 background: repeating-linear-gradient(135deg, rgba(0, 0, 0, 1) 0px, rgba(0, 0, 0, 1) 10px, rgba(255, 255, 255, 1) 10px, rgba(255, 255, 255, 1) 20px);  
 }  
 #wrap:hover #box{  
 margin-left: -800px;  
 transition: 10s;  
 }

---------------------------------------------------------------------------------

<div id="wrap">  
 <div id="box"></div>  
 </div>

8.3.3光斑动画



body {background: #000;}  
 #wrap {  
 margin-top: 150px;  
 height: 200px;  
 font-size: 80px;  
 font-weight: bold;  
 text-align: center;  
 line-height: 200px;  
 color: rgba(255, 255, 255, 0.5);  
 background: linear-gradient(135deg, rgba(255, 255, 255, 0) 140px, rgba(255, 255, 255, 1) 210px, rgba(255, 255, 255, 0) 280px);  
 -webkit-background-clip: text;  
 }

------------------------------------------------------------------------------------------

<div id="wrap">atguigu尚硅谷</div>

------------------------------------------------------------------------------------------

var box = document.getElementById('wrap');  
 var num = 0;  
 setInterval(function () {  
 num++;  
 if (num > 850) {  
 num = 0  
 }  
 box.style.backgroundPosition = num + 'px 0px'  
 }, 1)

# 9.过渡

## 9.1定义

允许css的属性值在一定时间区间内平滑的过渡,在鼠标点击,鼠标滑过或对元素任何改变中触发,并圆滑地以动画改变css的属性值

## 9.2属性

9.1.1 transition-property

1) 定义: 设置对象中的参与过渡的属性

2) 语法: transition-property: none | all | property

3) 参数说明:

none: 没有属性改变

**⭐all : 默认值,所有属性都改变⭐(面试题)**

property: 元素的属性名 color等

9.1.2 transition-duration

1) 定义: 设置对象过渡的持续时间

2) 语法: transition-duration: time

3) 参数说明:

规定完成过渡效果需要花费的时间,以秒或者毫秒计,默认值0

9.1.3 transition-timing-function

1) 定义: 设置对象中过渡的动画类型

2) 语法: transition-timing-function: 动画类型(只能使用一种)

3) 参数说明:

linear:线性过渡(匀速)cubic-bezier(0,0,1,1)

ease:平滑过渡(慢--快--慢),默认值 cubic-bezier(0.25,0.1,0.25,1)

ease-in:慢--快 cubic-bezier(0.42,0,1,1)

ease-out:快--慢 cubic-bezier(0,0,0.58,1)

ease-in-out:慢--快--慢 cubic-bezier(0.42,0,0.58,1)

贝塞尔曲线

transition-timing-function:贝塞尔曲线

9.1.4 transition-delay

1) 定义: 设置对象延迟的过渡时间

2) 语法: transition-delay: time

3) 参数说明:

过渡效果开始之前要等待的时间,默认是0

9.1.5 transition

1) 定义: 设置对象变换时的过渡

2) 语法: transition: property duration timing-function delay；

3) 参数说明:

时间顺序不能乱,其他参数位置不限,过渡时间在前,延迟时间在后

如果想给多个属性添加不同的过度,参数之间使用逗号分开

注: 当过渡写到hover中时,只有hover时有效果；

## 9.3过渡注意问题

### 1.覆盖操作:

transition: width 2s linear 3s , 2s ;

### 2.不是所有属性都能加过渡

元素变化的时候是否有中间的过程,如果有就可以加过渡,如果没有就不能加,加上也没用。

display: none;不能用

opacity: 0;/\*元素透明度\*/

### 3.在元素没有渲染完的情况下,过渡不生效;在元素渲染完的情况下,过渡生效

1) 开启延迟定时器  
 setTimeout(function () {  
 //box 背景色 yellowgreen  
 var box = document.getElementById('box');  
 box.style.background = 'yellowgreen';  
 },2000)

2) window.onload保证页面中所有资源全部都渲染(加载)完  
 资源 : div元素 ,css , img ,外部文件  
 window.onload = function () {  
 var box = document.getElementById('box');  
 box.style.background = 'yellowgreen';  
 }

### 4.用来监测过渡结束的方法

transitionend

4.1绑定事件的时候必须用

addEventListener('transitionend',function () {})

4.2兼容问题

addEventListener('webkitTransitionEnd',function () {})

4.3解绑操作

removeEventListener('transitionend',function () {})

# 10.变换

## 10.1定义

让一个元素在一个坐标系统中变形,这个属性包含一系列的变形函数,可以移动,旋转,缩放元素。2D变换和3D变换对于display: inline元素都不支持。

## 10.2 2D变换

### 10.2.1 旋转

1) 定义: 通过指定一个角度参数,对元素指定一个2D的旋转

2) 语法: transform: rotate(角度) 单位deg

3) 注意: 旋转角度正数表示顺时针旋转,负数表示逆时针旋转

### 10.2.2 平移

1) 定义: 根据X轴和Y轴的位置给定参数,使当前元素位置移动

2) 语法:

transform: translateX() --- 仅水平方向移动

transform: translateY() --- 仅垂直方向移动

transform: translate( X, Y) --- 水平方向和垂直方向同时移动

注: 单位px,如果只写一个参数,第二个默认是0,只有水平方向上的位移,与translateX效果一样

### 10.2.3 缩放

1) 定义: 设置元素的缩放程度

2) 语法:

transform: scaleX() --- 仅水平方向缩放

transform: scaleY() --- 仅垂直方向缩放

transform: scale(x,y) --- 使元素垂直和水平方向同时缩放

注: 没有单位,括号内为倍数,为1与原本效果等大,为0元素隐藏(看不见), 如果只写一个参数,元素垂直和水平方向同时缩放

### 10.2.4 扭曲/倾斜

1) 定义: 设置元素的倾斜状态

2) 语法:

transform: skewX() --- 仅使元素在水平方向上扭曲变形

transform: skewY() --- 仅使元素在垂直方向上扭曲变形

transform: skew(x,y) --- 使元素在水平方向和垂直方向上扭曲变形

注: 单位deg,0deg与180deg效果一样；扭曲角度为90deg时,元素垂直于屏幕(元素看不见),只写一个参数与水平扭曲的效果一样

### 10.2.5 变换基点

1) 定义: 元素饶着某一点旋转、缩放、扭曲,这个点就是变换基点

2) 语法: transform-origin: 水平方向 垂直方向

3) 默认变换基点:

rotate scale skew几何元素的正中心

translate 参照自身位置移动,不算变换基点

4) 改变变换基点

1.关键字

transform-origin: center; /\*默认\*/

transform-origin: center center; /\*默认\*/

transform-origin: left; (right top bottom)

和背景图位置差不多,给谁找变换基点

2.具体像素值

transform-origin: width/2 height/2; /\*默认\*/

transform-origin: 0px 0px; /\*左上角\*/

3.百分比

transform-origin: 50% 50%;/\*默认\*/

### 10.2.6元素的水平垂直居中

第一种:

position: absolute;  
 top: 0;  
 left: 0;  
 right: 0;  
 bottom: 0;  
 margin: auto;  
第二种:   
 position: absolute;  
 top:50%;  
 left: 50%;  
 margin-top: -100px;  
 margin-left: -100px;  
第三种:   
 元素宽度与高度是否已知,都能使用该方法  
 position: absolute;  
 top:50%;  
 left: 50%;  
 /\*transform: translate(-100px,-100px);\*/  
 transform: translate(-50%,-50%);

### 10.2.7综合变换

transform: rotate(360deg) scale(1.5);

在两个效果之间加空格,不能加逗号。

1.如果scale写在translate之前,元素的大小发生缩放,同时translate的数值也会发生缩放

2.rotate旋转的同时,不光元素旋转,同时坐标轴也发生了旋转,如果rotate写在translate之前,那么元素的translate要参照新的坐标轴进行位移。

3.skew扭曲的同时,元素发生了扭曲,同时坐标轴也发生了扭曲(skewX,Y轴发生扭曲；skewY,X轴发生扭曲；扭曲之后X轴Y轴之间的夹角仍是90度 ),如果skew写在translate之前,那么元素的translate要参照新的坐标轴进行位移。

## 10.3 3D变换

通常是父元素包含子元素

### 10.3.1 开启3D空间(3D舞台)

transform-style: preserve-3d(一般对父元素设置)

### 10.3.2设置景深

景深(灭点): 实现元素近大远小的效果

1.父元素设置景深: perspective: 300px;

2.子元素设置景深: transform: perspective (300px)

注:

1. 景深的数值不能为负数。

2. 当数值为0时,没有效果。

3. 景深数值越大,元素的效果越接近2d变换的效果。

### 10.3.3 设置3D变换

transform: rotateX ()

1.括号内为正值的时候元素左上角往屏幕里面走

2.括号内为90度、270度时元素垂直于屏幕,元素看不见

3. 0度和360度效果一样

transform: rotateY ()

1.括号内为正值的时候元素左上角往屏幕外面走

2.括号内为90度、270度时元素垂直于屏幕,元素看不见

3. 0度和360度效果一样

transform: rotateZ ()

效果和rotate2d变换效果一样

transform: translateZ ()

Z轴是垂直屏幕向外的

### 10.3.4 变换基点

transform-origin: ;

默认值:

transform-origin: center; /\*top left bottom right\*/

transform-origin: 50% 50% 0 ;

transform-origin: 150px 150px 0px;

z的值只能使用具体的像素,不能用百分比。

### 10.3.5景深中心点

默认值:

perspective-origin: center;

/\*top left right bottom\*/

### 10.3.6元素背面是否可见

backface-visibility: visible ;(默认值: 可见)

backface-visibility: hidden; 不可见

# 11.动画

## 11.1关键帧@keyframes

用来决定动画变化的关键位置

语法:

@keyframes animationname{

keyframes-selector{

cssStyles;

}

}

参数说明:

animationname: 动画的名称

keyframes-selector: 关键帧选择器,动画持续时间的百分比

0% - 100%之间, 或者使用form和to关键字也可以设置,form代表0%,to代表100%

## 11.2 animation属性

### 11.2.1 animation-name

1)定义: 设置动画名称

2)语法: animation-name: keyframename | none

3)参数说明:

keyframename: 指定要绑定到选择器的关键帧的名称

### 11.2.2 animation-duration

1)定义: 设置动画的持续时间

2)语法: animation-duration: time

3)参数说明:

单位是s或者ms,指定对象播放完成需要花费的时间,默认值是0

### 11.2.3 animation-timing-function

1)定义: 设置动画的过渡类型

2)参数说明:

检测的是关键帧的区间,并不是整个动画的过程。

linear:线性过渡(匀速)

ease:平滑过渡(0--慢--快--慢),默认值

steps() 括号内是数字,在关键帧里面跳

ease-in:慢--快

ease-out:快--慢

ease-in-out:慢--快--慢

贝塞尔曲线

### 11.2.4 animation-delay

1)定义: 设置动画的延迟时间

2)语法: animation-delay: time

3)参数说明:

定义动画开始前等待的时间,单位s或ms,默认值是0

### 11.2.5 animation-iteration-count

1)定义: 设置动画的循环次数

2)语法: animation-iteration-count : infinite | number

3)参数说明:

number为数字,其默认值是1

infinite: 无限循环

### 11.2.6 animation-direction

1)定义: 设置动画的方向

2)语法:

animation-direction: normal , reverse , alternate , alternate-reverse

3)参数说明:

Normal:正常方向

reverse :反向

alternate:先正常在反向,并持续交替运行,必须配合循环使用

alternate-reverse:先反向在正常,并持续交替运行, 必须配合循环使用

### 11.2.7 animation-play-state

1)定义: 设置动画是否播放

2)语法: animation-play-state: paused | running

3)参数说明:

paused: 动画暂停

running: 默认值,动画播放

### 11.2.8 animation-fill-mode

1)定义: 设置动画填充(动画外的状态)

2)语法: animation-fill-mode: backwards | both | forwards

3)参数说明:

动画外的状态:

1.动画执行之前的状态,动画延迟时间的状态

2.动画执行之后的状态

backwards : 让元素一开始与 from 状态保持一致

both : 让元素一开始与 from 状态保持一致,结束时候与to状态保持一致

forwards: 结束时候与to状态保持一致

### 11.2.9 animation

1)定义: 设置对象所应用的动画特效

2)语法:

animation : name duration timing-function delay interation-count direction play-state

## 11.3开机动画

原理: 检测项目进度(进度条)

项目进度=图片加载成功的个数/图片的总数

//创建一个数组用来保存项目中的所有图片

var arr =[];

for(var i in imgData){

//合并数组

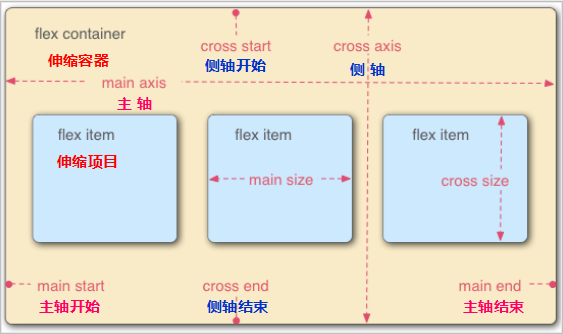
arr = arr.concat(imgData[i]);

}

//创建img对象

//var img = document.createElement('img');  
 var img = new Image();/\*通常使用\*/

# 12.伸缩盒子模型



伸缩盒子模型flex容器必须是父子关系:

父元素: 伸缩容器；

子元素: 伸缩项目；

横向的轴main axis: 主轴；

左边main start: 主轴开始；

右边main end: 主轴结束；

垂直方向的轴cross axis: 侧轴；

上方cross start: 侧轴开始；

下方cross end: 侧轴结束

子元素沿着主轴的方向排列。

伸缩盒子模型可以让块元素在一行排列,不用设置浮动,父元素也没有高度塌陷。

注: 本节属性如没有特殊说明,都是给父元素设置。

## 12.1新老版本对比

### 12.1.1 flex容器

新版本: display: flex;

display: -webkit-flex;

老版本: display: -webkit-box;

/\*css书写顺序:

先老版本,再新版本

display: -webkit-box;

display: -webkit-flex;

display: flex;

\*/

### 12.1.2主轴布局方向

新版本: flex-direction: row; 默认值

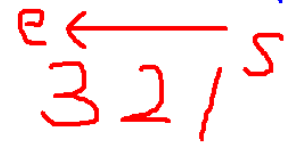
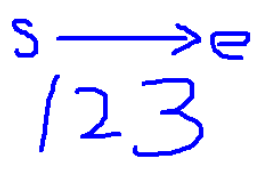
flex-direction: column; 主轴与侧轴发生对调

老版本: -webkit-box-orient: horizontal; 默认值

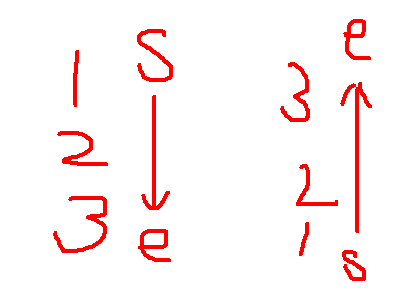
-webkit-box-orient: vertical; 主轴与侧轴发生对调

\*\*\*\*怎么找主轴的位置\*\*\*\*

如果子元素横向排列,



如果子元素纵向排列,



### 12.1.3主轴排列方向

新版本: flex-direction: row-reverse; 主轴上的start与end对调,从右到左

flex-direction: column-reverse; 先把主轴与侧轴对调,再把start与end对调

老版本: -webkit-box-direction: normal; 元素排从左到右,默认方向

-webkit-box-direction: reverse; 元素排从右到左,但是元素整体都在左边

/\*

flex-direction: ;

row 主轴横向从左到右,默认值；

column 主轴纵向从上到下,主轴与侧轴对调；

row-reverse 主轴横向从右到左；

column-reverse 主轴与侧轴对调,然后start与end对调；

\*/

### 12.1.4富裕空间管理(主轴)

新版本:

justify-content: flex-start; 默认值,富裕空间在右侧,子元素在左侧

justify-content: flex-end; 富裕空间在左侧,子元素在右侧

justify-content: center; 富裕空间在两边,子元素在中间

justify-content: space-around; 富裕空间包含伸缩项目

justify-content: space-between; 伸缩项目包含富裕空间

老版本:

-webkit-box-pack: start; 默认值: 富裕空间在右边

-webkit-box-pack: end; 富裕空间在左边

-webkit-box-pack: center; 富裕空间在两边,伸缩项目在中间

-webkit-box-pack: justify; 伸缩项目包含富裕空间

### 12.1.5富裕空间管理(侧轴)

新版本:伸缩项目内没有内容的部分

align-items: flex-start; 富裕空间在下边

align-items: flex-end; 富裕空间在上边

align-items: center; 富裕空间在两边

align-items: baseline; 富裕空间被基线分开

align-items: stretch; 拉伸,默认值,子元素高度与父元素相等

注: align-items控制的是单行侧轴

老版本:

-webkit-box-align: start; 富裕空间在下边

-webkit-box-align: end; 富裕空间在上边

-webkit-box-align: center; 富裕空间在上下两边,富裕空间包含伸缩项目的整体,使伸缩项目整体在中间

### 12.1.6 弹性空间(给子元素设置)

新版本:

flex-grow: 1; 将富裕空间按比例分配到项目上

老版本:

-webkit-box-flex: 1; 将富裕空间按比例分配到项目上

加上弹性空间后盒子的宽度

1.元素原有的宽度(内容撑开的)

2.富裕空间分出来的部分(等分)

.box1{  
 background: deeppink;   
 flex-grow: 1; /\*box1占1/6\*/  
 }  
 .box2{  
 background: yellowgreen;  
 flex-grow: 2; /\*box2占2/6\*/  
 }  
 .box3{  
 background: greenyellow;  
 flex-grow: 3; /\*box3占3/6\*/  
 }

## 12.2 新版本特有属性

### 12.2.1 元素实现换行

flex-wrap: nowrap; --- 默认值,父元素宽度不够,子元素自身宽度会被压缩

flex-wrap: wrap; --- 父元素宽度不够,子元素会进行换行

flex-wrap: wrap-reverse; --- 子元素换行的同时,侧轴的start与end发生对调

注: wrap换行后会生成3个侧轴,一个主侧轴,两个单行侧轴。

### 12.2.2 控制整体侧轴

align-content: flex-start; --- 项目整体进行打包,放在整体侧轴的start处

align-content: flex-end; --- 项目整体进行打包,放在整体侧轴的end处

align-content: center; --- 项目整体进行打包,放在整体侧轴的center处

注意: align-items 与 align-content 发生冲突时,看元素是否换行

如果没有换行align-items 生效

如果有换行align-content 生效

align-content 生效条件:

1)父元素必须要有换行语句flex-wrap: wrap;

2)父元素要有足够的高(因为需要整体打包才能看见效果,所以需要高度)

### 12.2.3 控制主轴和侧轴的位置及方向

flex-flow 是 flex-wrap 与 flex-direction 的缩写

flex-flow:wrap-reverse column-reverse;

与flex-wrap: wrap-reverse;flex-direction: column-reverse;实现效果一样

### 12.2.4 项目的排列顺序(子元素设置)

order: 1;

order排序,把排序元素先单独拿出来,让剩余元素先正常排列,排完之后,order元素在其后边再进行顺序排列

order是沿着主轴方向进行排序的

### 12.2.5 项目自身侧轴的富裕空间(子元素设置)

每一个项目控制自身的侧轴

align-self: flex-start;

align-self: flex-end;

align-self: center;

### 12.2.6 收缩率(flex项目)

当伸缩项目设置宽度,比容器宽度还要大时,元素并没有发生溢出的情况,那此时就出现一个东西叫 收缩因子(收缩率)

flex-grow:拉伸因子 0 ---不拉伸

flex-shrink:收缩因子 默认值: 1 --- 收缩

收缩率越大,元素宽度越小

### 12.2.7 基准值(flex项目)

子元素的基准值

flex-basis: 1;

flex-basis: 0;/\*将原有的宽度清零\*/

### 12.2.8 简写属性

语法: flex: flex-basis flex-grow flex-shrink的简写,

默认值:

flex-basis: auto;

flex-grow: 0;

flex-shrink: 1;

flex: 1;

flex-basis: 0;

flex-grow: 1;

flex-shrink: 1;

## 12.3等分布局

### 12.3.1第一种

父元素 display: flex;

子元素 flex-basis: 0;

flex-grow: 1;

### 12.3.2第二种(老版本)

父元素 display: -webkit-box;

子元素 width: 0;

-webkit-box-flex: 1;

### 12.3.3第三种

父元素 display: flex;

flex: 1;

/\*

元素换行到底设置宽度还是高度？？

column 主轴和侧轴对调过后,设置高度；

如果主轴在水平方向上,元素设置宽度来实现换行；

如果主轴在垂直方向上,元素设置高度来实现换行；

\*\*元素实现换行效果,但是当前元素并没有设置宽度,让元素实现换轴\*\*

\*/

# \*\*\*\*\*\*\*jQuery\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 1. jQuery核心函数与对象

$('button').click(function () {  
 console.log('测试');  
 })

$是window上的一个属性,可以当做函数使用$()；

$()的返回值是一个jQuery对象,所以也可以当做对象使用；

$().click是jQuery对象上的一个属性,它的值是一个函数。

.click()括号内的函数是实参。

\* JQuery的优点:   
 使用简单方便 – 宗旨: Write Less, Do More。写得少,做得多。吃得少干得多。  
 JQuery能做的js能做,js能做的JQuery也能做,但是jquery简单、易懂、兼容性好(解决了几乎所有的兼容性问题)。  
 链式编程  
 隐式迭代(遍历):可以不用循环遍历对dom操作  
 读写二合一  
 在操作dom如果没有硬性规定,尽量不用原生js

jQuery对象是一个伪数组,可以且只能使用索引下标和length,它的本质对象

## 1.1 作为一般函数调用: $(param)

### 1.1.1 参数为函数

$(function () {} )

dom加载结束后执行此函数,类似window.onload

### 1.1.2 参数为选择器字符串

id $('#id')

class $('.class')

标签 $('div')

添加完后变成jQuery对象。

### 1.1.3 参数为dom对象

this 在所有的框架中,只能是dom对象。

### 1.1.4参数为html标签字符串

$('<input type="text" name="msg3"/><br/>').appendTo('div')

## 1.2作为对象使用: $.xxx()

### 1.2.1 $.each() : 遍历

/\*遍历数组\*/

var arr = [1,2,3,4];  
 $.each(arr,function (index,item) {  
 console.log(index,item);  
 })  
 /\*遍历对象\*/  
 var obj = {  
 name : 'tom',  
 age : 18,  
 height : 200  
 }  
 $.each(obj,function (n,value) {  
 console.log(n,value);  
 })  
 /\*遍历jQuery对象\*/

var $btn = $('button')

$btn.each(function (i,item) {  
 console.log(i,item);  
 //this 每一个dom对象  
 console.log(this.innerText);  
 console.log($(this).text());  
 })

### 1.2.2 $.trim() : 去除两端的空格

//需求3. 去掉" my atguigu "两端的空格  
 var str = " my atguigu ";  
 console.log(str);  
 console.log($.trim(str));

### 1.2.3size()/length: 包含的DOM元素个数

//需求1. 统计一共有多少个按钮

var $btn = $('button')  
 console.log($btn.length);  
 console.log($btn.size());

### 1.2.4[index]/get(index): 得到对应位置的DOM元素

//需求2. 取出第2个button的文本  
 console.log($('button')[1].innerText);  
 console.log($($('button')[1]).text());  
 console.log($('button').get(1).innerText);  
 console.log($($('button').get(1)).text());

### 1.2.5index(): 得到在所在兄弟元素中的下标

//需求4. 输出'测试三'按钮是所有按钮中的第几个  
 index(): 得到在所在兄弟元素中的下标

console.log($('#btn3').index());  
 console.log($btn.index());

## 1.3基本行为

<button id="box">测试一</button>  
 <button>测试二</button>  
 <button id="btn3">测试三</button>  
 <button>测试四</button>

### 1.3.1 size()/length: 包含的DOM元素个数

//需求1. 统计一共有多少个按钮  
 var $btn = $('button')  
 console.log($btn.length);  
 console.log($btn.size());

### 1.3.2 [index]/get(index): 得到对应位置的DOM元素

//需求2. 取出第2个button的文本  
 console.log($('button')[1].innerText);  
 console.log($($('button')[1]).text());  
 console.log($('button').get(1).innerText);  
 console.log($($('button').get(1)).text());

### 1.3.3 each(): 遍历包含的所有DOM元素

//需求3. 输出所有button标签的文本  
 $btn.each(function (i,item) { /\*i是索引下标,item是dom对象\*/  
 console.log(i,item);  
 //this 每一个dom对象  
 console.log(this.innerText);  
 console.log($(this).text());  
 })

### 1.3.4 index(): 得到在所在兄弟元素中的下标

//需求4. 输出'测试三'按钮是所有按钮中的第几个  
 index(): 得到在所在兄弟元素中的下标  
 console.log($('#btn3').index());  
 console.log($btn.index());

## 1.4对象类型转换

### 1.4.1把dom对象转换成jQuery对象

$(dom对象) 不用加引号。

### 1.4.2把jQuery对象转换成dom对象

直接加一个索引下标: jQuery对象[index];

加 .get(): jQuery对象.get(index)

# 2. jQuery选择器

## 2.1基本选择器

- #id : id选择器  
 - element : 元素选择器  
 - .class : 属性选择器  
 - \* : 任意标签  
 - selector1,selector2,selectorN : 取多个选择器的并集(组合选择器)  
 - selector1selector2selectorN : 取多个选择器的交集(相交选择器)

1. 选择id为div1的元素  
 $('#div1').css('color','red')  
 2. 选择所有的div元素  
 $('div').css('color','red')  
 3. 选择所有class属性为box的元素  
 $('.box').css('color','red')  
 4. 选择所有的div和span元素  
 $('div,span').css('color','green')  
 5. 选择所有class属性为box的div元素

$('div.box').css('color','green')

## 2.2层次选择器

- ancestor descendant  
 在给定的祖先元素下匹配所有的后代元素  
 - parent>child  
 在给定的父元素下匹配所有的子元素  
 - prev+next  
 匹配所有紧接在 prev 元素后的 next 元素  
 - prev~siblings  
 匹配 prev 元素之后的所有 siblings 元素

1. 选中ul下所有的span

$('ul span').css('background','pink')  
2. 选中ul下所有的子元素span  
 $('ul>span').css('background','pink')  
3. 选中class为box的下一个li  
 $('.box+li').css('background','green')  
4. 选中ul下的class为box的元素后面的所有兄弟元素  
 $('.box~li,.box~span').css('background','green')  
 $('.box~\*').css('background', 'pink')

## 2.3过滤选择器

基本 :first :last :not(selector) :even :odd :eq(index) :gt(index) :lt(index)

内容 :contains(text)

可见性 :hidden :visible

属性 [[attribute]](http://www.codesocang.com/jq/attributeHas.html) [[attribute=value]](http://www.codesocang.com/jq/attributeEquals.html) [[attribute!=value]](http://www.codesocang.com/jq/attributeNotEqual.html) [[attribute^=value]](http://www.codesocang.com/jq/attributeStartsWith.html) [[attribute$=value]](http://www.codesocang.com/jq/attributeEndsWith.html) [[attribute\*=value]](http://www.codesocang.com/jq/attributeContains.html)

1. 选择第一个div  
 $('div:first').css('background','pink')  
 2. 选择最后一个class为box的元素  
 $('.box:last').css('background','pink')  
 3. 选择所有class属性不为box的div  
 $('div:not(.box)').css('background','pink')  
 4. 选择第二个和第三个li元素  
 $('li:eq(1),li:eq(2)').css('background','pink')  
 $('li:gt(0):lt(2)').css('background','green')  
 $('li:lt(3):gt(0)').css('background','blue')

注: :gt和:lt连续使用时,前面的选择器选择之后,剩下的元素重新编号,后面的选择器在进行选取,建议先使用:lt,再使用:gt。

gt 大于  
 $('li:gt(1)').css('background','pink')  
 lt 小于  
 $('li:lt(1)').css('background','pink')  
 5. 选择内容为BBBBB的li  
 $('li:contains("BBBBB")').css('background','pink')  
 6. 选择隐藏的li  
 console.log($('li:hidden').text());  
 7. 选择有title属性的li元素  
 $('li[title]').css('background','pink')  
 8. 选择所有属性title为hello的li元素  
 $('li[title="hello"]').css('background', 'pink')

# 3. $工具方法

## 3.1. $.each(): 遍历数组或对象中的数据

/\*遍历数组\*/

var arr = [1,2,3,4];  
 $.each(arr,function (index,item) {  
 console.log(index,item);  
 })  
 /\*遍历对象\*/  
 var obj = {  
 name : 'tom',  
 age : 18,  
 height : 200  
 }  
 $.each(obj,function (n,value) {  
 console.log(n,value);  
 })  
 /\*遍历jQuery对象\*/

var $btn = $('button')

$btn.each(function (i,item) {  
 console.log(i,item);  
 //this 每一个dom对象  
 console.log(this.innerText);  
 console.log($(this).text());  
 })

## 3.2. $.trim(): 去除字符串两边的空格

// 去掉" my atguigu "两端的空格  
 var str = " my atguigu ";  
 console.log(str);  
 console.log($.trim(str));

## 3.3. $.type(obj): 得到数据的类型

var str = '1111'  
 console.log($.type(str));  
 console.log(typeof str);  
 var obj = {name:'tom'}  
 console.log($.type(obj));  
 //jQuery是一个伪数组: 假的数组,只能索引值和length , 本质是对象  
 var $box = $('#box')  
 console.log($.type($box));

## 3.4. $.isArray(obj): 判断是否是数组

var arr = [1,2];  
 console.log($.isArray(arr)); //true  
 console.log($.isArray($box)); //false

## 3.5. $.isFunction(obj): 判断是否是函数

console.log($.isFunction($));  
 console.log($.isFunction(jQuery));  
 console.log($ === jQuery);

/\*window.jQuery = window.$ = function() {}\*/

# 4. jQuery属性

## 4.1 .attr() .removeAttr()

1. 读取一个div的title属性  
 读写二合一: 一个参数是读,两个参数是写   
 console.log($('#div1').attr('title'));  
 console.log($('#div1').attr('class'));  
 console.log($('#div1').attr('id'));  
 2. 给所有的div设置name属性(value为atguigu)  
 $('div').attr('name','atguigu')  
 3. 移除所有div的title属性  
 $('div').removeAttr('title')  
 \*\*添加多个属性名值对  
 $('div').attr({  
 name1:'at',  
 name2:'gui',  
 name3:'gu'  
 })  
 \*链式编程  
 $('div')  
 .attr('name1','at')  
 .attr('name2','gui')  
 .attr('name3','gu')

注: attr()添加的属性会发生覆盖操作

## 4.2 .prop() .removeprop()

\*\*\*布尔值属性 不能使用attr, 必须使用prop()  
 $(':checkbox').prop('checked',true)

//11. 点击'全选'按钮实现全选  
 $('button:first').click(function () {})  
 //12. 点击'全不选'按钮实现全不选  
 $('button:last').click(function () {  
 $(':checkbox').prop('checked',false)  
 })

## 4.3 .addClass() .removeClass() .toggleClass()

//4. 给所有的div设置class='guiguClass'  
 $('div').attr('class','guiguClass') //覆盖操作  
 $('div').addClass('guiguClass')

//5. 给所有的div添加class='abc'  
 $('div').addClass('abc')  
 //6. 移除所有div的guiguClass  
 $('div').removeClass('guiguClass')  
 //toggleClass 如果有,结果删除;如果没有,结果是添加  
 $('div').toggleClass('box')

## 4.4 .html() .text() .val()

\* 读取和设置内容:  
 \* 设置:  
 \* 目标.text('内容') 和 innerText用法一样  
 \* 目标.html('内容') 和 innerHTML用法一样  
 \* 读取:  
 \* 目标.text()  
 \* 目标.html()

//7. 得到最后一个li的标签体文本  
 var $li = $('ul>li:last');  
 console.log($li.html());  
 console.log($li[0].innerHTML);  
 console.log($li.text());  
 console.log($li[0].innerText);  
 //8. 设置第一个li的标签体为"<h1>mmmmmmmmm</h1>"  
 var $li = $('ul>li:first');  
 $li.html('<h1>mmmmmmmmm</h1>')  
 $li.text('<h1>mmmmmmmmm</h1>')  
 //9. 得到输入框中的value值  
 console.log($(':text').val());  
 //10. 将输入框的值设置为atguigu  
 $(':text').val('atguigu')

# 5. css模块

## 5.1 css使用

/\*设置单个属性\*/

.css( '属性名' , '属性值')

/\*设置多个属性\*/

.css( {

属性名: '属性值'

属性名: '属性值'

属性名: '属性值'

} )

.css('属性名' , '属性值').css('属性名' , '属性值').css('属性名' , '属性值')

## 5.2 offset和position.html

获取/设置标签的位置数据  
 offset(): 相对页面左上角的坐标

position(): 相对于父元素左上角的坐标,只能读取,设置相对于父元素左上角的坐标使用 .css() 。

注意: 写的操作要放到对象中,而且不能带单位。  
 需求:  
 1. 点击 btn1  
 打印 div1 相对于页面左上角的位置  
 打印 div2 相对于页面左上角的位置  
 打印 div1 相对于父元素左上角的位置  
 打印 div2 相对于父元素左上角的位置  
 2. 点击 btn2  
 设置 div2 相对于页面的左上角的位置  
 //1. 点击 btn1  
 $('#btn1').click(function () {  
 //打印 div1 相对于页面左上角的位置  
 console.log($('.div1').offset()); //{top: 20, left: 10}返回值是一个对象  
 console.log($('.div1').offset().top);  
 console.log($('.div1').offset().left);  
 //打印 div2 相对于页面左上角的位置  
 console.log($('.div2').offset());  
 console.log($('.div2').offset().top);  
 console.log($('.div2').offset().left);  
 //打印 div2 相对于父元素左上角的位置  
 console.log($('.div2').position());  
 console.log($('.div2').position().top);  
 console.log($('.div2').position().left);  
 })  
 //2. 点击 btn2  
 $('#btn2').click(function () {  
 //设置 div2 相对于页面的左上角的位置  
 $('.div2').offset({  
 top : 10,  
 left : 20  
 })

注: 这里没有单位

//设置 div2 相对于父元素的左上角的位置  
 $('.div2').css({  
 top : '10px',  
 left : '20px'  
 })

注: 由于.position()只能读取,设置相对于父元素左上角的坐标使用.css()  
 })

## 5.3 元素滚动.html

1. 读取/设置滚动条的Y坐标  
 读取: scrollTop()  
 设置: scrollTop(60)

2. 读取页面滚动条的Y坐标(兼容chrome和IE)

$(document.body).scrollTop()+$(document.documentElement).scrollTop()

兼容chrome、firefox、IE高版本:

$(document.documentElement).scrollTop() === $('html').scrollTop()

只有IE低版本认识(6.7.8),其他高版本浏览器解析效果是0:   
 $(document.body).scrollTop() === $('body').scrollTop()

3. 设置滚动条到指定位置(兼容chrome和IE)

$('body,html').scrollTop(60);

## 5.4 元素尺寸

5.4.1. 内容尺寸(content-box)

.height() height  
 .width() width

console.log($('div').height());  
 console.log($('div').width());

5.4.2. 内部尺寸(padding-box)

.innerHeight() height+padding  
 .innerWidth() width+padding

\*\*\*\*div.clientWidth  
 console.log($('div').innerWidth());  
 console.log($('div').innerHeight());  
 console.log($('div')[0].clientWidth);  
 console.log($('div')[0].clientHeight);

5.4.3. 外部尺寸(border-box)

.outerHeight(false/true) height+padding+border 如果是true, 加上margin  
 .outerWidth(false/true) width+padding+border 如果是true, 加上margin

\*\*\*div.offsetWidth  
 console.log($('div').outerWidth(false)); //默认是false  
 console.log($('div').outerHeight());  
 console.log($('div')[0].offsetWidth);  
 console.log($('div')[0].offsetHeight);  
 console.log($('div').outerWidth(true)); //height+padding+border+margin

console.log($('div').outerHeight(true)); //width+padding+border+margin

# 6. jQuery对象的过滤

## 6.1 筛选-过滤

在jQuery对象中的元素对象数组中过滤出一部分元素来  
 1. first() 第一个  
 2. last() 最后一个  
 3. eq(index|-index) 第n个

4. filter(selector) 筛选出与指定表达式匹配的元素集合  
 5. not(selector) 不包括

6. has(selector) 保留包含特定后代的元素,去掉那些不含有指定后代的元素。

//1. ul下li标签第一个  
 var $li = $('ul>li');  
 $li.first().css('background','deeppink')  
 //2. ul下li标签的最后一个  
 $li.last().css('background','green')  
 //3. ul下li标签的第二个  
 $li.eq(1).css('background','deeppink')  
 //4. ul下li标签中title属性为hello的  
 $li.filter('[title="hello"]').css('background','green')  
 //ul下li标签中title属性  
 $li.filter('[title]').css('background','green')  
 //5. ul下li标签中title属性不为hello的

$li.filter('[title!="hello"]').css('background','green')  
 $li.not('[title="hello"]').css('background','pink')

//ul下li标签中title属性,并且title属性不为hello的  
 $li.filter('[title]').filter('[title!="hello"]').css('background','green');  
 $li.filter('[title]').not('[title="hello"]').css('background','pink')  
 $li.filter('[title][title!="hello"]').css('background','green');  
 //6. ul下li标签中有span子标签的  
 $li.has('span').css('background','blue');

## 6.2 筛选\_查找孩子-父母-兄弟标签.html

在已经匹配出的元素集合中根据选择器查找孩子/父母/兄弟标签  
 1. children(): 子标签中找  
 2. find() : 后代标签中找  
 3. parent() : 父标签  
 4. prevAll() : 前面所有的兄弟标签  
 5. nextAll() : 后面所有的兄弟标签  
 6. siblings() : 前后所有的兄弟标签

//1. ul标签的第2个span子标签  
 var $ul = $('ul');  
 console.log($ul.children('span')[1]);//dom对象  
 console.log($ul.children('span').get(1));//dom对象  
 console.log($ul.children('span').eq(1)); //jQuery对象  
 $ul.children('span').eq(1).css('background','blue')  
 //2. ul标签的第2个span后代标签  
 $ul.find('span')[1].style.background = 'green'  
 $ul.find('span').eq(1).css('background','blue')  
 //3. ul标签的父标签  
 $ul.parent().css('background','blue')  
 //4. id为cc的li标签的前面的所有li标签  
 $('#cc').prev('li').css('background','blue')  
 $('#cc').prevAll('li,span').css('background','blue')  
 $('#cc').prevAll('\*').css('background','blue')  
 $('#cc').prevAll().css('background','blue')  
 //5. id为cc的li标签的所有兄弟li标签  
 $('#cc').siblings().css('background','hotpink')  
 $('#cc').next().css('background','hotpink')  
 $('#cc').nextAll().css('background','pink')

## 6.3案例

### 6.3.1 小圆点切换(排它)

<ol>  
 <li class="one">1</li>  
 <li class="one">2</li>  
 <li class="one">3</li>  
 <li class="one">4</li>  
 <li class="one">5</li>  
 </ol>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

var $li = $('.one');  
 $li.click(function () {  
 //第一种  
 $(this).css('background','green');  
 $(this).siblings().css('background','#ccc');  
 //第二种  
 $(this)

.css('background','pink')

.siblings().css('background','#ccc')  
//第三种  
$(this)

.attr('class','one current')

.siblings().attr('class','one')  
//第四种  
$(this)

.addClass('current')

.siblings().removeClass('current')

});

# 7.文档处理

## 7.1文档的增删改

### 7.1.1 添加/替换元素

\* append(content)  
 目标.append('元素') 将元素追加到目标内部的最后面  
 目标.appendTo('元素')将目标追加到元素内部的最后面

父元素的最后面.append(子元素)

子元素.appendto(父元素的最后面)

\* prepend(content)  
 目标.prepend('元素') 将元素追加到目标内部的最前面  
 目标.prependTo('元素')将目标追加到元素内部的最前面

父元素的最前面.prepend(子元素)

子元素.prependto(父元素的最前面)

\* before(content)  
 目标.before('元素') 将元素追加到目标的前面

\* 目标.insertBefore('元素') 将目标追加到元素的前面  
\* after(content)  
 目标.after('元素') 将元素追加到目标的后面

\* 目标.insertAfter('元素') 将目标追加到元素的后面  
\* replaceWith(content)  
 用指定内容替换所有匹配的标签删除节点

$li.replaceWith('<p>p</p>')  
 ('<p>p</p>').replaceAll($li)

### 7.1.2 删除元素

\* empty() 删除所有匹配元素的子元素  
 \*目标.empty() 清空标签的内容,斩草留根

\* remove() 删除所有匹配的元素

\*目标.remove() 连同标签一起删除,斩草除根

## 7.2 jQuery创建元素的三种方法

### 7.2.1第一种

$( 'ul' ).html( '<li>233333</li>' ) //会发生覆盖

### 7.2.2第二种

var $li = $( '<li>233333</li>' )

$('ul').append($li)

$li.appendto($('ul'))

$('ul').prepend($li)

$li.prependto($('ul'))

### 7.2.3第三种

$('ul').append( '<li>233333</li>' )

$('ul').prepend( '<li>233333</li>' )

$('<li>233333</li>').appendto( $('ul') )

$('<li>233333</li>').prependto( $('ul') )

# 8.事件处理

## 8.1 事件绑定

1. $('').click( function() {} ) //不是所有事件都能使用

2. $('').on( 'click' , function() {} )

\*\*\*\*给.inner绑定鼠标移入移出事件(三种)

1. $('.inner').mouseover( function() {} )

$('.inner').mouseout( function() {} )

2. $('.inner').on('mouseenter',function () {})  
 $('.inner').on('mouseleave',function () {})

3. $('.inner').hover(function () {//鼠标移入} , function () { //鼠标移出})

//5. 点击btn3得到事件坐标  
 $('#btn3').click(function (event) {  
 //相对于视口的左上角  
 console.log(event.clientX, event.clientY);  
 //相对于事件元素左上角  
 console.log(event.offsetX, event.offsetY );  
 //相对于页面的左上角  
 console.log(event.pageX, event.pageY);  
 })

## 8.2 事件解除

//点击btn1解除.inner上的所有事件监听 ---解绑  
 $('#btn1').on('click',function () {  
 $('.inner').off()  
 });  
 //点击btn2解除.inner上的mouseover事件  
 $('#btn2').on('click',function () {  
 $('.inner').off('mouseover')  
 });

## 8.3 事件委派

1. 事件委托:  
 \* 将多个子元素(li)的事件监听委托给父辈元素(ul)处理  
 \* 监听回调是加在了父辈元素上  
 \* 当操作任何一个子元素(li)时, 事件会冒泡到父辈元素(ul)  
 \* 父辈元素不会直接处理事件, 而是根据event.target得到发生事件的子元素(li), 通过这个子元素调用事件回调函数  
 2. 事件委托的2方:  
 \* 委托方: 业主 li  
 \* 被委托方: 中介 ul  
 3. 使用事件委托的好处  
 \* 添加新的子元素, 自动有事件响应处理  
 \* 减少事件监听的数量: n==>1  
 4. jQuery的事件委托API  
 \* 设置事件委托: $(parentSelector).delegate(childrenSelector, eventName, callback)  
 \* 移除事件委托: $(parentSelector).undelegate(eventName)

### 8.3.1 原生

//1.绑定事件需要给祖先元素  
 var list = document.getElementById('list');  
 list.onclick = function (e) {  
 //2.找触发事件的子元素 event.target  
 if(e.target.nodeName == 'LI'){  
 e.target.style.background = 'red'  
 }  
 };

### 8.3.2 jQuery

第一种形式:

//第一种形式

//父元素.delegate(子元素,事件名称,回调函数)  
 $ul.delegate('li','click',function () {

//找触发事件的子元素 this  
 console.log(this);  
 $(this).css('background','red')  
})

//删除ul上的事件委托的监听器  
 $('#btn2').click(function () {  
 $ul.undelegate('click')  
 })

第二种形式:

//第二种形式 --- 用的多  
 //父元素.on(事件名称,子元素,回调函数)  
 $ul.on('click','li',function () {  
 $(this).css('background','blue')  
 })

# 效果

## 9.1 淡入淡出

淡入淡出: 不断改变元素的透明度来实现的  
1. fadeIn(): 带动画的显示  
2. fadeOut(): 带动画隐藏  
3. fadeToggle(): 带动画切换显示/隐藏

//1. 点击btn1, 慢慢淡出  
 $('#btn1').click(function () {  
 $('.div1').fadeOut('slow');  
 $('.div1').fadeOut(2000,function () {  
 console.log('淡出');  
 });  
 })  
 //2. 点击btn2, 慢慢淡入  
 $('#btn2').click(function () {  
 $('.div1').fadeIn(2000,function () {  
 console.log('淡入');  
 })  
 })  
 //3. 点击btn3, 淡出/淡入切换,动画结束时提示“动画结束了”  
 $('#btn3').click(function () {  
 $('.div1').fadeToggle(function () {  
 console.log('切换');  
 })  
 })  
 //4.透明度变换  
 $('#btn4').click(function () {  
 $('.div1').fadeTo(2000,0.2,function () {  
 console.log('22222');  
 })  
 })

## 9.2滑动

滑动动画  
 1. slideDown(): 带动画的展开  
 2. slideUp(): 带动画的收缩  
 3. slideToggle(): 带动画的切换展开/收缩

\* 滑动动画(卷帘门动画) img必须加宽高,否则会变成对角线动画

//1. 点击btn1, 向上滑动  
 $('#btn1').click(function () {  
 $('.div1').slideUp(3000,function () {  
 console.log('2222');  
 });  
 })  
 //2. 点击btn2, 向下滑动  
 $('#btn2').click(function () {  
 $('.div1').slideDown(3000,function () {  
 console.log('2222');  
 });  
 })  
 //3. 点击btn3, 向上/向下切换  
 $('#btn3').click(function () {  
 $('.div1').slideToggle(1000,function () {  
 console.log('2222');  
 });  
 })

## 9.3显示与隐藏

显示隐藏,默认没有动画  
 1. show(): (不)带动画的显示  
 2. hide(): (不)带动画的隐藏  
 3. toggle(): (不)带动画的切换显示/隐藏

//1. 点击btn1, 立即显示  
 $('#btn1').click(function () {  
 $('.div1').show();  
 })  
 //2. 点击btn2, 慢慢显示  
 $('#btn2').click(function () {  
 $('.div1').show(5000,function () {  
 console.log('77777');  
 });  
 })  
 //3. 点击btn3, 慢慢隐藏  
 $('#btn3').click(function () {  
 $('.div1').hide(5000,function () {  
 console.log('77777');  
 });  
 })  
 //4. 点击btn4, 切换显示/隐藏  
 $('#btn4').click(function () {  
 $('.div1').toggle();  
 $('.div1').toggle(3000);  
 })

## 9.4自定义动画

jQuery动画本质 : 在指定时间内不断改变元素样式值来实现的  
 1. animate(): 自定义动画效果的动画  
 2. stop(): 停止动画

//1.宽度和高度扩大200px  
 $('.div1').animate({  
 width : '200px',  
 height : '200px'  
 },2000,'linear',function () {})

//2.先宽度扩大200px后高度扩大200px(采用链式调用)  
 $('.div1')  
 .animate({width : '200px'},1000)//.stop()  
 .delay(3000)  
 .animate({height : '200px'},1000)//.stop()

//3.移动到(100,20)处  
 $('.div1').animate({  
 left : '100px',  
 top : '20px'  
 },5000)

//4.移动距离为(100,50)有无单位都行  
 $('.div1').animate({  
 left: '+=100',  
 top: '+=50'  
 })

//5.停止动画(停止当前正在进行的动画)  
 $('#btn4').click(function () {  
 $('.div1').stop();  
 });

# 10.多库共存

问题 : 如果有2个库都有$, 就存在冲突  
 解决 : jQuery库可以释放$的使用权, 让另一个库可以正常使用, 此时jQuery库只能使用jQuery了  
 API : jQuery.noConflict()

//jQuery库可以释放$的使用权  
 jQuery.noConflict();  
 //函数调用  
 $(); //my.js  
 //调用jQuery  
 console.log(jQuery('#box'));

# 11.页面加载

区别: window.onload与 $(document).ready()  
 \* window.onload  
 \* 包括页面的图片加载完后才会回调(晚)  
 \* 只能有一个监听回调  
 \* $(document).ready()  
 \* 等同于: $(function(){})  
 \* 页面加载完(dom)就回调(早)  
 \* 可以有多个监听回调

# 12.less使用

## 12.1 less使用注意点

1.less中创建变量

@变量名: 变量值；

@zero: 0;

@yg: yellowgreen;

2. less使用中子元素通常嵌套在父元素里,父元素的样式要和子元素的选择器对齐,基本嵌套,选择器会变成后代选择器。

#wrap{

.box{} =====> #wrap .box{}

}

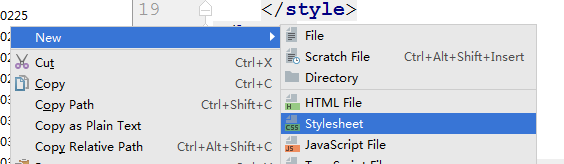
//3. <style type="text/less"> style标签的类型需要改成less

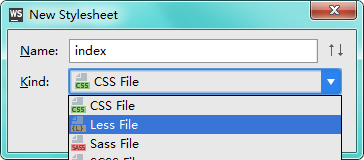
//4. 根据官网我们需要一个less编译的 less.js 文件,并在最下方引入,因为需要读取页面中所有less相关的文件,才可以进行编译

## 12.2 less使用步骤

1. 创建一个空的文件夹(css或style)

2. 在空文件夹里创建一个less文件





3. 在less文件里写css样式

4. 把css文件夹拖到考拉里,会编译出一个css文件

5. 在html文件里面引入css文件

## 12.3 less中的变量

使用@来申明一个变量: @pink: pink;  
 1.作为普通属性值来使用: 直接使用@pink  
 2.作为选择器和属性名:

@w: width; (属性名)

使用: @{w}

@wrap: wrap; (选择器)

使用: #@{wrap}

#@{selector的值}的选择器形式, @{selector的值}属性名  
3.作为URL: @url  
4.变量的延迟加载

1). 先找子元素,如果子元素中有两个或者多个相同的变量默认拿最后一个

2). 再找父元素,把子元素的东西排除掉,如果父元素中有两个或者多个相同的变量默认拿最后一个,如果没有去上一层找

@var: 0;  
 .class {  
 @var: 1;  
 .brass {  
 @var: 2;  
 three: @var;  
 @var: 3;  
 }  
 one: @var;  
 }

## 12.4 less中的嵌套

1.基本嵌套规则  
 2.&的使用: 微元素、伪类、css结构类、:: .(交集选择器) 通常在前面加上一个&,

&代表前边的所有父级元素

## 12.5 less加减乘除的运算

width: 200px + 200px;

width: 300px - 100px;

width: 200px \* 2;

width: 400px / 2;

## 12.6 less封装混合mixin

1.语法: .名字() {}

例:

//封装混合

.juzhong(@w,@h:250px,@bg){  
 position: absolute;  
 top: 0;  
 left: 0;  
 right: 0;  
 bottom: 0;  
 margin: auto;  
 width: @w;  
 height: @h;  
 background: @bg;  
 }

//调用混合

.juzhong(@w:500px,@h:500px,@bg:grey);

.juzhong(@w:300px,@bg:greenyellow);

2. 外部定义

引入外部less文件:

@import "mixins/mixin.less"

# \*\*\*\*\*\*\*h5\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 1.语义化标签

## 1.1 标签

<header>头部</header> : 块元素,通常用来表示页面的头部,还可以用来表示某个区域的头部,没有个数的限制。

<footer>底部</footer> : 块元素,通常用来表示页面的底部,还可以用来表示某个区域的底部,没有个数的限制。

<hgroup></hgroup> : 标题分组

<nav>导航</nav> : 块元素,通常只有一个

如果导航是静止的,nav标签里面直接包裹a标签；

如果导航可以发生移动(整个一起移动),要使用nav>ul>li>a的形式；

<section></section> : 表达页面中的某一个区域(轮播图区域、内容区域等),功能类似div,块元素

<aside></aside> : 侧边栏,块元素

<article></article> : 通常表示侧边栏对应的区域,块元素,一般和侧边栏一起使用

# 2.表单新属性

2.1 placeholder: 输入框提示文本: ="我叫XXX"

input直接设置color指的是用户输入的颜色；

设置placeholder提示文本的颜色: input::-webkit-input-placeholder{ color: red; }

2.2 autocomplete: 是否保存用户输入值,默认是on；关闭设置:"off"

2.3 autofocus: 自动获取焦点

2.4 required: 验证不能为空值

2.5 获取和失去焦点js方法

var input = document.getElementById('input');

input.focus(); //获取焦点

input.blur(); //失去焦点

# 3.音视频标签

## 3.1音频

标签: <audio><audio>

属性介绍:

src :引入音频路径

controls="controls" 控制显示还是隐藏用户控制界面

loop="loop" 循环播放

autoplay="autoplay" 自动播放

muted = "muted" : 是否静音

preload="none" -- 不加载多媒体文件

preload="metadata" -- 加载基本的播放信息

preload="auto" -- 预加载一部分多媒体资源

## 3.2视频

标签: <video><video>

属性介绍:

src :引入视频路径

controls="controls" 控制显示还是隐藏用户控制界面

autoplay="autoplay" 自动播放

loop = "loop": 循环播放

muted = "muted" : 是否静音 //只有加上muted后autoplay才会生效

preload: 该属性旨在告诉浏览器作者认为达到最佳的用户体验的方式是什么

none: 提示作者认为用户不需要查看该视频,服务器也想要最小化访问流 量;换句话说就是提示浏览器该视频不需要缓存。  
 metadata: 提示尽管作者认为用户不需要查看该视频,不过抓取元数据(比如: 长度)还是很合理的。  
 auto: 用户需要这个视频优先加载;换句话说就是提示:如果需要的话,可以下载 整个视频,即使用户并不一定会用它。

空字符串: 也就代指 auto 值。

width: 宽

height: 高

poster : 引入一张图片,视频封面

注: width height poster是视频标签独有的,且视频的宽高比不会变。

## 3.3标签文本显示问题

video audio canvas标签:

1.如果浏览器支持该标签,文本不可见

2.如果浏览器不支持该标签,文本可见

## 3.4兼容性问题

source: 添加音视频资源的路径,可以添加所有格式的资源,浏览器支持哪个格式就使用哪个格式,找其中最优的资源。

<audio>

<source src="resource/audio/OUTPUT.aac" type='audio/aac; codecs="aac"'/>  
 <source src="resource/audio/OUTPUT.mp3" type='audio/mpeg'/>  
 <source src="resource/audio/OUTPUT.ogg" type='audio/ogg; codecs="vorbis"'/>

</audio>

<video>

<source src="resource/video/OUTPUT.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>  
 <source src="resource/video/OUTPUT.ogv" type='video/ogg; codecs="theora, vorbis"'>  
 <source src="resource/video/OUTPUT.webm" type='video/webm; codecs="vp8, vorbis"'>

</video>

对于IE8:

<audio>

您的浏览器不支持该标签,1.建议浏览器升级(更换浏览器)2. <a href="resource/audio/OUTPUT.mp3">手动下载</a>

</audio>

<video>

您的浏览器不支持该标签,1.建议浏览器升级(更换浏览器)2. <a href="resource/video/OUTPUT.mp4">手动下载</a>

</video>

## 3.5 音视频js对象

### 3.5.1常用js事件

**onloadedmetadata 当音视频元数据加载完毕时触发**

**ontimeupdate 播放过程中实时触发**

**onvolumechange 声音改变时触发(音量按钮,音量进度条)**

### 3.5.2常用js属性

duration 音频总时长(返回未格式化的秒,带小数的)  
 currentTime 音频已经播放时长(返回未格式化的秒)  
 volume: 0~1的任意值。控制音量。(点击静音按钮对应的音量有问题)  
 muted: 布尔值。静音。(ture表示静音,false表示非静音。拖动进度条对应的布尔值也有问题)

paused: 布尔值。音频文件是否暂停。(ture表示暂停,false表示播放)

ended: 布尔值。音频文件播放结束(ture表示播放结束,false表示播放中或者暂停)

//音视频的播放进度=音视频已经播放的时长/总时长

### 3.5.3常见的js方法

play() 从当前位置播放  
 pause() 如果音频在播放中，则暂停播放  
 load() 重新加载媒体

## 3.6自定义播放器

视频画面恢复成初始状态，只需要让当前视频播放时间恢复成0。

# 4.canvas

## 4.1介绍

标签：<canvas></canvas>

默认大小300 x 150

IE8不支持

\*\*\*注:不能在css中设置宽高,图形会发生错乱,直接在标签里设置宽高,也可以在js里设置,不用加style.

## 4.2使用步骤

4.2.1 获取画布

var myCanvas = document.getElementById('myCanvas');

4.2.2 获取画笔（上下文）

var painting = myCanvas.getContext('2d');

4.2.3开始绘制新路径（从新起笔）

painting.beginPath();

4.2.4 开始绘画

方法一:

//填充矩形

//参数1、2：矩形左上角的坐标

//参数3、4：矩形的宽高

painting.fillRect(0,0,100,100) /\*不用加单位\*/

//描边矩形

painting.strokeRect() //参数一样,所有的矩形参数都一样

方法二:

painting.rect(50,50,100,100);

painting.fill();

\*\*painting.beginPath(); //解决影响

painting.rect(50,50,100,100);

painting.stroke(); //对之前的所有图形都会产生影响

注: fill 填充 stroke 描边 rect 矩形. 使用painting.rect()必须配合描边或者填充方法一起使用，否则无法显示图像.

## 4.3给元素添加样式

4.3.1 填充图形样式

painting.fillStyle = 'red' //填充颜色

4.3.2 描边图形样式

painting.strokeStyle = 'blue' //线条颜色

painting.lineWidth = '10' //线条粗细

描边注意问题:在原有盒子基础上，描边会里外均等分布

## 4.4橡皮擦(矩形)

painting.clearRect(120,0,100,100);

//通常是清除整个画布

painting.clearRect(0,0,myCanvas.width,myCanvas.height)

## 4.5线段

//线段起始点

painting.moveTo(100,100)

//线段结束点

painting.lineTo(300,100)

//线段末端

painting.lineCap = 'butt' : 线段末端以方形结束。(默认值)  
 round : 线段末端以圆形结束,在末端加一个圆形  
 square: 线段末端以方形结束，但是增加了一个宽度和线段相同， 高度是线段厚度一半的矩形区域

注: 线条状的图形加painting.stoke(); 闭合图形加painting.fill();只有矩形可以不加.

线段案例:

三角形

painting.strokeStyle = 'red';  
 painting.moveTo(100,100)  
 painting.lineTo(300,100)  
 painting.lineTo(300,200)  
 painting.closePath() //首尾相连(闭合路径)

painting.stroke()

设置线段连接处：

painting.lineJoin = miter : 直角 //默认值

round : 圆角

bevel : 斜角

## 4.6圆形与圆弧

### 4.6.1圆形

painting.arc(200,200,100,0,360/180\*Math.PI,false);

参数说明：

参数1，参数2：圆心点坐标

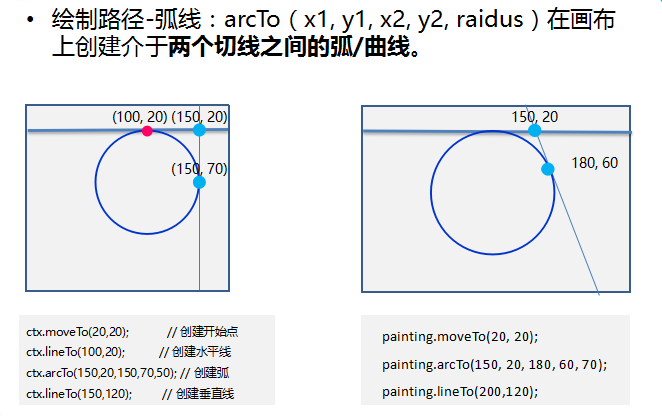
参数3：半径

参数4，参数5：圆的起始弧度与结束弧度

参数6：是否顺时针画图,布尔值,默认false顺时针

### 4.6.2 圆弧

painting.moveTo(20,20);  
 painting.lineTo(100,20);  
 painting.arcTo(150,20,150,70,20)  
 painting.lineTo(150,120);



## 4.7元素变换

变换的语句都要写到图形之前

### 4.7.1 位移

painting.translate(100,100);

位移注意问题：位移的语句必须写在元素之前,位移改变的是坐标轴，之前的盒子不会受影响

### 4.7.2 缩放

painting.scale(2,0.5)

注:缩放写在位移之前,元素的大小发生改变,位移的距离也发生改变,变换基点在左上角.

### 4.7.3 旋转

painting.rotate(45/180\*Math.PI)

注:括号内的数值要写弧度,变换基点在左上角.如果旋转写在位移之前,元素会沿着新坐标轴的方向发生位移.

注:旋转和位移实质上都是坐标轴发生的变换.

painting.save() // 保存当前的操作,不对下一次的操作产生影响

painting.restore()

## 4.8加载图片

1) 加载图片：painting.drawImage();

2) 步骤：

首先，需要使用图片就得有img标签

var img = new Image();

其次，引入图片路径

img.src = 'img/....';

然后，等图片加载完成后再去设置图片显示

img.onload = function() {};

最后，图片显示

drawImage( img , 0 , 0 , 100 , 100 ):

参数1：图片的dom对象

参数2，参数3：图片在画布中显示的初始位置

参数4，参数5：图片在画布中的宽度与高度

## 4.9渐变

1) 线性渐变：

var linearG = painting.createLinearGradient(20,20,100,100);

参数1，参数2，参数3，参数4 的连线决定了渐变的 **方向 和 区间**

2) 添加渐变色：

linearG.addColorStop(0,'red')

linearG.addColorStop(0.5,'green')

linearG.addColorStop(1,'blue')

参数1: 0-1之间的小数,颜色占据的比例  
 参数2: 颜色

3) 添加到盒子里面

painting.fillStyle = linearG;

4) 径向渐变：

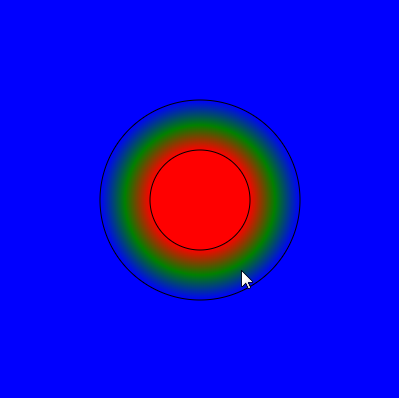
painting.createRadialGradient(200,200,50,130,200,100);

参数1，参数2，参数3，第一个小圆的圆心和半径  
 参数4，参数5，参数6，第二个大圆的圆心和半径

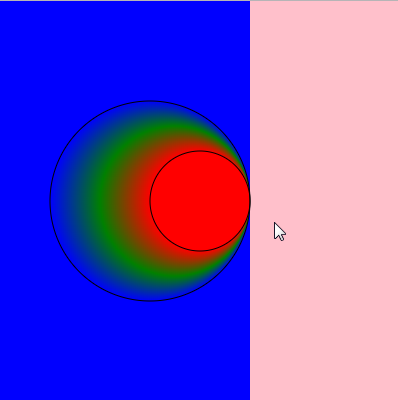
测试:

var RadialG = painting.createRadialGradient(200,200,50,200,200,100);

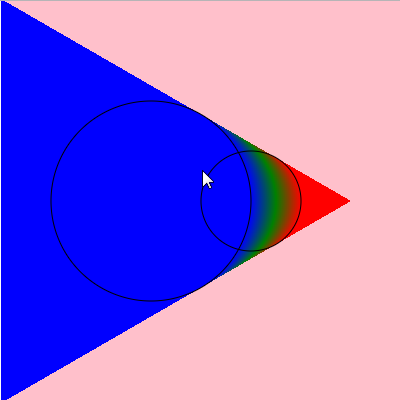
大圆套小圆,渐变方向小圆外到大圆里



var RadialG = painting.createRadialGradient(200,200,50,150,200,100);

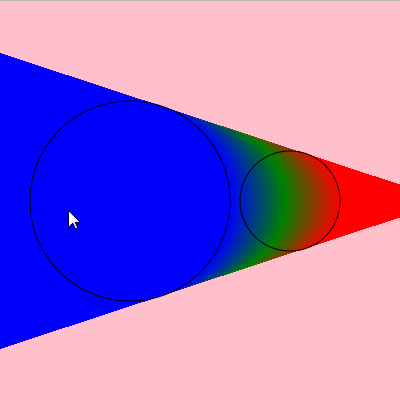
大圆套小圆,出现公切线,公切线外面没有颜色,渐变方向小圆外到大圆里  
 

var RadialG = painting.createRadialGradient(250,200,50,150,200,100);

出现两条公切线,渐变方向在两条公切线之间,由小圆里到大圆外  
 

var RadialG = painting.createRadialGradient(290,200,50,130,200,100);

出现两条公切线,渐变方向在两条公切线之间,由小圆里到大圆外



## 4.10 文字

1) 填充文字：painting.fillText('你好啊',50,50);

注: 坐标点对齐文字的基线

2) 镂空文字：painting.strokeText('你好啊',50,50);

3) 文字样式：painting.font = 'bold 40px 微软雅黑';

4)水平对齐方式

painting.textAlign = 'start'; --- 默认值

painting.textAlign = 'center';

painting.textAlign = 'end';

注: 设置的是坐标对于文字整体的位置

5)垂直对齐方式

painting.textBaseline = 'top'; 文字顶部有线有缝隙  
 painting.textBaseline = 'hanging'; 文字顶部没有线有缝隙  
 painting.textBaseline = 'middle';  
 painting.textBaseline = 'bottom'; 文字底部有线有缝隙  
 painting.textBaseline = 'alphabetic'; 默认值

## 4.11 阴影

painting.shadowColor = 'blue'; 阴影颜色设置  
 painting.shadowBlur = 20; 阴影模糊程度  
 painting.shadowOffsetX = 100; 阴影水平方向偏移量  
 painting.shadowOffsetY = 50; 阴影垂直方向偏移量

注: 添加多个阴影用painting.save()包起来

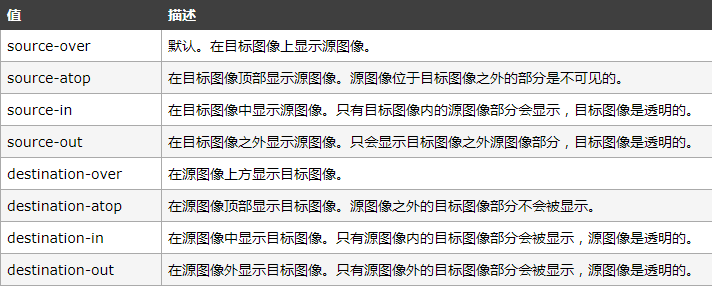
## 4.12图片合成

1)定义：globalCompositeOperation属性设置或返回如何将一个源（新的）图像绘制到目标（已有）的图像上

源图像 = 您打算放置到画布上的绘图

目标图像 = 您已经放置在画布上的绘图

2) 属性值：



## 4.13像素操作

### 4.13.1读取像素

提取矩形中的信息  
 painting.getImageData(100,100,100,100);  
 返回值：参数1参数2 读取像素区域左上角坐标

参数3参数4 读取像素区域宽度 高度

返回值是一个对象

width:读取区域的宽度,该区域横向上像素点的个数  
 height:读取区域的高度,该区域纵向上像素点的个数

data:数组,读取区域所有像素点的rgba信息,每一个像素上有四个数字来表示颜色rgba(0,0,0,0)黑色的透明色

注: 1.只能读取画布元素上的颜色,不能读取画布的颜色;如果画布上边没有元素,返回的颜色是黑色的透明色.

2.rgba表达元素透明度,canvas的范围是0-255,css的范围是0-1.

2)写入像素：

painting.putImageData(填入的对象,填充区域水平坐标点,填充区域垂直坐标点);

填入的对象自带宽高

### 4.13.2创建像素对象

painting.createImageData(100,100)

参数一 参数二: 宽度 高度

返回值也是个对象,

width:读取区域的宽度,该区域横向上像素点的个数  
 height:读取区域的高度,该区域纵向上像素点的个数

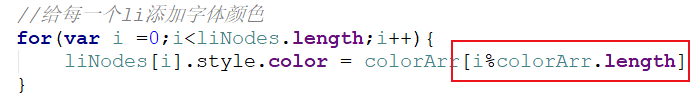
data:数组,读取区域所有像素点的rgba信息,每一个像素上有四个数字来表示颜色rgba(0,0,0,0)黑色的透明色.

# 项目技术

1.颜色、图片等并列关系的素材,通常放在数组里面保存。

2. js中判断条件只有两个时,尽量往布尔值上靠。

3. 求数字的范围时往取余上靠。



4.h1通常只有一个,用来放公司的logo

5.用a元素模拟按钮

6.每一个元素都得有高度

7.\*\*\*li在一行排列并且总宽度大于屏幕时,浮动不能让他们在一行排列,可以把li的类型转换成inline-block,然后给父元素设置文本不换行white-space: nowrap;并让父元素飘起来,否则拿不到li的总宽度.

8.碰到input元素使用属性选择器;设置placeholder提示文本的颜色:

input::-webkit-input-placeholder{ color: red; }