## figure

<figure>

<img src="E:\源码\HTML5 + CSS3\HTML5\code\img\banner\_1.jpg" alt="123">

<figcaption>title</figcaption>

<p>text<box/p>

</figure>

## datalist

<input type="text" list="abcd" />

<datalist id="abcd">

<option value="110">报警</option>

<option value="120">急救</option>

<option value="114">查询</option>

</datalist>

## details

<details>

<summary>more</summary>

121dlfjslkfjdlkfjsalkfjsdlkfjlk

</details>

## section

<section>

定义文档中的某一个区块。

建议 section 只出现 标题（h1~h6）和内容（p）,还可以使用 Img

一般的时候，可以用在一些展示性的列表（有标题和段落、或有标题段落和图片）

</section>

## audio

<audio src="audio/浪人情歌.mp3" controls autoplay loop preload muted></audio>

<audio controls >

<source src="audio/浪人情歌.mp3" type="audio/mpeg">

</audio>

controls 显示浏览器自带的控制条 （每个浏览器都不一样）

autoplay 自动播放

muted 静音

loop 循环

preload 预加载

当 元素身上出现了 autoplay 默认就是预加载的

href 和 src区别

href 引入资源 可以并行加载 link . href

src 替代资源 当该文件没有加载完成的时候，不会进行后面的操作。

音频转换 工具：

格式工厂

## video

<video poster="img/1.jpg" controls width="800px" height="600px">

<source src="video/1.mp4" type="video/mp4">

</video>

controls 显示浏览器自带的控制条 （每个浏览器都不一样）

autoplay 自动播放

muted 静音

loop 循环

preload 预加载

当 元素身上出现了 autoplay 默认就是预加载的

poster="i.jpg" 封面（用在未播放之前的图片）

媒体元素

controls : 显示或隐藏用户控制界面

autoplay : 媒体是否自动播放

loop : 媒体是否循环播放

currentTime : 开始到播放现在所用的时间

duration : 媒体总时间(只读)

volume : 0.0-1.0的音量相对值

muted : 是否静音

autobuffer : 开始的时候是否缓冲加载，autoplay的时候，忽略此属性

媒体元素\_2

paused : 媒体是否暂停(只读)

ended : 媒体是否播放完毕(只读)

error : 媒体发生错误的时候，返回错误代码 (只读)

currentSrc : 以字符串的形式返回媒体地址(只读)

play() : 媒体播放

pause() : 媒体暂停

load() : 重新加载媒体

媒体元素\_3

loadstart progress suspend emptied stalled play pause loadedmetadata loadeddata waiting playing canplay canplaythrough seeking seeked timeupdate ended ratechange durationchange volumechange

var oV = document.getElementById('v1');

oV.poster = '2.png';

oV.width = 500;

oV.height = 500;

alert( oV.videoWidth );

oV.addEventListener('ended',function(){

alert(123);

},false);

视频转换 工具：

webm 狸窝

ogv ogv转换工具

## input (type = number date url tel range(拖动条) search)

<input type="date" name="date" />

<input type="number" name="number" min="0" max="100" step="2" />

<input type="range" name="range" min="0" max="100" value="50" step="2">

<input type="search" name="search">

<input type="url">

<input type="tel">

## input 新属性 autofocus autocomplete pattern

## .box::first-letter .box::first-line .box::selection :target

## 属性选择器：

div[data-hover] 匹配div 包含 某个属性的

div[class="abc"] 匹配div class = 什么的

div[class^="abc"] 以什么什么开头

div[class$="abc"] 以什么什么结尾

div[class\*="abc"] 包含 什么什么 的'

多重（多条件）

div[class\*="asdf"][id][attr="123"]

## 伪类选择器：

编号默认从1开始

:nth-child(编号 | 多少倍 | 运算)

运算符前后 必须要加空格

.list li:nth-child(3n - 1)

.list li:nth-last-child(5)

伪类选择器：

first-child

last-child

.list li:first-child

伪类选择器：

of-type 同类型的 第几个：

nth-of-type

nth-last-of-type

first-of-type

last-of-type

.list li:last-of-type

odd 奇数

even 偶数

div:nth-child(odd)

:only-child 选择只有一个子元素的东西

.box h3:only-child

.box p:only-of-type

:empty 空， 指的是 不能含有任何的字符 （enter）

.box:empty

not()

.abc \*[class\*='aaa']:not(p)

不是什么东西

## 表单

:enabled 激活的

:disabled 未被激活的

:checked 选中的

## box-shadow

box-shadow: x y size 增强（可选参数） color;

x轴 y轴 大小（不允许使用负值） 增强（可选参数） 颜色

box-shadow: 0 0 10px red;

box-shadow: 0 0 10px 5px red;

box-shadow: inset 0 0 10px red; /\* inset 是往内的阴影 \*/

box-shadow: -10px 0 10px red,0 10px 10px yellow,10px 0 10px orange,0 -10px 10px blue; /\* 多个条件可以用逗号隔开\*/

## text-shadow

text-shadow: 0 0 10px red;

## 文字描边

-webkit-text-stroke:宽度 颜色

## border-radius

border-radius: 20px;

## calc

calc(200px - 2\*5px);

## background

background: rgba(0,255,0,0.5); 文字不透明

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

background-color: #0f0;

opacity: 0.5; 文字透明

background: url(img/2.png) top left no-repeat,url(img/2.png) top right no-repeat,url(img/1.jpg) center center no-repeat;

多幅图片可以用逗号隔开

background-size: cover;

background-size:

x y 百分比的值

如果x给了值，y轴默认auto

contain

等比例缩小 来适应元素

cover

等比例放大 填满元素

auto

默认 跟没写一样

缩写

size 要写在 position 之后 加 /

属性兼容性查看：<http://caniuse.com/>

background-origin: padding-box;

background-origin: 背景图像绘制区域：

content-box border-box padding-box

background-clip: padding-box;

background-clip: 背景绘制区域：

content-box border-box padding-box

text：只在文字内显示背景

## 遮罩

【mask-image】

　　默认值为none，值为透明图片，或透明渐变

【mask-repeat】

　　默认值为repeat，可选值与background-repeat相同

【mask-position】

　　默认值为0 0，可选值与background-position相同

【mask-clip】

　　默认值为border-box，可选值与background-clip相同

【mask-origin】

　　默认值为border-box，可选值与background-origin相同

【mask-size】

　　默认值为auto，可选值与background-size相同

【mask-mode】

　　默认值为match-source，可选值为alpha、luminance、match-source，或者它们的组合

【mask-composite】

　　默认值为add，可选值为add、subtract、intersect、exclude

　　[注意]只有firefox支持mask-mode和mask-composite

.box{

width:800px;height:600px;

background:url(miaov.jpg) no-repeat;

background-size:100% 100%;

border:10px solid #000;

-webkit-mask:url(mask.png) no-repeat;

transition:1s;

}

.box:hover{

-webkit-mask-position:100% 100%;

}

## 渐变

线性渐变

background-image: linear-gradient(to bottom,red,orange,yellow,green,skyblue,purple);

background-image:linear-gradient(red 0,red 10px,yellow 10px,yellow 20px);

background-image:linear-gradient(red 30%,yellow 70%,blue 100%);

linear-gradient(red [百分比、像素],yellow [百分比、像素]) 颜色1 颜色2 颜色3.......

linear-gradient(red 百分比1,yellow 百分比2,blue 百分比3) 表示从百分比1到百分比2是从red到yellow的渐变 从百分比2到百分比3是yellow到blue的渐变

方向：

1 to top、to bottom 、 to right 、 to left

2 角度： 单位 deg

0deg = to top

90deg = to right

180deg = to bottom

270deg = to left

正值 顺时针旋转的

支持负值 逆时针旋转的

径向渐变

background-image: radial-gradient(red,yellow);

## box-sizing

box-sizing: border-box;

box-sizing: content-box (默认值 跟没给没有区别)

border-box 元素的宽度 就是给的宽度,元素的高度、宽度认为是content+padding+border

- 不会去计算 padding值border值

## 弹性布局1

开启弹性布局：

display: -webkit-box;

ul{

height:400px;

border: 1px red solid;

display: -webkit-box;

-webkit-box-pack:center;

-webkit-box-align:end;

/\*

ul 是一个弹性的盒子

-webkit-box-pack: 水平对齐方式

start | center | end | justify（分散对齐）

-webkit-box-align: 垂直对齐方式

start | end | center | baseline | stretch(默认值)

\*/

-webkit-box-direction: reverse;

/\*

-webkit-box-direction: reverse;

是否反转

reverse 反转

normal 正常

\*/

-webkit-box-orient:vertical;

/\*

-webkit-box-orient: 排列方式

不给 默认为 水平

horizontal 水平

vertical 垂直

\*/

}

ul li{

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

/\*

均分剩余宽度：

1 代表几份的意思

\*/

list-style: none;

height: 40px;

}

## 弹性布局2

display: flex;

/\*

display:flex;

开启弹性布局

flex: 1

均分宽度 与其中的文字无关

1 份数

\*/

-webkit-flex-direction:row;

/\*

-webkit-flex-direction （排列方式 - 决定主轴的方向）

row(默认) | row-reverse | column | column-reverse

水平方向，起点在左端 | 水平方向，起点在右端 | 垂直方向，起点在上沿| 垂直方向，起点在下沿

\*/

-webkit-flex-wrap: nowrap;

/\*

-webkit-flex-wrap:

nowrap | wrap | wrap-reverse

单行 | 换行(宽度不够自动换行) | 换行并反转

\*/

justify-content: flex-end;

align-items:center;

/\*

justify-content（水平对齐方式）

flex-start | flex-end | center | space-between | space-around;

左对齐 | 右对齐 | 居中对齐 | 两端对齐(之间的间隔相等) | 每个项目两侧的间隔相等

align-items（垂直对齐方式）

flex-start | flex-end | center | baseline | stretch

起点对齐 | 终点对齐 | 中点对齐 | 基线对齐 | 默认值(未设置高度或设为auto，将占满整个容器的高度)

align-content （多行对齐方式）

注：需要多行才行

多根轴线的对齐方式

flex-start | flex-end | center | space-between | space-around | stretch

左对齐 | 右对齐 | 居中对齐 | 两端对齐(之间的间隔相等) | 每个项目两侧的间隔相等 | 轴线占满整个交叉轴

align-self

允许单个项目有与其他项目不一样的垂直对齐方式

auto | flex-start | flex-end | center | baseline | stretch

该属性可能取6个值，除了auto，其他都与align-items属性完全一致。

可覆盖align-items属性

默认值auto，表示继承父级元素的align-items，如果没有父级，则等同于stretch

当这两个属性遇到了

flex-direction: column / column-reverse 整个顺序就都变了。

\*/

-webkit-flex: 1 1 300px;

/\*

缩写 flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

\*/

/\*

解决UC、微信兼容性问题

安卓平台的uc和微信

要加-webkit-

display: flex;

display: -webkit-box;

flex: 1;

-webkit-box-flex: 1;

-moz-box-flex: 1;

box-flex: 1;

webkit、moz、原生

align-items: center;

-webkit-box-align: center;

flex-direction: column;

box-orient: vertical;

webkit、moz、原生

\*/

}

ul li{

height: 30px;

text-align: center;

flex: 1;

-webkit-flex-shrink:1;

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

/\*

为复杂布局而生

-webkit-flex-shrink

缩小比例

此时剩余空间是不足时都将等比例缩小

0 表示不参加收缩比例

若没写该属性，则为1

-webkit-flex-grow

扩展比率

剩余空间是正值的时，伸缩项目相对于伸缩容器里其他伸缩项目能分配到空间比例

若没写该属性，则为0

0代表不参与扩展

\*/

}

ul li:nth-child(odd){

background-color: red;

align-self: flex-end;

-webkit-flex-basis: 200px;

/\*

flex-basis

在分配空间之前，也已经分得到空间

（最小宽度）

\*/

}

## user-modify

使div可以输入

-webkit-user-modify: read-write;

/\*

-webkit-user-modify

read-only | read-write | read-write-plaintext-only

只读 | 内容可读写 | 内容可读写，但富粘贴的文本格式会丢失

另一个方法

<div class="box" contenteditable="true"></div>

\*/

## user-select

控制文本是否可选中

-webkit-user-select: none;

/\*

auto | none | text

用户可选内容（包括样式、图片） | 不可选 | 可选文本（仅文本）

\*/

## 滤镜

/\*-webkit-filter: brightness(.5)\*/

/\*filter: contrast(.5);\*/

/\*filter: grayscale(.5);\*/

/\*filter: hue-rotate(0deg);\*/

/\*filter: saturate(.5);\*/

/\*filter: sepia(.5);\*/

/\*filter: drop-shadow(x y radius color);\*/

/\*filter: drop-shadow(30px 30px 20px red);\*/



## 倒影

-webkit-box-reflect: below 0 -webkit-linear-gradient(transparent,transparent 50%,rgba(255,255,255,.6));

-webkit-box-reflect：none | <direction> <offset>? <mask-box-image>?

box-reflect：none | <direction> <offset>? <mask-box-image>?

从box-reflect语法中可以得知，其主要包括以下几个属性值：

* none:此值为box-reflect默认值，表示无倒影效果；
* <direction>:此值表示box-reflect生成倒影的方向，主要包括以下几个值：
  + above:表示生成的倒影在对象（原图）的上方；
  + below:表示生成的倒影在对象（原图）的下方；
  + left:表示生成的倒影在对象(原图)的左侧；
  + right:表示生成的倒影在对象(原图)的右侧；
* <offset>:用来设置生成倒影与对象（原图）之间的间距，其取值可以是固定的像素值，也可以是百分比值，如：
  + :使用长度值来设置生成的倒影与原图之间的间距，只要是CSS中的长度单位都可以，此值可以使用负值；
  + :使用百分比来设置生成的倒影与原图之间的间距，此值也可以使用负值
* <mask-box-image>:用来设置倒影的遮罩效果，可以是背景图片，也可以是渐变生成的背景图像。当box-reflect属性中的<mask-box-image>属性值出现时，必须显式的设置<offset>值，如果不需要间距，将其设置为0。

## transform

## translate

transform: translate(200px,100px);

/\*

transform 变换

translate 2d平移

默认 第一个参数 x 第二个参数y

如果y没给，为0

translateX 、translateY、translateZ(需要开启3d才能看到)

支持正负值

\*/

## scale

transform: scale(1);

/\*

缩放：

单位 倍数

scale(1.5) 原始（给的宽度）大小的1.5倍

1 为 原始比例

如果没有写方向，默认是 x和y

scaleX \ Y

\*/

## rotate

transform: rotate(45deg);

/\*

旋转：

单位 ： 角度 （deg）

rotate 如果没去置顶xyz，默认为 z轴

rotateX \ Y \ Z

\*/

## origin

transform-origin: left top center;

/\*

transform-origin: 变幻的轴心 以哪个点做为 原点

值：transform-origin : x y z

默认值是 center center

可以用px值

\*/

## skew

transform: skew(45deg);

/\*

斜切：

单位：角度 支持正负值

skew 如果没写第二个参数，默认为0

skewX \ Y

\*/

执行顺序 ： 后写的先执行

body{height:300px;border:1px solid #000;}

.box{width:100px;height:100px;background:Red;margin:10px; transition:1s;}

body:hover .box:nth-of-type(1){ -webkit-transform:translateX(100px) scale(0.5);}

先scale后translate

body:hover .box:nth-of-type(2){ -webkit-transform:scale(0.5) translateX(100px);}

先translate后scale

## 矩阵

matrix(a,b,c,d,e,f) 矩阵函数

通过矩阵实现缩放

x轴缩放 a=x\*a c=x\*c e=x\*e;

y轴缩放 b=y\*b d=y\*d f=y\*f;

通过矩阵实现位移

x轴位移: e=e+x

y轴位移: f=f+y

通过矩阵实现倾斜

x轴倾斜: c=Math.tan(xDeg/180\*Math.PI)

y轴倾斜: b=Math.tan(yDeg/180\*Math.PI)

matrix(a,b,c,d,e,f) 矩阵函数

通过矩阵实现旋转

a=Math.cos(deg/180\*Math.PI);

b=Math.sin(deg/180\*Math.PI);

c=-Math.sin(deg/180\*Math.PI);

d=Math.cos(deg/180\*Math.PI);

变换兼容IE9以下IE版本只能通过矩阵来实现

filter: progid:DXImageTransform.Microsoft.Matrix( M11= 1, M12= 0, M21= 0 , M22=1,SizingMethod='auto expand');

IE下的矩阵没有E和F两个参数 M11==a; M12==c; M21==b; M22==d 不能实现位移

初始值：

matrix(1,0,0,1,0,0)

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

progid:DXImageTransform.Microsoft.Matrix( M11= 1, M12= 0, M21= 0 , M22=1,SizingMethod='auto expand');

Matrix( M11= a, M12= c, M21= b , M22=d,SizingMethod='auto expand');

位移:

x:e+disX

y:f+disY

body:hover .box{ -webkit-transform:matrix(1,0,0,1,200,300);}

缩放:

x轴缩放 a=x\*a c=x\*c;

y轴缩放 b=y\*b d=y\*d;

oBox.style.WebkitTransform="matrix(0.5,0,0,0.2,0,0)";

oBox.style.MozTransform="matrix(0.5,0,0,0.2,0,0)";

oBox.style.transform="matrix(0.5,0,0,0.2,0,0)";

oBox.style.filter="progid:DXImageTransform.Microsoft.Matrix( M11= 0.5, M12= 0, M21= 0 , M22=0.2,SizingMethod='auto expand')";

x轴斜切: c=Math.tan(xDeg/180\*Math.PI)

y轴斜切: b=Math.tan(yDeg/180\*Math.PI)

oBox.style.WebkitTransform="matrix(1,0,0.58,1,0,0)";

oBox.style.MozTransform="matrix(1,0,0.58,1,0,0)";

oBox.style.transform="matrix(1,0,0.58,1,0,0)";

oBox.style.filter="progid:DXImageTransform.Microsoft.Matrix( M11= 1, M12= 0.58, M21= 0 , M22=1,SizingMethod='auto expand')";

旋转：

a=Math.cos(deg/180\*Math.PI);

b=Math.sin(deg/180\*Math.PI);

c=-Math.sin(deg/180\*Math.PI);

d=Math.cos(deg/180\*Math.PI);

oBox.style.WebkitTransform="matrix(0.87,0.5,-0.5,0.87,0,0)";

oBox.style.MozTransform="matrix(0.87,0.5,-0.5,0.87,0,0)";

oBox.style.transform="matrix(0.87,0.5,-0.5,0.87,0,0)";

oBox.style.filter="progid:DXImageTransform.Microsoft.Matrix( M11= 0.87, M12= -0.5, M21= 0.5 , M22=0.87,SizingMethod='auto expand')";

IE下的旋转基点不是中心 所以要修正

var oBox=document.getElementById("box");

var iDeg=0;

setInterval(function(){

iDeg++;

toRotate(oBox,iDeg);

},30);

function toRotate(obj,iDeg)

{

var a=Math.round(Math.cos(iDeg/180\*Math.PI)\*100)/100;

var b=Math.round(Math.sin(iDeg/180\*Math.PI)\*100)/100;

var c=-b;

var d=a;

obj.style.WebkitTransform="matrix("+a+","+b+","+c+","+d+",0,0)";

obj.style.MozTransform="matrix("+a+","+b+","+c+","+d+",0,0)";

obj.style.transform="matrix("+a+","+b+","+c+","+d+",0,0)";

obj.style.filter="progid:DXImageTransform.Microsoft.Matrix( M11="+a+", M12= "+c+", M21= "+b+" , M22="+d+",SizingMethod='auto expand')";

//以下是修正方法

obj.style.left=(obj.parentNode.offsetWidth-obj.offsetWidth)/2+"px";

obj.style.top=(obj.parentNode.offsetHeight-obj.offsetHeight)/2+"px";

}

## transition

过渡

.box{

width: 200px;

height: 200px;

background-color: orange;

margin: 100px auto;

transition: all 2s;

/\*transition: 哪个属性动 运动在多长时间内完成 延迟时间 运动的类型;\*/

/\*transition: width 5s linear, height 2s linear;\*/

/\*transition: all 5s 2s ease;\*/

/\*transition: all 5s ease-in-out;\*/

/\*

linear 线性过渡(匀速) 跑高速的时候，一直100公里/小时

ease 平滑过渡(缓冲) 到达目的之的时候，松开了油门

ease-in 由慢到快(加速) 起步，踩油门

ease-out 由快到慢(减速) 刹车了

ease-in-out 由慢到快再到慢(先加后减 ) 起步，遇到s\*插队，减速

steps(数字, start/end) 分步动画(逐帧动画) 新手，起步 熄火 起步....

参数2未设定为 end

\*/

/\*move(obj,{width:},time)\*/

}

.box:hover{

width: 500px;

height: 500px;

opacity: 0.3;

}

过渡完成事件

Webkit内核： obj.addEventListener('webkitTransitionEnd',function(){},false);

firefox: obj.addEventListener('transitionend',function(){},false);

var oBox=document.getElementById("box");

oBox.onclick=function()

{

this.style.width=this.offsetWidth+100+"px";

addEnd(oBox,end);

};

function end()

{

this.style.width=this.offsetWidth+100+"px";

removeEnd(this,end);

}

function addEnd(obj,fn)

{

obj.addEventListener('WebkitTransitionEnd',fn,false);

obj.addEventListener('transitionend',fn,false);

}

function removeEnd(obj,fn)

{

obj.removeEventListener('WebkitTransitionEnd',fn,false);

obj.removeEventListener('transitionend',fn,false);

}

## steps

.box{

width: 97px;

height: 91px;

background-color: orange;

margin: 100px auto;

background-image: url(img/1.jpg);

transition: steps(7) .5s;

/\*

steps(数字, start/end) 分步动画(逐帧动画) 新手，起步 熄火 起步....

参数2未设定为 end

\*/

}

.box:hover{

background-position: 0 -637px;

}

## perspective

body{

perspective: 200px;

/\*景深 \*/

/\*

景深：

值越小 代表的是我们离物体越近

一般推荐大家使用 500~800

开启3D必须，在需要开3D的元素的父元素上必须设置perspective

\*/

}

.box{

width: 200px;

height: 200px;

margin: 100px auto;

background-color: orange;

transition: .5s;

transform-style: preserve-3d;

transform-style: preserve-3d;

/\*

开启3D

flat | preserve-3d

设置该元素在2d平面 | 3d空间

\*/

/\*

flat | preserve-3d

设置该元素在2d平面 | 3d空间

\*/

}

.box:hover{

transform: translateZ(100px);

}

Perspective- origin 景深基点

.box{

width:100px;

height:100px;

background:red;

position:relative;

-webkit-transform-style:preserve-3d;

transition:2s;

-webkit-transform-origin:center center -50px;

}

## animation

/\*transition: 谁（要动的属性） 时间 延时（可选） 运动方式（linear、ease...）;\*/

/\*animation: js中的函数名 时间 延时（可选） 运动方式（linear、ease...）;\*/

animation: move 2s linear;

animation-iteration-count: infinite;

/\*

animation-name: move; 名称

animation-duration: 5s; 执行时间

animation-timing-function: initial; 运动类型（linear、ease...）

animation-delay: initial; 延迟

animation-iteration-count: initial; 循环次数 可选 number | infinite

animation-direction: initial; 是否反向运动 alternate

animation-fill-mode: initial;

设置或检索动画时间之外的状态

none | forwards | backwards | both

不设置 | 结束时 | 开始时 | 结束或开始的状态

animation-play-state: initial; 动画状态 running | paused

\*/

}

/\*@keyframes move{

from{

width: 100px;

}

to{

width: 600px;

}

}\*/

标准下：

@keyframes move{

0%{

left: 0;

}

100%{

left: 100%;

}

}

Chrome：

@-webkit-keyframes move

{

0%{

width:100px;

}

100%

{

width:500px;

}

}

火狐：

@-moz-keyframes move

{

0%{

width:100px;

}

100%

{

width:500px;

}

}

动画结束后的动作

obj.addEventListener('webkitAnimationEnd', function (){}, false);

## viewport

Viewport(可视区尺寸)

移动设备上的viewport就是设备的屏幕上能用来显示我们的网页的那一块区域

浏览器上(也可能是一个app中的webview)用来显示网页的那部分区域

它可能比浏览器的可视区域要大，也可能比浏览器的可视区域要小。

默认情况下，手机网页采用980px的宽

content 参数

width – viewport宽度(数值/device-width)

也可以是具体的数值

height – viewport高度(数值/device-height)

initial-scale – 初始缩放比例

任意值

user-scalable – 是否允许用户缩放(yes/no) | (0/1)

maximum-scale – 最大缩放比例

minimum-scale – 最小缩放比例

<meta name="viewport" id="viewport" content="width=375,initial-scale=1,user-scalable=1">

只需要了解：（

content

target-densitydpi

仅供了解

一个屏幕像素密度是由屏幕分辨率决定的，通常定义为每英寸点的数量（dpi）。Android支持三种屏幕像素密度：低像素密度，中像素密度，高像素密度。一个低像素密度的屏幕每英寸上的像素点更少，而一个高像素密度的屏幕每英寸上的像素点更多。Android Browser和WebView默认屏幕为中像素密度。

取值范围：

device-dpi: 使用设备原本的 dpi 作为目标 dp。 不会发生默认缩放。

high-dpi: 使用hdpi 作为目标 dpi。 中等像素密度和低像素密度设备相应缩小。

medium-dpi: 使用mdpi作为目标 dpi。 高像素密度设备相应放大， 像素密度设备相应缩小。 这是默认的target density.

low-dpi: 使用mdpi作为目标 dpi。中等像素密度和高像素密度设备相应放大。

<value>: 指定一个具体的dpi 值作为target dpi. 这个值的范围必须在70–400之间。

为了防止Android Browser和WebView 根据不同屏幕的像素密度对你的页面进行缩放，你可以将viewport的target-densitydpi 设置为 device-dpi。当你这么做了，页面将不会缩放。相反，页面会根据当前屏幕的像素密度进行展示。在这种情形下，你还需要将viewport的width定义为与设备的width匹配，这样你的页面就可以和屏幕相适应。

）

## CSS的新单位

rem

root the element (font size of the root element)

vw / vh

view width / height 屏幕被分为100份

.box{

width: 10vw;

height: 10vh;

background-color: yellowgreen;

}

vmax / vmin

最大 / 最小

不是我们所理解的 最小高度那个意思

最小的或最大的 视窗宽高的某一个

相对于视口的宽度或高度中较大/较小的那个。其中最大/最小的那个被均分为100单位的vmax/vmin

## retina

2倍 或者 3倍的屏幕

.div3{

background: url(icon@3x.png) 0 0 no-repeat;

background-image: -webkit-image-set(url(icon.png) 1x,url([icon@2x.png](mailto:icon@2x.png)) 2x,url(icon@3x.png) 3x);

}

<img src="icon.png" srcset="icon@2x.png 2x" alt="">

## 媒体查询

声明：@media

条件：min-width、height/max-width、height设备类型

min-width 最小宽度 >=

max-width 最大宽度 <=

原则：

永远要从 最小的设备写起 （phone > pad > pc > pc宽屏）

.box{

width: 200px;

height: 200px;

background-color: red;

}

@media (min-width:768px){

.box{

width: 600px;

background-color: grey;

position: absolute;

}

}



and 并且

not 不是什么

only 不支持的忽略

device-width/min、max

device-width

max-device-width / min-device-width

@media (orientation : landscape) 横屏

@media (orientation : portrait) 竖屏

Link方式引入

<link rel="stylesheet" href="style/common.css" type="text/css">

<link rel="stylesheet" href="style/pad.css" type="text/css" media="(min-width:768px)">

<link rel="stylesheet" href="style/pc.css" type="text/css" media="(min-width:1000px)">

<link rel="stylesheet" href="style/pc-w.css" type="text/css" media="(min-width:1190px)">

## sass

less(偏前端的) / sass(偏后端的)

用这两个东西 可以换个方式写css

less视频

http://www.imooc.com/learn/102

http://www.imooc.com/learn/61

Sass 需要编译环境：

koala（目前已经停止更新了）

命令的方式

ruby

http://rubyinstaller.org/downloads（自备梯子）

- 安装时候 注意添加环境变量 （...PATH）

- 可以参考 R

- 安装路径中 不能有中文，不能有特殊符号，不能有空格。

- 编译sass的过程，路径中 也不能 有中文，不能有特殊符号，不能有空格。

（x64 - 64bit(位系统) 、 x86-32bit-32位系统）

安装命令：

gem install sass - win $ gem install sass - mac

- 需要切换根

<http://gems.ruby-china.org/>

注释：

// 后的注释不会被编译

/\* \*/ 后的注释会被编译

scss

sass

一次编译

sass 要编译的文件 编译后的文件

sass common.scss a.css

单文件监听

sass --watch common.scss:xx.css

文件夹监听

sass --watch 文件名名称（要监听的文件夹）：编译到哪儿（编译后的位置）

自行搜索 （windows cmd 命令大全）

dir 查看当前目录文件

cd 打开文件 cd a cd a.txt cd .. 上一层

cls 清除当前屏幕

ctrl + c 是否终止操作 y / n

按上下箭头 使用你最近使用的几个命令

--style 编译后的模式

nested 默认 expanded css纵向写法 compact 横向 compressed 压缩（所有都压缩成一行代码）

变量：

var aaa = 20;

$width:300px;

$w:300px 100px 200px;

语法：

// 变量 可以存 css单位、颜色、文字

$width:300px;

$height:200px;

$bgc:yellowgreen pink;

$ms:punctuation, PingFangSC-Regular, "Microsoft YaHei", "微软雅黑";

$bor:solid 20px red; //多值变量

background-color: nth(哪个变量,第几个)

background-color: nth($bgc,2);

//sass中的变量作用域 和js中的 作用域一样的

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

//嵌套 标签怎么嵌套 他就可以怎么嵌套

.list{

$lh:40px;

li{

height: $lh;

line-height: $lh;

}

.active{

background-color: yellowgreen;

}

a{

float: left;

}

span{

float: right;

width: 12px;

height: 12px;

background-color: yellow;

margin-top: ($lh - 12px) / 2;

}

}

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

//属性嵌套 是要用 ：来区分的

.box{

//复合样式

font:{

style:normal;

weight: bold;

size: 14px;

family: $ms;

}

margin: {

top:12px;

right: 12px;

bottom: 12px;

left: 12px;

}

}

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

.news2{

&\_title{

width: 200px;

}

}

& 表示连接上一级

// 这里在CSS中将会是

.news2\_title{

width: 200px;

}

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

.news{

//@at-root .news\_title{

// width: 10px;

// }

//@at-root .news\_content{

// width: 20px;

// }

@at-root {

.news\_title{

width: 10px;

}

.abc{

width: 10px;

}

}

}

@at-root 表示绕过上一级，在此处CSS中将不会以此形式：

.news .abc{

width: 10px;

}

而是

.abc{

width: 10px;

}

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

//混合 @mixin 声明

/\*

@mixin opa() 不写参数 直接可以调用

@mixin opa($opa) 调用时候 必须传参

@mixin opa($opa:40) 调用的时候，如果传了就用传了的，没传就用默认的

\*/

@mixin opa($opa:50){

opacity: $opa / 100; //运算符必须用空格隔开

filter:alpha(opacity=$opa);

}

//@include 是调用 @mixin 所声明的东西

@mixin shadow($shadow...){ //参数后加三个点表示参数不确定，可以是多个

box-shadow: $shadow;

}

@mixin sj($fx:top,$size:10px,$color:red){

@if ($fx == top) {

border-color: transparent transparent $color transparent;

border-style: dashed dashed solid dashed;

}

@else if($fx == right){

border-color: transparent transparent transparent $color;

border-style: dashed dashed dashed solid;

}

@else if($fx == bottom){

border-color: $color transparent transparent transparent ;

border-style: solid dashed dashed dashed ;

}

@else if($fx == left){

border-color: transparent $color transparent transparent ;

border-style: dashed solid dashed dashed ;

}

width: 0;

height: 0;

overflow: hidden;

border-width: $size;

}

.demo{

@include sj($fx:bottom,$color:yellowgreen);

//@include opa(80);

//@include shadow(0 0 10px red,0 0 20px yellow);

}

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

继承

.fl{

float: left;

}

.box{

width: 300px;

height: 300px;

@extend .fl;

}

.box2{

@extend .box;

@extend %clearfix;

}

//占位选择器 （在用到的时候 才会被编译出来）

%clearfix{

\*zoom:1;

&:after,

&:before{

content: '';

display: table;

}

&:after{

clear: both;

}

}

.abc{

width: 300px;

@extend %clearfix;

}

占位选择器的作用：自己不会再CSS中直接显示，如

. clearfix{

\*zoom:1;

}

而是被继承的时候才显示作用

如上的.abc

.abc{

width: 300px;

zoom:1;

}

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

//字符串插值

$icon:'../images/icon/';

$img:'../images/';

$sina:'http://img2.sina.com.cn/2016-11-21/qiaobusi/images/icon/';

// #{变量名}

.content{

background: url(#{$sina}a.jpg) left top no-repeat;

}

//sass中的运算 运算符的前后 必须要有空格

.ys{

color: #F00 + 255;

width: $width / 2;

$fs:12px;

$lh:24px;

font: #{$fs}/#{$lh} "Microsoft YaHei","微软雅黑";

}

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

//js的 for(var i=0;i<aaa.length;i++){}

$max:10;

$min:5;

//sass

// <=

@for $i from $min through $max{

.item\_#{$i}{

background-position: -($i - 1) \* 20px 0;

}

}

// <

@for $i from 1 to 10{

.demo\_#{$i}{

background-position: -($i - 1) \* 20px 0;

}

}

$a:5;

@while($a >= 1){

.aaa\_#{$a}{

width: 20px \* $a;

}

$a:$a - 1;

}

//each

$icon\_name:user,pass,checked,select;

@each $i in $icon\_name{

.icon\_#{$i}{

width: 10px;

}

}

//三元 ?成立:不成立

# 文本超出后以省略号显示

text-overflow: ellipsis;/\*文字如果超出后，以省略号的形式展现\*/

width: 85px; /\*通过宽度断定超出\*/

overflow: hidden;/\*超出要裁掉\*/

white-space: nowrap;/\*文字不换行\*/

# 换行

1. 强制不换行

div{

white-space:nowrap;

}

2. 自动换行

div{

word-wrap: break-word;

word-break: normal;

}

3. 强制英文单词断行

div{

word-break:break-all;

}

4. CSS设置不转行：

overflow:hidden 隐藏

white-space：normal 默认

pre 换行和其他空白字符都将受到保护

nowrap 强制在同一行内显示所有文本，直到文本结束或者遭遇 br 对象

5. 设置强行换行：

word-break:

normal ; 依照亚洲语言和非亚洲语言的文本规则，允许在字内换行

break-all : 　该行为与亚洲语言的normal相同。也允许非亚洲语言文本行的任意字内断开。该值适合包含一些非亚洲文本的亚洲文本

keep-all : 　与所有非亚洲语言的normal相同。对于中文，韩文，日文，不允许字断开。适合包含少量亚洲文本的非亚洲文本与之间的高度解决办法

英文不换行

CSS里加上 word-break: break-all; 问题解决。这个问题只有IE才有，在FF下测试,FF可以自己加滚动条，这样也不影响效果

建议大家做Skin时，记得在body里加 word-break: break-all; 这样可以解决IE的框架被英文撑开的问题

以下引用word-break的说明, 注意word-break 是IE5+专有属性

语法： word-break : normal | break-all | keep-all 参数： normal : 　依照亚洲语言和非亚洲语言的文本规则，允许在字内换行 break-all : 　该行为与亚洲语言的normal相同。也允许非亚洲语言文本行的任意字内断开。该值适合包含一些非亚洲文本的亚洲文本 keep-all : 　与所有非亚洲语言的normal相同。对于中文，韩文，日文，不允许字断开。适合包含少量亚洲文本的非亚洲文本 说明： 设置或检索对象内文本的字内换行行为。尤其在出现多种语言时。 对于中文，应该使用break-all 。

# HTML5

新的选择器

## querySelector

//var oDiv = document.querySelector('[title=hello]');

var oDiv = document.querySelector('.box'); //只能选择一组中的第一个元素

//alert( oDiv.length );

oDiv.style.background = 'red';

## querySelectorAll

var aDiv = document.querySelectorAll('.box'); //获取一组元素

alert( aDiv.length );

## getElementsByClassName

var aDiv = document.getElementsByClassName('box');

alert( aDiv.length );

# 获取class列表属性

var oDiv = document.getElementById('div1');

//alert( oDiv.classList ); //类似与数组的对象

//alert( oDiv.classList.length ); //3 获取长度

//oDiv.classList.add('box4'); 添加class

//oDiv.classList.remove('box2'); 删除class

oDiv.classList.toggle('box2'); 没有就添加class有就删除

# json新方法

//eval : 可以解析任何字符串变成JS

//parse : 只能解析JSON形式的字符串变成JS (安全性要高一些)

/\*var str = 'function show(){alert(123)}';

eval(str);

show();\*/

/\*var str = 'function show(){alert(123)}';

JSON.parse(str);

show();\*/ 不能运行show函数

/\*var str = '{"name":"hello"}'; //一定是严格的JSON形式 属性名加引号

var json = JSON.parse(str);

alert( json.name );\*/

var json = {name : "hello"};

var str = JSON.stringify(json); 解析json变为字符串 解析出来的也是严格形式

JSON.stringify() 还可以接受另外两个参数，第一个参数可以是数组，或函数，如果是数组，则生成的字符串中只会包含数组中出现的项，如果是函数，则函数包含两个参数，第一个是属性名，第二个是属性值，属性名必须是字符串，可以在函数中通过属性名操作属性值并返回，如果返回的是undefined，则忽略这个属性。第二个参数表示是否在生成的字符串中保留缩进 ，可以是数字，代表缩进的空格数，最大10，也可以是字符串，代表用字符串来缩进，最大10个字符 。

JSON. stringify (jsontext,function( key,value ){

Switch(key){

Case “authro”:

Return value.join(“,”);

Case “year”:

Return 5000;

Case “edition”:

Return undefined; // 此时edition属性将被忽略，生成的字符串中将没有这个属性

Default:

Return value; // 一定要有default

}

});

JSON.parse() 也可以接收第二个参数，该参数是一个函数，将在每一个键值对上调用

函数的用法和作用与JSON. Stringify 相似，同样需要有返回值，如果返回值是undefined，则删除该键值对

自定义序列化： toJSON()

Var book = {

“title” : “bookkk”,

“year”: 2006,

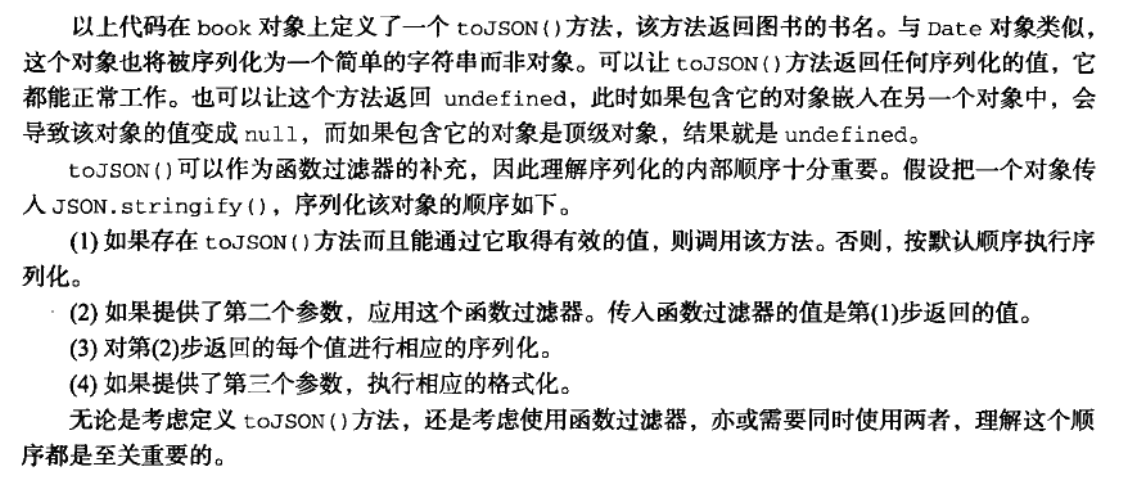
“edition” : 3,

toJSON : function(){

return this.title;

}

}



alert( str );

应用：深度复制json

var a = {

name : { age : 100 }

};

var str = JSON.stringify(a);

var b = JSON.parse(str);

# 自定义属性dataset

var oDiv = document.getElementById('div1');

//alert( oDiv.dataset.miaov );

alert( oDiv.dataset.miaovAll );

<body>

<div id="div1" data-miaov="妙味" data-miaov-all="妙味课堂">div</div>

//自定义属性前要加 data- 才能识别

</body>

# 延迟加载和异步加载

延迟加载，等页面加载完再加载js，相当于在window.onload前

<script src="a.js" defer="defer"></script>

<script src="b.js" defer="defer"></script>

<script src="c.js" defer="defer"></script>

和页面同时加载

<script src="a.js" async ="async"></script>

<script src="b.js" async ="async"></script>

<script src="c.js" async ="async"></script>

# 历史管理

//触发历史管理 : 1.通过跳转页面 2.hash onhashchange：改变hash值来管理 3.pushState

var oInput = document.getElementById('input1');

var oDiv = document.getElementById('div1');

var json = {};

oInput.onclick = function(){

var num = Math.random();

var arr = randomNum(35,7);

json[num] = arr;

oDiv.innerHTML = arr;

window.location.hash = num;

};

window.onhashchange = function(){

oDiv.innerHTML = json[window.location.hash.substring(1)];

};

function randomNum(iAll,iNow){

var arr = [];

var newArr = [];

for(var i=1;i<=iAll;i++){

arr.push(i);

}

for(var i=0;i<iNow;i++){

newArr.push( arr.splice( Math.floor(Math.random()\*arr.length) ,1) );

}

return newArr;

}

# 拖放事件

draggable ：

设置为true，元素就可以拖拽了

拖拽元素事件 : 事件对象为被拖拽元素

dragstart , 拖拽前触发

drag ,拖拽前、拖拽结束之间，连续触发

dragend , 拖拽结束触发

目标元素事件 : 事件对象为目标元素

dragenter , 进入目标元素触发，相当于mouseover

dragover ,进入目标、离开目标之间，连续触发

dragleave , 离开目标元素触发，相当于mouseout

drop , 在目标元素上释放鼠标触发

var aLi = document.getElementsByTagName('li');

var oDiv = document.getElementById('div1');

var i = 0;

for(var i=0;i<aLi.length;i++){

aLi[i].ondragstart = function(){

this.style.background = 'green';

};

aLi[i].ondrag = function(){ //开始与结束连续触发

document.title = i++;

};

aLi[i].ondragend = function(){

this.style.background = 'yellow';

};

}

oDiv.ondragenter = function(){

this.style.background = 'blue';

//要想触发drop事件，就 必须在dragenter 和 dragover当中阻止默认事件

ev.preventDefault();

};

oDiv.ondragover = function(ev){

//enter和leave之间连续触发

//要想触发drop事件，就 必须在dragenter 和 dragover当中阻止默认事件

//document.title = i++;

ev.preventDefault();

};

oDiv.ondragleave = function(){

this.style.background = 'red';

};

oDiv.ondrop = function(){

alert(123);

};

<body>

<ul>

<li draggable="true">a</li>

<li draggable="true">b</li>

<li draggable="true">c</li>

</ul>

<div id="div1"></div>

</body>

解决火狐下的问题

必须设置dataTransfer对象才可以拖拽除图片外的其他标签

dataTransfer对象

setData() : 设置数据 key和value(必须是字符串)

getData() : 获取数据，根据key值，获取对应的value

dataTransfer对象\_2

effectAllowed

effectAllowed : 设置光标样式(none, copy, copyLink, copyMove, link, linkMove, move, all 和 uninitialized)

setDragImage

三个参数：指定的元素，坐标X，坐标Y

files

获取外部拖拽的文件，返回一个filesList列表

filesList下有个type属性，返回文件的类型

for(var i=0;i<aLi.length;i++){

aLi[i].ondragstart = function(ev){

var ev = ev || window.event;

ev.dataTransfer.setData('name','hello');

ev.dataTransfer.effectAllowed = 'link';

ev.dataTransfer.setDragImage(oImg,0,0);

};

aLi[i].ondragend = function(){

this.style.background = 'yellow';

};

}

oDiv.ondragenter = function(){

this.style.background = 'blue';

};

oDiv.ondragover = function(ev){

//enter和leave之间连续触发

//要想触发drop事件，就 必须在dragover当中阻止默认事件

//document.title = i++;

ev.preventDefault();

};

oDiv.ondragleave = function(){

this.style.background = 'red';

};

oDiv.ondrop = function(ev){

};

FileReader(读取文件信息)

readAsDataURL

参数为要读取的文件对象，将文件读取为DataUrl

onload

当读取文件成功完成的时候触发此事件

this. result , 来获取读取的文件数据，如果是图片，将返回base64格式的图片数据 可以设置img.src = this.result

var oDiv = document.getElementById('div1');

oDiv.ondragenter = function(){

this.innerHTML = '可以释放啦';

};

oDiv.ondragover = function(ev){

ev.preventDefault();

};

oDiv.ondragleave = function(){

this.innerHTML = '将文件拖拽到此区域';

};

oDiv.ondrop = function(ev){

ev.preventDefault();

var fs = ev.dataTransfer.files;

//alert(fs.length);

//alert( fs[0].type );

var fd = new FileReader();

fd.readAsDataURL( fs[0] );

fd.onload = function(){

alert( this.result );

};

};