# Css3笔记

目录

[Css3笔记 1](#_Toc8309859)

[预处理器 pre-processor 3](#_Toc8309860)

[后处理器 post-processor 3](#_Toc8309861)

[postCss+插件(充分体现扩展性) 3](#_Toc8309862)

[预处理器与后处理器的区别 3](#_Toc8309863)

[Relationship Selectors(关系选择器,用的比较少) 4](#_Toc8309864)

[E + F 4](#_Toc8309865)

[E ~ F 4](#_Toc8309866)

[E > F 4](#_Toc8309867)

[Attribute Selectors(属性选择器,几乎不用) 5](#_Toc8309868)

[E [attribute~=value] 5](#_Toc8309869)

[E[attribute|=value] 5](#_Toc8309870)

[E[attribute^=value] 5](#_Toc8309871)

[E[attribute$=value] 5](#_Toc8309872)

[E[attribute\*=value] 5](#_Toc8309873)

[Pseudo-Element Selectors(伪元素选择器) 5](#_Toc8309874)

[E::placeholder 5](#_Toc8309875)

[E::selection 6](#_Toc8309876)

[Pseudo-Classes Selectors(伪类选择器) 6](#_Toc8309877)

[E:not(s) 6](#_Toc8309878)

[:root 6](#_Toc8309879)

[E:target 6](#_Toc8309880)

[E:first-child 7](#_Toc8309881)

[E:last-child 7](#_Toc8309882)

[E:only-child 8](#_Toc8309883)

[E:nth-child(n) 8](#_Toc8309884)

[E:nth-last-child(n) 9](#_Toc8309885)

[E:first-of-type 9](#_Toc8309886)

[E:last-of-type 10](#_Toc8309887)

[E:only-of-type 10](#_Toc8309888)

[E:nth-of-type(n) 11](#_Toc8309889)

[E:nth-last-of-type(n) 11](#_Toc8309890)

[E:empty 12](#_Toc8309891)

[E:checked 12](#_Toc8309892)

[E:enabled 13](#_Toc8309893)

[E:disabled 13](#_Toc8309894)

[E:read-only 13](#_Toc8309895)

[E:read-write 13](#_Toc8309896)

[border-radius(边框半径) 13](#_Toc8309897)

[box-shadow 15](#_Toc8309898)

[border-image 16](#_Toc8309899)

[border-image-source 16](#_Toc8309900)

[border-image-slice 16](#_Toc8309901)

[border-image-width 17](#_Toc8309902)

[border-image-outset 17](#_Toc8309903)

[border-image-repeat 17](#_Toc8309904)

[background 17](#_Toc8309905)

[background-image 17](#_Toc8309906)

[background-origin 18](#_Toc8309907)

[background-position 19](#_Toc8309908)

[background-clip 19](#_Toc8309909)

[background-attachment 19](#_Toc8309910)

[background-size 19](#_Toc8309911)

[background-repeat 19](#_Toc8309912)

[currentColor 20](#_Toc8309913)

[text 20](#_Toc8309914)

[text-shadow 20](#_Toc8309915)

[font-face 21](#_Toc8309916)

[box 21](#_Toc8309917)

[box-sizing 21](#_Toc8309918)

[overflow 21](#_Toc8309919)

[resize 22](#_Toc8309920)

[display 22](#_Toc8309921)

[flex-direction 22](#_Toc8309922)

[flex-wrap 22](#_Toc8309923)

[justify-content 22](#_Toc8309924)

[align-items 23](#_Toc8309925)

[align-content 23](#_Toc8309926)

[order 24](#_Toc8309927)

[align-self 24](#_Toc8309928)

[Flex:flex-grow 24](#_Toc8309929)

[flex-shrink 24](#_Toc8309930)

[flex-basis 24](#_Toc8309931)

[动画 26](#_Toc8309932)

[transition 26](#_Toc8309933)

[animation 26](#_Toc8309934)

[animation-timing-function 27](#_Toc8309935)

[animation-iteration-count 27](#_Toc8309936)

[animation-direction 27](#_Toc8309937)

[animation-play-state 27](#_Toc8309938)

[animation-fill-mode 27](#_Toc8309939)

[transform 28](#_Toc8309940)

[rotate 29](#_Toc8309941)

[rotate3d 29](#_Toc8309942)

[scale() 29](#_Toc8309943)

[skew() 29](#_Toc8309944)

[translate() 29](#_Toc8309945)

[perspective() 29](#_Toc8309946)

[matrix() 29](#_Toc8309947)

[transform-origin 29](#_Toc8309948)

[perspective 30](#_Toc8309949)

[perspective-origin 30](#_Toc8309950)

[transform-style 30](#_Toc8309951)

[Css3性能优化 30](#_Toc8309952)

[响应式网页开发 31](#_Toc8309953)

[设置视口 31](#_Toc8309954)

[响应式布局实现 32](#_Toc8309955)

[媒体查询 32](#_Toc8309956)

[单位值 35](#_Toc8309957)

## 预处理器 pre-processor

Less/sass

cssNext(插件):用来实现一些未来标准的（未完全在各大浏览器实现的功能）

按照语法规则写代码，根据符合预处理器代码规则的代码生成css，去执行。

## 后处理器 post-processor

autoprefixer是后处理器的一个插件。

写完css代码之后补全css代码

## postCss+插件(充分体现扩展性)

postCss只是一个工具，不是后处理器的代表；不能划分为某一个处理器。后面加上插件，插件实现了什么功能，就代表着postCss+插件实现了什么功能。

用js实现的css的抽象语法树AST(Abstract Syntax Tree)。剩下的事留给后人去做。

## 预处理器与后处理器的区别

1. 执行逻辑或者编写形式不同。
2. 后处理器比预处理器对未来可维护性好一些。

## Relationship Selectors(关系选择器,用的比较少)

E + F:下一个满足条件的兄弟元素节点

例：选中的是 class为demo的div的下一个标签为p的兄弟元素。



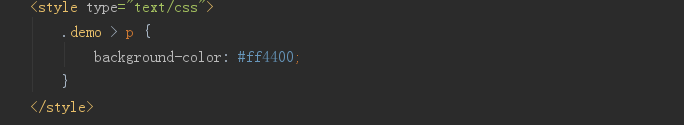
E ~ F:e后面多个满足条件的兄弟元素节点

例：class为demo的div后面所有标签为p的兄弟元素



E > F:选中E的直接子元素F

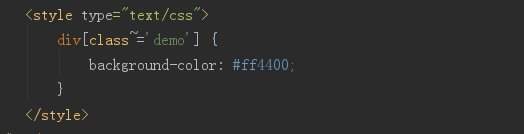
例:class为demo的div的标签为p的子元素。



## Attribute Selectors(属性选择器,几乎不用)

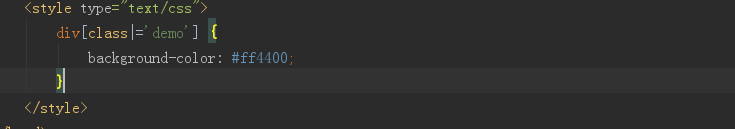
E [attribute~=value]: 选择属性名是attribute,属性值里存在独立的value的元素。

例:属性名是class，属性值存在独立的demo的div。



E[attribute|=value]: 选择一个以value开头或者以value-开头的元素

例:选择以demo或者demo-开头的class的div。



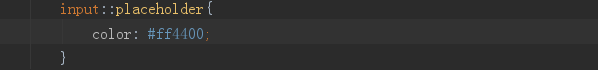
E[attribute^=value]: 以value开头的元素

E[attribute$=value]: 以value结尾的元素

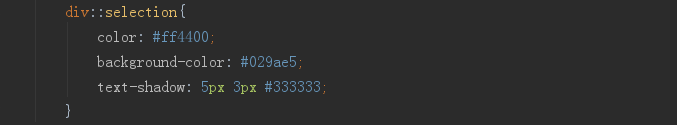
E[attribute\*=value]: 只要存在value的元素就可以

## Pseudo-Element Selectors(伪元素选择器)

E::placeholder:输入框提示改变颜色只能改变字体颜色



E::selection:改变字体选中的样子；只能设置color、background-color、text-shadow三个属性。（兼容性很好）

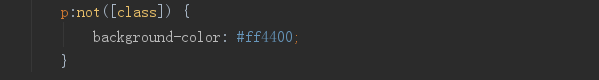


## Pseudo-Classes Selectors(伪类选择器)

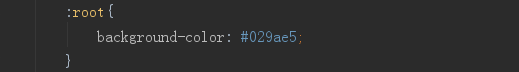
伪类选择器叫做被选中的元素的一种状态

E:not(s): 选择非 s 元素的每个元素。

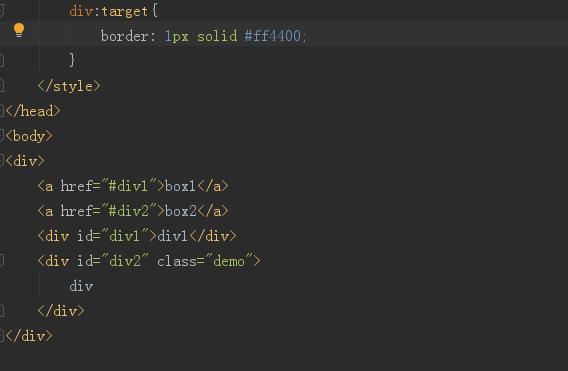
例:选中没有class的p。



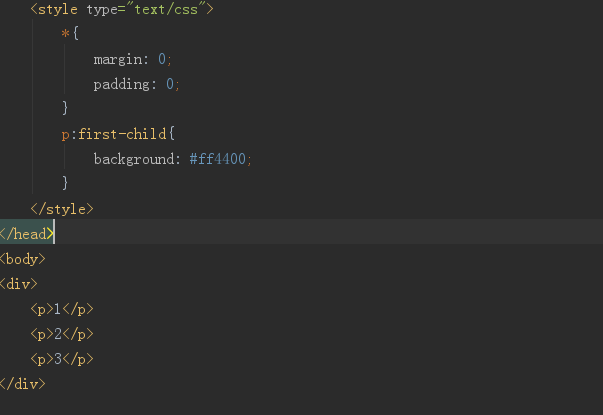
:root:选择根节点，在HTML中与html标签选择器是相等的。Root根标签选择器;root包含html



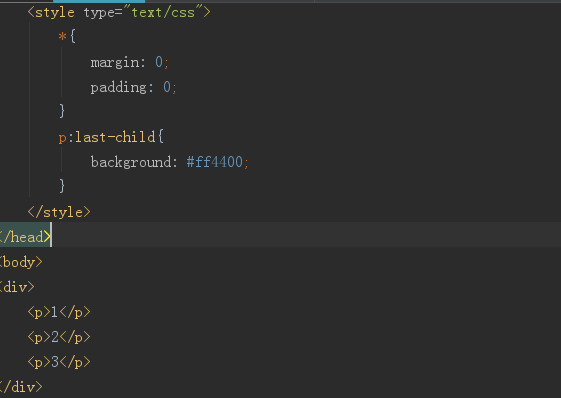
E:target:选中被选中的目标(被锚点标记===location.has之后)



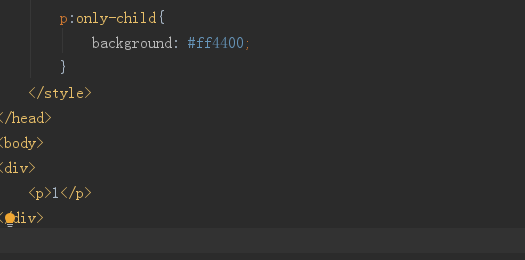
E:first-child:第一个子元素，考虑其它元素对选择结果的影响



E:last-child:最后一个子元素，考虑其它元素对选择结果的影响



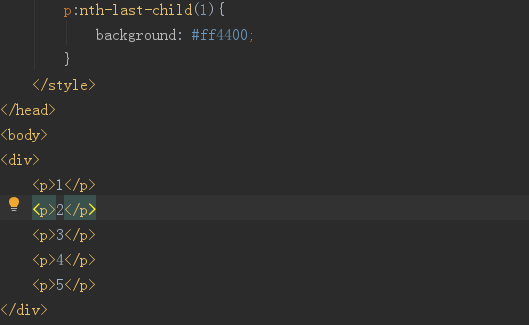
E:only-child:只有一个子元素，考虑其它元素对选择结果的影响



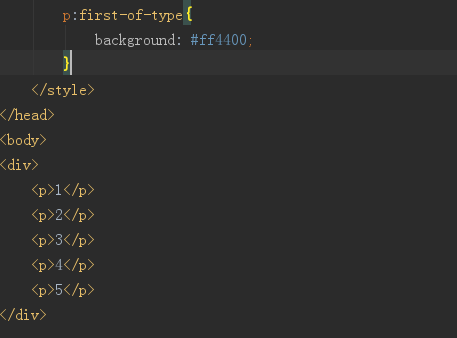
E:nth-child(n):选择父级下的第几个子元素n从0开始，css从1开始；考虑其它元素对选择结果的影响



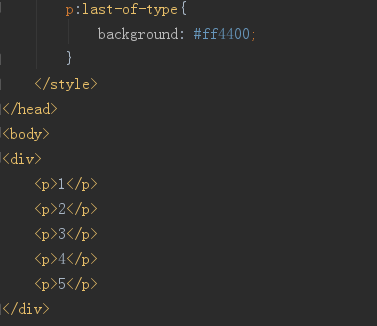
E:nth-last-child(n): 选择父级下的第几个子元素,从最后一个元素开始计数。考虑其它元素对选择结果的影响



E:first-of-type:选择第一个类型的子元素，不考虑其它元素影响



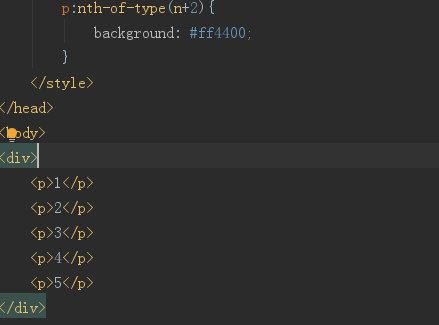
E:last-of-type:选择最后一个类型的子元素



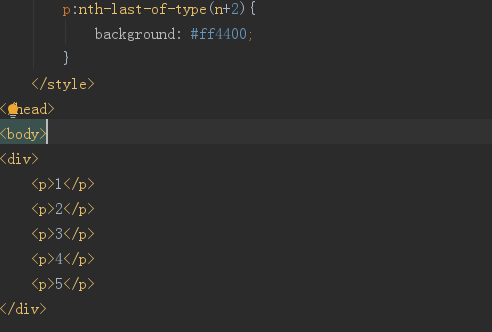
E:only-of-type:选择只有一个类型的子元素



E:nth-of-type(n):查找相同类型的子元素



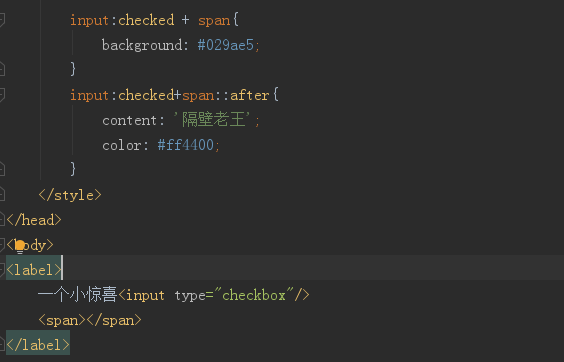
E:nth-last-of-type(n): 选择父级下的第几个相同类型的子元素,从最后一个元素开始计数。



E:empty:选择没有元素的节点



E:checked:选取选中元素



E:enabled:选中可以编辑的input

E:disabled:选中不可编辑的input

E:read-only:选中只读的元素。

E:read-write:选中可读可写的元素。

## border-radius(边框半径)

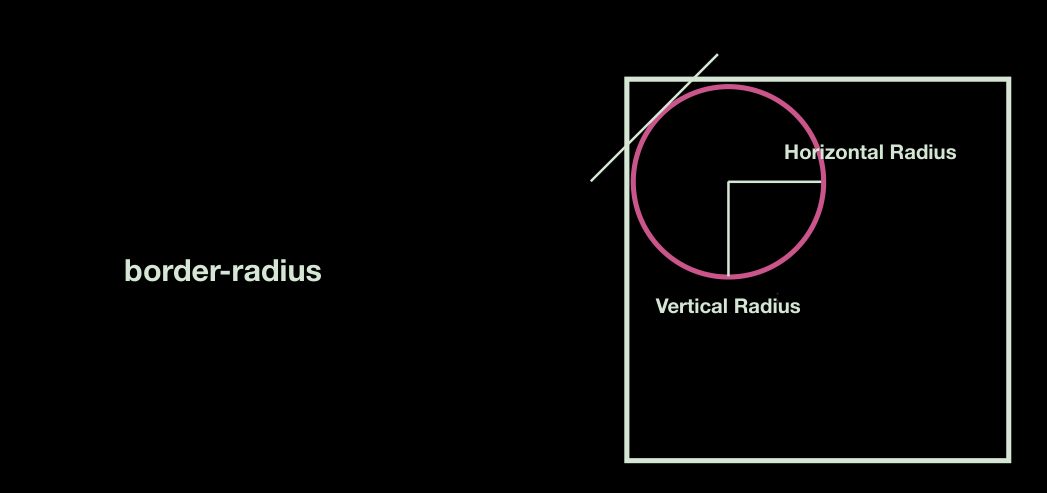
分四个方向设置边框半径：左上、右上、左下、右下。

两个值第一个代表左上和右下；第二个值代表右上和左下。

三个值第一个代表左上；第二个值代表右上和左下；第三个值代表右下。

Border-radius另一种写法: border-radius: 10px 20px 30px 40px/10px 20px 30px 40px;可以分别设置某个方向的水平半径和垂直半径。

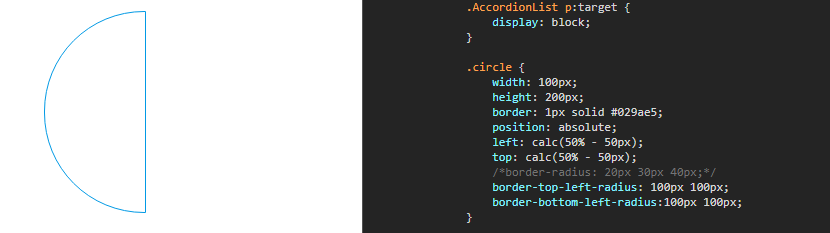
Border-radius原理:构建出一个以水平半径和垂直半径为半径的圆，用正方形的边和圆的边进行切割，剩下的那个角就是所得圆角。



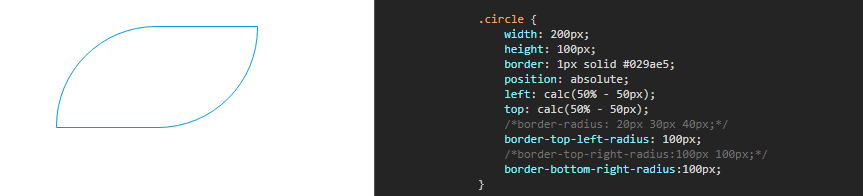
border-top-left-radius:单独设置左上角，可以传一个或两个值；分别控制水平方向半径和垂直方向半径。当水平半径和垂直半径等于正方形的宽和高时会得到一个四分之一圆。



利用border-radius做半圆



利用border-radius做叶子形



## box-shadow

给盒子加阴影的；可以加内阴影或外阴影。

参数： 1阴影水平偏移量

2阴影垂直偏移量

3阴影的模糊范围:基于边框的原来位置，向边框两边同时去模糊。模糊值越大，图形越虚，越可以透出底色。

4传播的距离（可填可不填），spread其实就是扩大的意思;在水平和垂直方向同时增加；在四个方向同时增加阴影的大小。

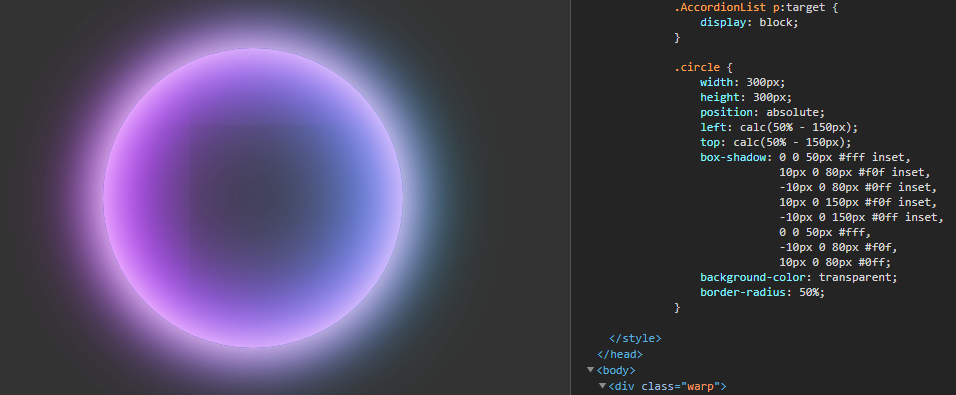
内阴影时，增加往里增加，减少是往外逻辑上减少;

5阴影颜色

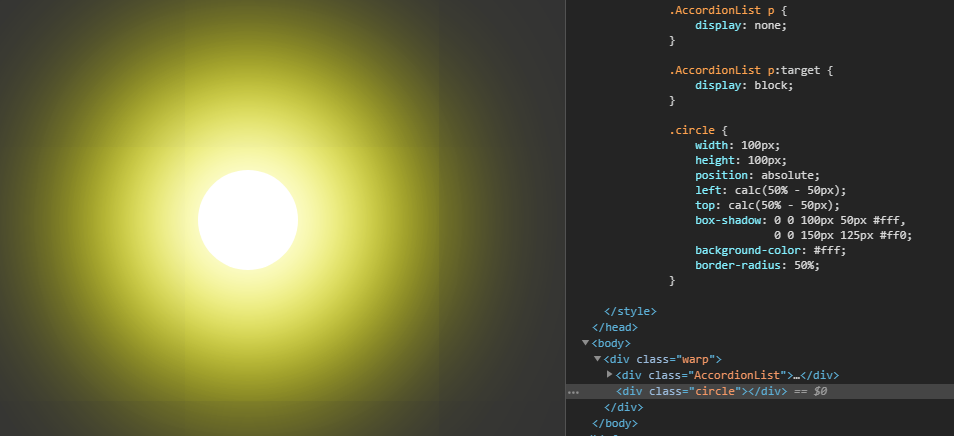
6 inset（内阴影可以写在第一个参数也可以写在最后一个参数）。

阴影在文字的下面在背景颜色的上面。

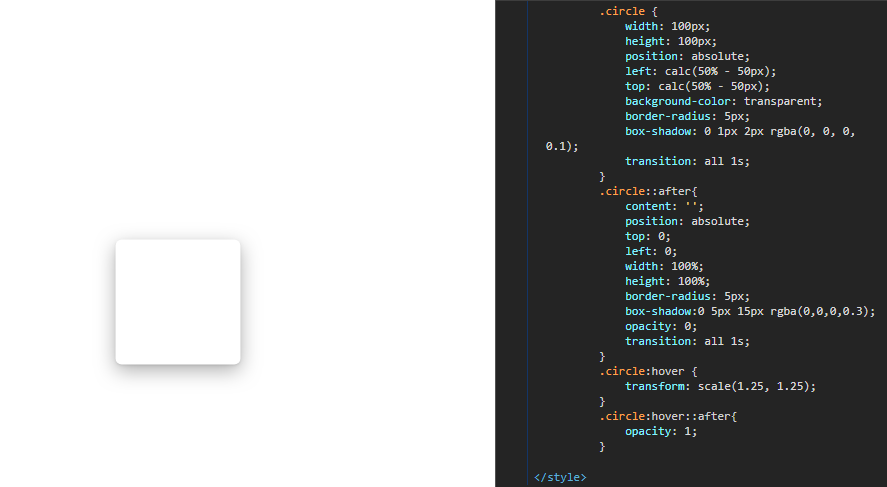
例一：



例二:



例三:

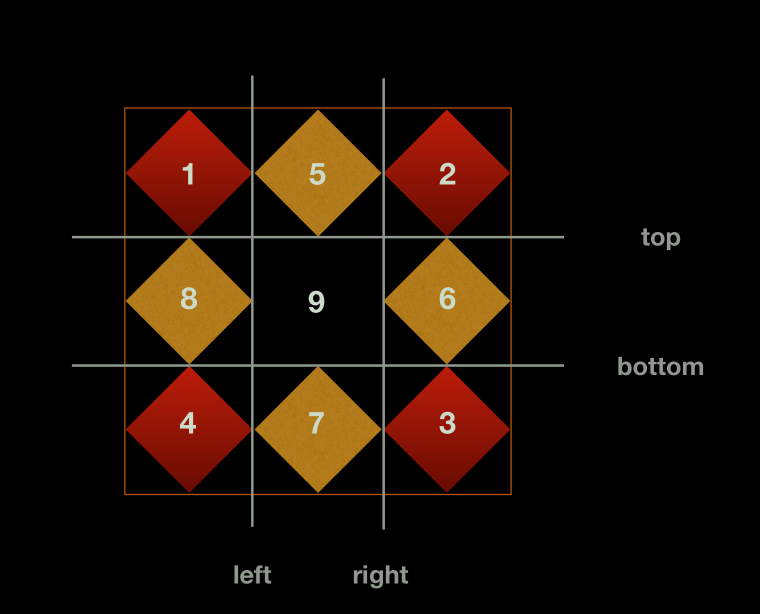


## **border-image**

border-image可以给border加上背景颜色，可以用image填充，也可以调出渐变色。

border-image-source:可以填图片地址ur()，也可以填线性渐变liner-gradient();使用border-image-source时，border-color失效。和border-image-slice配合使用。

border-image-slice:只能填数字和百分比，不能填px,但是数字等于px；有五个参数，前四个对应上右下左四条线的位置来切割背景图片,将背景图片切割为九部分;还可以填一个fill,fill可以把内容区也填充，一般情况下不适用fill。



border-image-width:默认值为1，填数字等于数字\*border的宽度;填auto会取border-image-slice的值并加上px。

border-image-outset:背景图片往外延伸，不可用填负值。（不常用）

border-image-repeat: stretch(默认值拉伸) round(当不满足添加块时拉伸,当区域增加到一半块的宽度时增加一个) repeat(平铺) space(兼容性不怎么好)先拿空白拉伸，到一定程度添加一个块。也可以同时填两个值表示水平和垂直方向。

border-image简写方式: source slice repeat

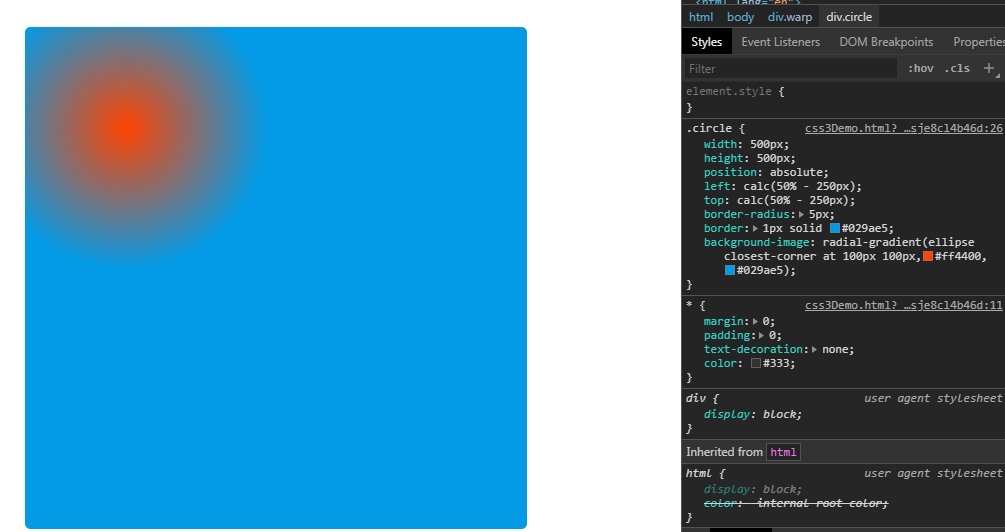
## background

background-image:url(),url();可以写多个图片地址，应用场景为当第一张图片过大，加载过慢时用第二张较小的图片显示，等第一张图片加载完之后再显示第一张。

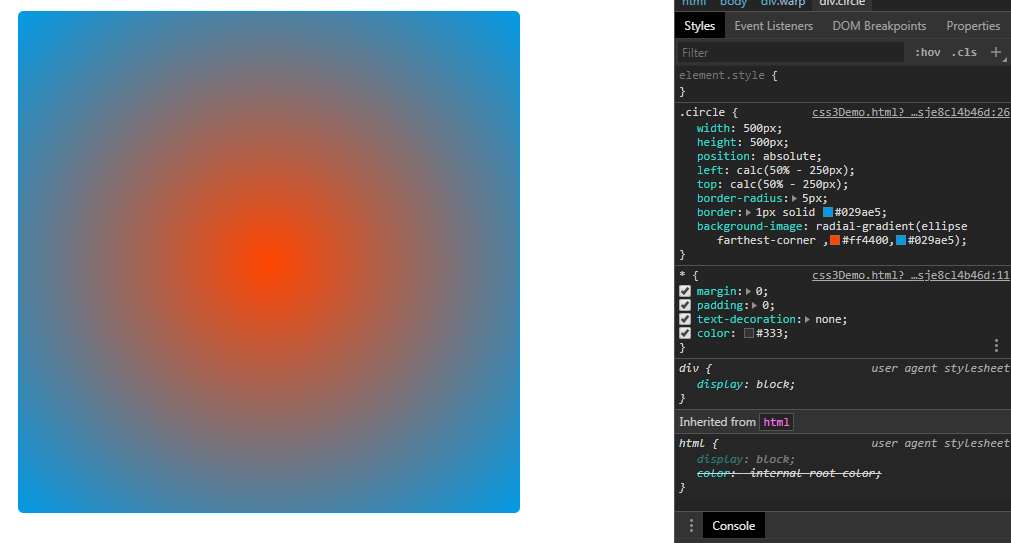
Linear-gradient:可以设置线性渐变第一个值可以填方向与角度例如to left,后面的值除了可以填颜色还可以同时填终止位置例如#029ae5 20px.

radial-gradient:径向渐变，第一个值可以设置圆心的位置可以不填例如:circle at 100px 100px 或 circle at right bottom 也可以写ellipse（椭圆）可以填放射半径closest-corner(最近的角) closest-side(最近的边) farthest-corner(最远的角) farthest-side(最远的边)例如: ellipse closest-corner at 100px 100px,#ff4400,#029ae5

最近的角:



最远的角:



background-origin:border-box:从border开始渲染背景图片。

padding-box:是默认值，从padding开始渲染背景图片

content-box:从内容开始渲染背景图片

background-position:origin值是什么position就相对于谁定位。

background-clip:配合origin一起使用

border-box: 是默认值，从border停止渲染背景图片。(没有什么用)

padding-box:从padding停止渲染背景图片

content-box:从内容停止渲染背景图片

text:拿文字的部分截取背景图片。只在webkit下好使



background-attachment:改变背景图片定位属性。

scroll:默认值，相对于容器定位，相对于视口定位，是跟着容器走的，容器不动，背景图片就不动

local:相对于容器的内容区定位，相当于页面定位的absolute;当做某一块的背景图片。

fixed:相对于真正的可视区视口定位的，当超出容器范围，背景图片不显示。

background-size:

cover用一张图片完整的填充满容器，而不改变图片原始比例状态。存在超出的风险。一条边对齐，另一条边大于等于容器边大小。

contain:在不改变原有图片比例下，让容器包含一张完整的图片。存在repeat的风险。

都是让图片的一条边与容器的一条边对齐，然后改变图片的另一条边的比例。一条边对齐，另一条边小于等于容器边大小。

background-repeat:

no-repeat:不平铺

round:拉伸

space:先拿空白填充，到一定距离添加图片

可以同时写 round 和 space

repeat-x:沿x轴平铺

repeat-y:沿y轴平铺

repeat系列不能两个值同时写，也不能和round或space组合使用。

currentColor:当样式中不设置border-color时,css1、css2会计算color的值赋给border-color;css3 使border-color=currentColor(中转颜色);currentColor计算border-color的值。

## text

text-shadow:给文字添加阴影。

参数：1阴影水平偏移量

2阴影垂直偏移量

3阴影的模糊范围

4阴影颜色

浮雕效果:



镂刻效果:



越往后越虚化，越虚化颜色越浅，越容易融入到背景当中。





background-clip:text和text-shadow一起使用时,background-clip会把文字变为背景的一部分,背景在内容的下面;文字阴影算内容的一部分,所以阴影在文字内容的上面。解决方案给text-shadow颜色设置透明度。

描边效果:仅在webkit内核的浏览器中有效



font-face:从网上引用字体包。

格式: @font-face{

font-family:’自定义字体名字’;

src:url(字体网络地址);

}

用处:1.单独一些文字,字体要特殊字体处理可以引用字体包,如果字体很少的话也可以让美工给图片不用下载字体包。

2.处理特殊符号。

## box

一个盒模型功能的强弱与否直接决定了在布局上是否简洁;布局是否简洁直接影响到后期编程是否复杂。

W3C Standards mode(w3c标准盒子): boxWidth = width + padding(2) + border(2)

IE 6 Quirks mode(Ie6混杂模式盒子):boxWidth = width

box-sizing:border-box(混杂盒模型)

需求一:宽度不固定，但内边距固定。

overflow: visible(默认值) hidden(超出部分隐藏) scroll(超出部分显示滚动条) auto(内容超出显示滚动条,没超出不显示)。

也可以单独设置overflow,只要overflow-x或overflow-y有一个值设置的不是visible,另一个值自动改为auto。

resize:both(在水平、垂直方向让用户改变元素大小),必须配合overflow一起使用;不可以继承值,不太常用。

display:flex/inline-flex 弹性盒子

弹性盒子有两个轴:主轴和交叉轴;主轴可以发生改变。主轴和交叉轴不会因主轴发生改变而引发歧义。

设置display:flex的父容器可以设置的属性:

flex-direction: 设置主轴方向，最常用。

row:默认值 自左向右。主轴是水平方向。

row-reverse: 主轴是水平方向,自右向左。

clolumn:主轴是垂直方向,自上向下。当主轴方向发生改变时,子元素的排列顺序也会发生改变。

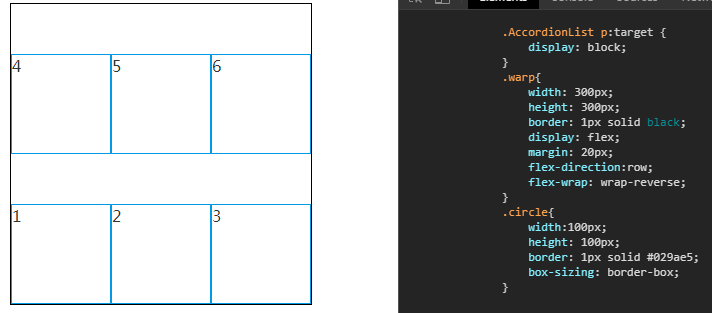
column-reverse:主轴是垂直方向，自下向上。

flex-wrap: 所有子元素的宽度超过父容器宽度是否换行。

nowrap:默认值,不换行,将子元素的宽度压缩。

wrap:换行

wrap-reverse:倒着换行,如下图所示



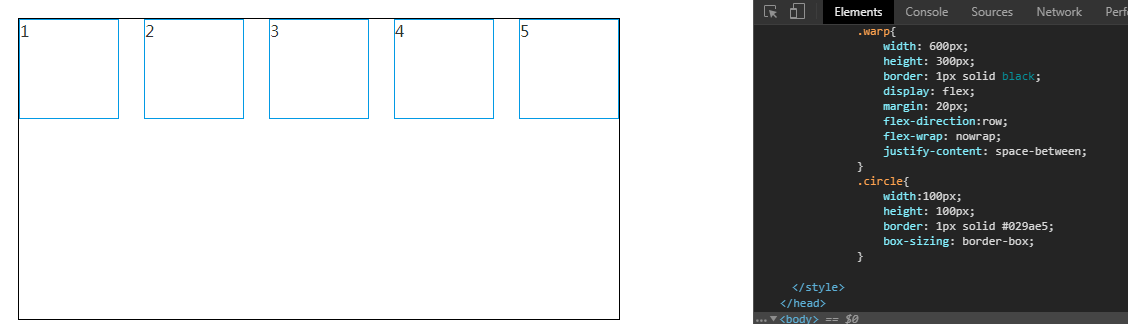
justify-content:基于主轴做一个对齐方式。

flex-start:默认值,方向自左向右。

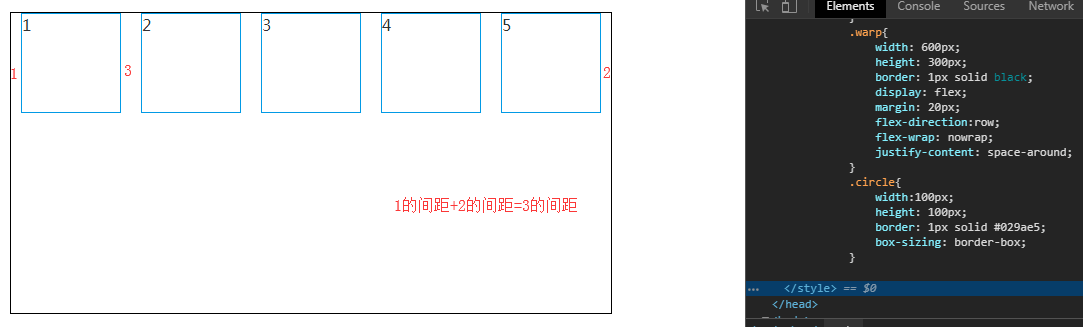
flex-end:主轴方向的另一侧对齐。

center:基于主轴的中间方向对齐。

space-between:向两边分布中间自适应



space-around:尽量让子元素之间的间距相等,两端的间距等于子元素的间距。



align-items:基于交叉轴做一个对齐方式。

stretch:默认值,当子元素没有设置高度时,会将子元素的高拉伸到父容器的高。

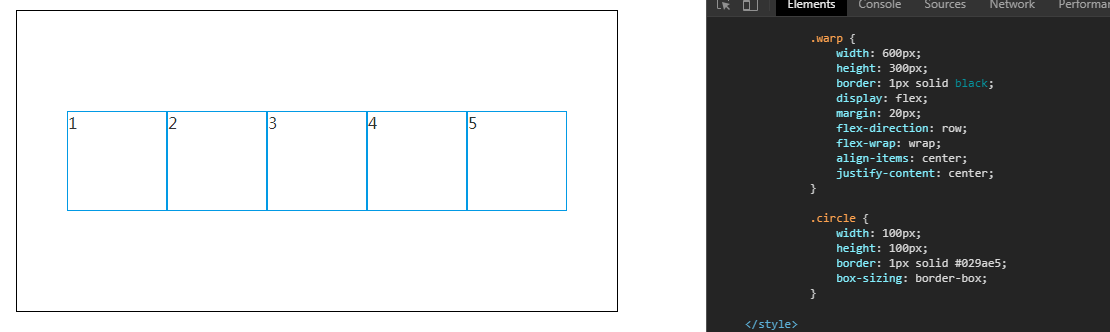
flex-start:交叉轴方向对齐。

flex-end:交叉轴尾方向对齐

center:交叉轴中间对齐。

baseline:基于文字底线对齐

单行子元素居中:align-items主要还是针对,单行元素来处理对齐方式的。



align-content:和align-items功能上基本一样,唯一的区别是align-content必须得作用到多行元素。基于交叉轴进行多行元素的位置的分配。

flex-start:交叉轴起始方向对齐。

flex-end:交叉轴尾方向对齐

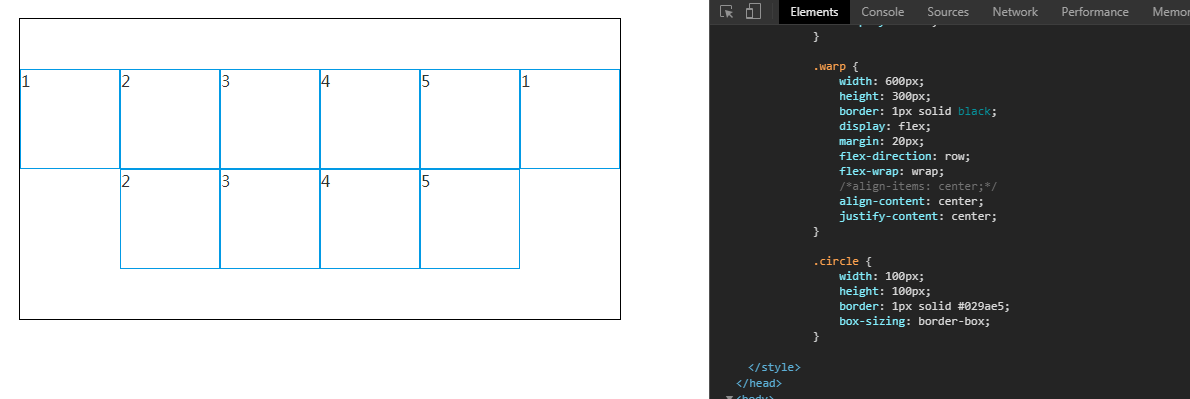
center:交叉轴中间对齐。

stretch:默认值。

space-between:上下占领，中间平均分配

space-around: 尽量让子元素之间的间距相等,两端的间距等于子元素的间距。

多行子元素居中:



设置display:flex的子元素可以设置的属性:

order:默认值0，作用类似z-index;值越小越在前面。

align-self:子项作为一个个体,基于交叉轴的分配状况。当父级和自身都设置交叉轴分布情况时，自身的优先级高于父级的。align-content优先级大于align-self。

Flex:flex-grow:默认值0,当主轴方向还有剩余空间时,flex-grow会根据自己的比例来分配剩余的空间。

flex-shrink:默认值1,当内容宽度超出父容器时,压缩子元素宽度,不加边框压缩公式: 子元素1真实内容区的大小\*比例+子元素2真实内容区的大小\*比例+……=加权值;

子元素1真实内容区的大小\*比例/加权值\*压缩值=自己要压缩的值。

flex-basis:原本是想取代width;flex-basis权重高于width；默认值是auto

flex-grow与flex-basis一起使用时,grow会以basis值为基础计算值,看剩余多少空间。

只设置basis或者basis大于等于width时元素的min-width等于basis的值;当不换行内容超过内容区时，会撑开内容区。

当basis和width同时设置时,并且basis小于width时,basis<countwidth<width；basis设置了容器宽度的最小值，width设置了容器宽度的最大值,盒子内容宽度只能在最小值和最大值之间伸缩。

无论什么情况下，被不换行内容撑开的容器，不会被压缩计算。

当子元素需要压缩计算时,可以加word-break: break-word使英文换行。

Flex: grow shrink basis(简写格式)

none关键字的计算值为: 0 0 auto

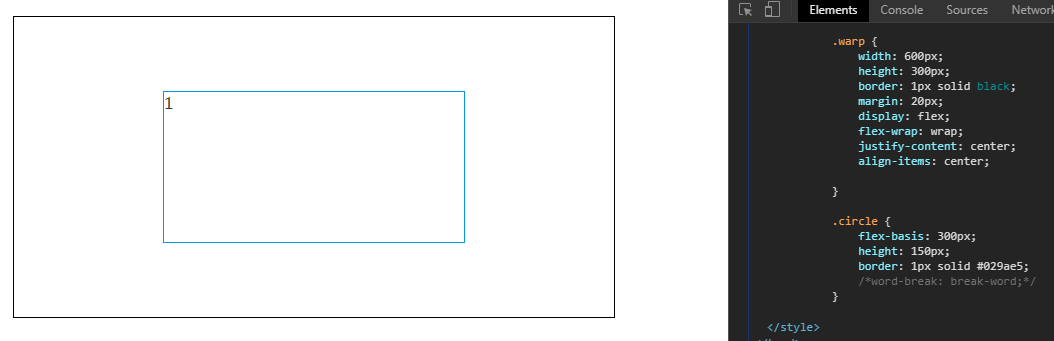
如果缩写「flex: 1」, 则其计算值为「1 1 0%」

如果缩写「flex: auto」, 则其计算值为「1 1 auto」

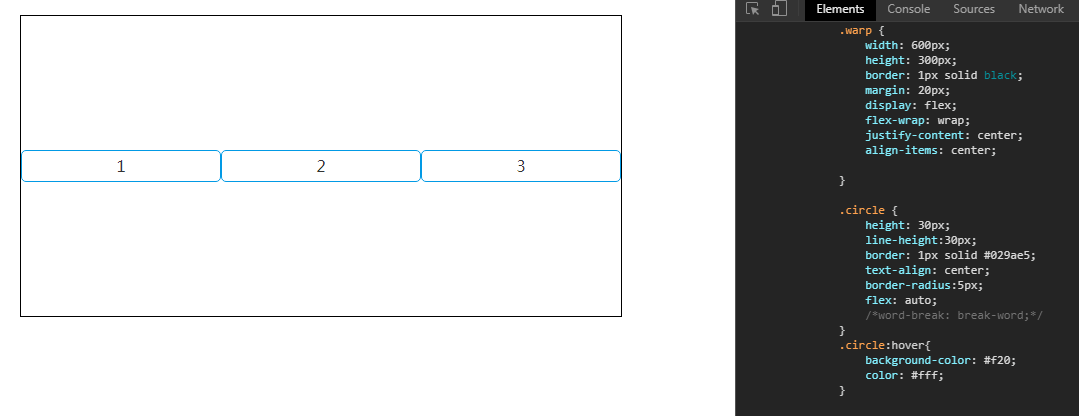
如果「flex: none」, 则其计算值为「0 0 auto」

如果「flex: 0 auto」或者「flex: initial」, 则其计算值为「0 1 auto」，即「flex」初始值

Flex作用:1.居中

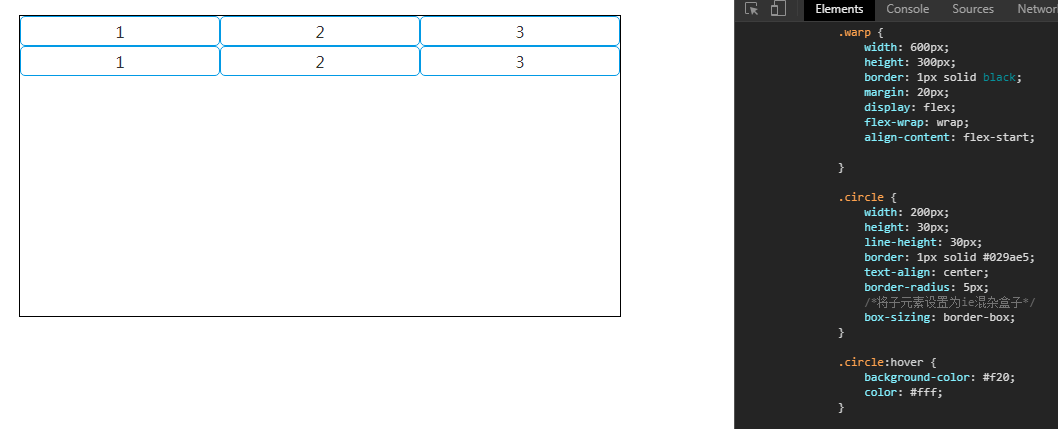


2.可动态增加导航栏

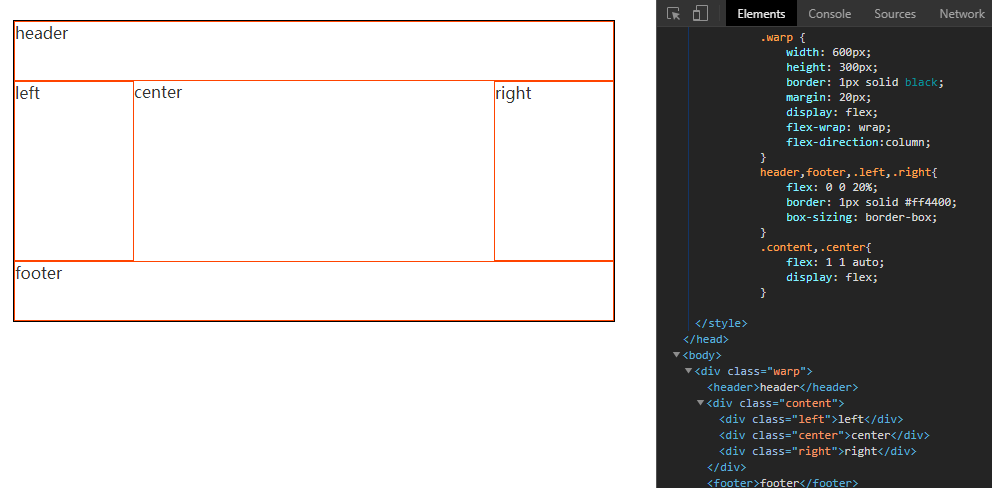


3.等分布局

4.流式布局



5.圣杯布局



## 动画

transition:(过渡动画)会监控状态是否发生变化，如果发生状态变化并且发生状态变化属性支持动画过渡，就以动画过渡的形式改变状态。

transition有四个值分别是

transition-property:监听到底要监听哪个属性的状态改变。默认值all

transition-duration:过渡动画几秒完成。默认值0

transition-timing-function:过渡动画运动状态的函数。默认值ease

transition-delay:等多少秒执行过渡动画相当于setTimeOut 默认0

cubic-bezier:贝塞尔曲线

animation:是实现css3最重要的一个属性,transiton实现的是从a状态变化到b状态的动画效果，再变到c状态是办不到的。Animation可以从a状态变到b状态再变到c状态一直变下去。

transition与animation的区别:transition实现一个状态改变的一段动画；animation是一个完整的多状态改变的动画。

用animation来实现状态的改变，必须把状态放在一个容器(@keyframes)里面记录下来并按顺序执行。

@keyframes 自定义名称{

关键帧用百分数定义{}

关键帧0%可以换成from

关键帧100%可以换成to

}

用逗号隔开可以使用多个keyframes。

运动状态是每一段的运动状态。Css3做不到每一段运动状态不一样。在纯css3中也无法实现不规律的曲线运动轨迹。

animation-timing-function:定义动画运动状态的函数; 默认值ease;

steps(<integer>[, [ start | end ] ]?)：接受两个参数的步进函数。第一个参数必须为正整数，指定函数的步数。第二个参数取值可以是start或end，指定每一步的值发生变化的时间点。第二个参数是可选的，默认值为end。end保留当前帧状态，直到这段动画时间结束。start保留下一帧状态，直到这段动画时间结束。

用end的缺陷可以用forwards属性来弥补缺陷。start不好弥补缺陷。

steps(1,start)= step-start

steps(1,end)=step-end

cubic-bezier(<number>, <number>, <number>, <number>)：特定的贝塞尔曲线类型，4个数值需在[0, 1]区间内

animation-iteration-count:动画执行多少次,默认1,可以填数字或infinite(无限的)。

animation-direction:定义运动方向

normal：正常方向(默认值)

reverse：反方向运行

alternate：动画先正常运行再反方向运行，并持续交替运行

alternate-reverse：动画先反运行再正方向运行，并持续交替运行

animation-play-state:控制运动的暂停或运动; running：运动paused：暂停。(兼容性有问题,最好不适用)

animation-fill-mode:设置动画运动完之后的状态

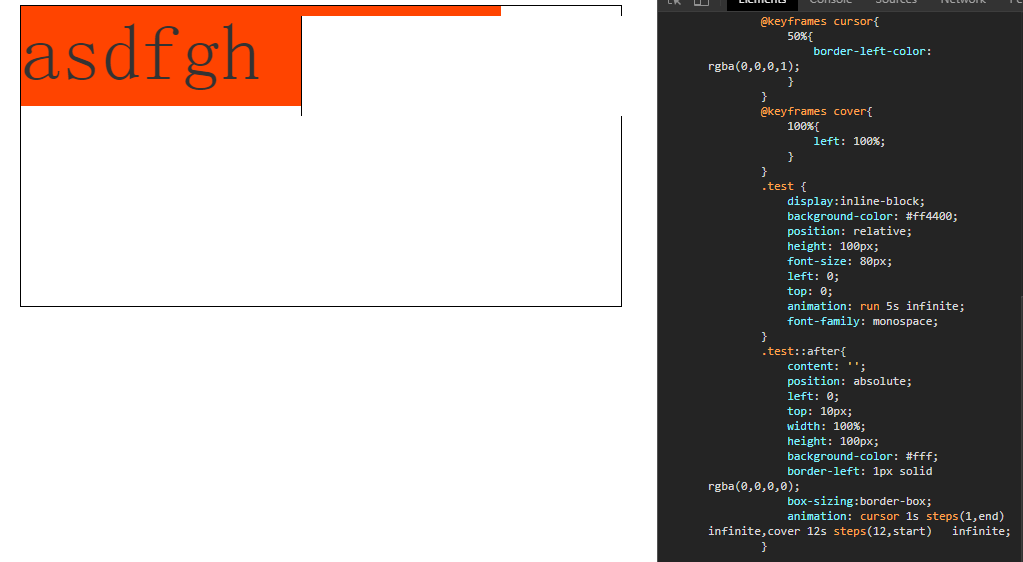
none：默认值。不设置对象动画之外的状态

forwards：在动画结束后,保存最后一帧的状态

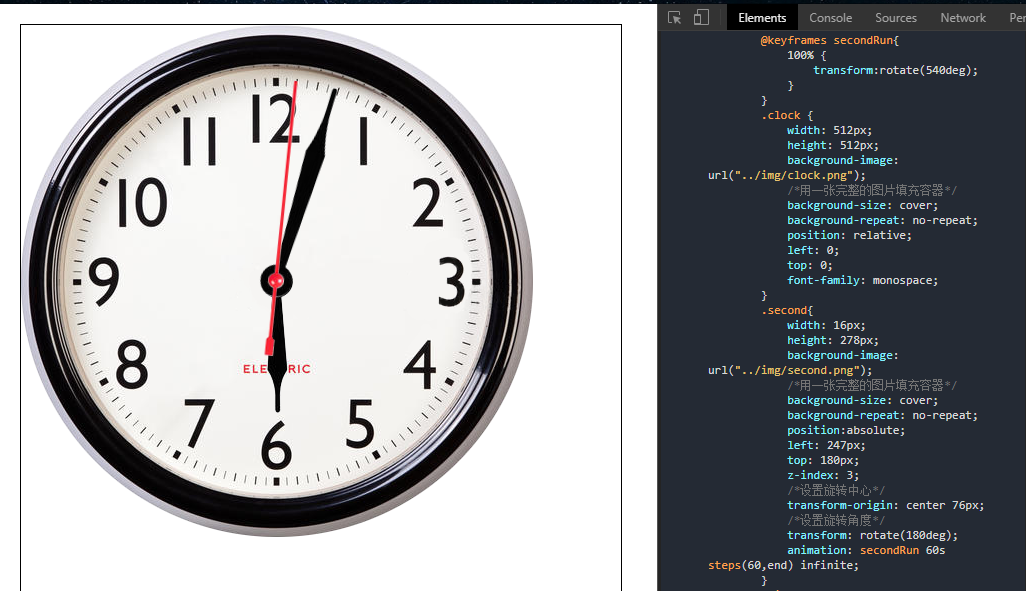
backwards：在动画开始前,保存第一帧的状态

both：在动画开始前,保存第一帧的状态; 在动画结束后,保存最后一帧的状态(用的比较多)

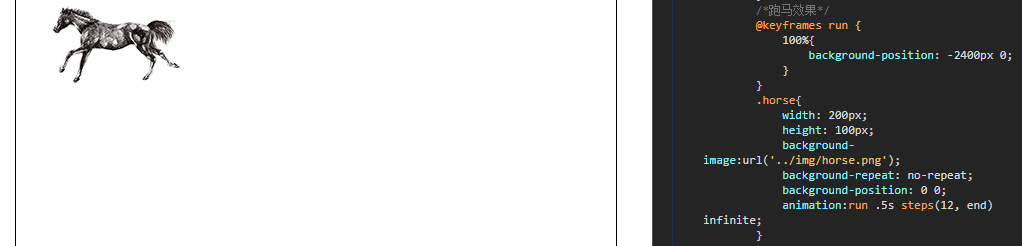
用steps实现的效果1打字效果:



时钟效果:



跑马效果:



transform:

rotate():2d变换写法;

rotateX、rotateY、rotateZ是3d变换

rotate3d():首先设置矢量，然后进行三D变换。设置矢量空间轴,然后沿着空间轴进行3d旋转。

参数:x,y,z,angle;x,y,z三个参数的比值会成为矢量的方向。矢量大小不重要，方向很重要。Angle以矢量方向去旋转

3d旋转，轴是跟着旋转的,在旋转的基础上再去旋转,就是围绕旋转之后的轴去旋转;基于这个前提旋转的顺序也不能改变,一旦改变，结构也会跟着改变。

旋转角度相同，顺序不同,得到的结果也不同。

什么是矢量/向量？矢量是一个有方向的度量长度的量

scale():伸缩.当值大于1代表扩张，小于1代表收缩;伸缩的不是元素本身，而是伸缩的此元素变化坐标轴的刻度;scale可以进行叠加操作;伸缩轴会随着旋转而旋转的，伸缩的影响会一直被保留。

skew():倾斜,倾斜的是坐标轴,不是元素本身。坐标轴不但被倾斜还被拉伸了。

translate():相对于自身移动,当不知道元素宽高时，可以利用translate和left、top一起使用使元素居中。

perspective():给每个子元素添加景深,只能写在transform属性最前面，写在后面有些浏览器不识别。

matrix():矩阵是实现一系列功能的底层原理，是transform选中的计算规则。

|a,c,e | |x| |ax+cy+e|

|b,d,f | \* |y| = |bx+dy+f|

|0,0,1| |1| |1|

矩阵的乘法的前提条件:第一个矩阵的列数等于第二个矩阵的行数

矩阵的乘法的规则: 1.第二个矩阵有多少列，得数就有多少列

2.第一个矩阵的行乘以第二个矩阵的列,按位相乘然后放到得数矩阵里。

平面变换的矩阵基础:

|1,0,e| |x| |x+e|

|0,1,f | \* |y| = |y+f|

|0,0,1| |1| |1|

X,Y是要变换图形上每一个点原始坐标。

Matrix(1,0,0,1,e,f)= translate(x,y);

Matrix(a,0,0,d,0,0)= scale(x,y);

transform-origin:设置变换中心，给谁设置就参照谁,可以填像素、百分比、代替值。可以填空间变换中心。

perspective:景深，默认值none，不允许填负值，只能填1~正无穷;景深相当于距离屏幕真正的有一个空间上的纵深，增加立体感。只有设置在父级身上,子元素才会生效。看到的是物体在屏幕上的投影。当没有translateZ时,元素就在屏幕上。景深可以叠加。

perspective-origin:控制从哪个角度去看,默认值center,center

transform-style:默认值flat，preserve-3d给父级加上之后，子元素的渲染空间就会变成3d渲染。

## Css3性能优化

像素:一个像素由三个像点组成,依照空间混色法排列;是相对单位。

物理像素:设备出厂时，像素的大小

Dpi:一英寸所能容纳的像素点数 1英寸=2.54厘米 96dpi≈2.54/100=0.25mm(毫米);打印机在一英寸的屏幕里面可以打印多少个墨点

Ppi:一英寸所能容纳的像素点数(点距数)

参照像素:为了在不同机器上显示图像有一样的体验效果; 96dpi 一臂之遥的视角去看，显示出的具体大小 标杆1/96\*英寸 参照像素=css像素=逻辑像素

设备像素比:dpr=物理像素/css像素

Css编程的逻辑像素方式，叫做逻辑屏幕。

屏幕的好坏不看分辨率看dpi 1920\*1080固定的宽高下，展示的像素点数

Crt显示屏：电子枪射在涂抹了荧光粉的屏幕上,高频持续电子束打在荧光粉上,荧光粉被点亮,持续打就持续亮;根据电子束强弱不同,每个被点亮的荧光粉的颜色强弱不同，最后混合出不同的颜色，形成一个像素点;多个像素点就成像了。

点距:从一个同色的点到另外一个同色点的距离。crt显示屏求点距的方法是几乎所有屏幕都通用的。

Cpu:擅长做一些基于逻辑运算

Gpu(显卡):擅长做一些图像绘制,适合处理高精度的浮点运算。

Reflow(重构):改变窗口大小 改变文字大小 内容的改变，输入框输入文字 激活伪类,如：hover 操作class属性 脚本操作dom 计算offsetWidth和offsetHeight 设置style属性

Repaint(重绘):如果只改变某个元素的背景色、文字颜色、边框颜色,不影响它周围或内部的属性；Repaint速度快于reflow。

触发GPU在另外一个层面处理操作 opacity transform:translate3d() translateZ()都可以触发 使用transform时,最好加上translateZ(0)触发GPU

Will-change:transform;是提前告诉浏览器针对这个属性进行复杂变化,将使用这个属性的元素单独拎到一个层上去。专业、标准的处理GPU加速问题。最好在运动之前告诉浏览器启动一个层，运动结束之后销毁。

浏览器刷新页面的频率:16.7mm

## 响应式网页开发

目的:只需要开发一套页面 让用户在不同的设备（大小或者分辨率如何变化）上看到的页面呈现效果是完美的

响应式网页设计或称自适应网页设计或称回应式网页设计/对应式网页设计，是一种网页设计的技术做法，该设计可使网站在不同的设备（从桌面计算机显示器到移动电话或其他移动产品设备）上浏览时对应不同分辨率皆有适合的呈现，减少用户进行缩放，平移和滚动等操作行为。

真正的响应式设计方法不仅仅是根据可视区域大小而改变网页布局，而是要从整体上颠覆当前网页的设计方法，是针对任意设备的网页内容进行完美布局的一种显示机制。

用一套代码解决几乎所有设备的页面展示问题

设计工作由产品经理或者美工来出

### 设置视口

模拟移动端的meta

<metaname="viewport"content="width=device-width,minimum-scale=1.0,maximum-scale=1.0,user-scalable=no">

width： 可视区宽度

device-width: 设备宽度

minimum-scale： 最小缩放比

maximum-scale： 最大缩放比

user-scalable： 是否允许用户缩放

Css像素根据设备像素进行计算 1css 像素 == 1 是设备像素 设备的分辨率 dpi值来计算css像素真正展现的大小 适配各种不同分辨率的设备

### 响应式布局实现

1. 流体网格：可伸缩的网格 （大小宽高 都是可伸缩（可用flex或者百分比来控制大小）float）---》 布局上面 元素大小不固定可伸缩

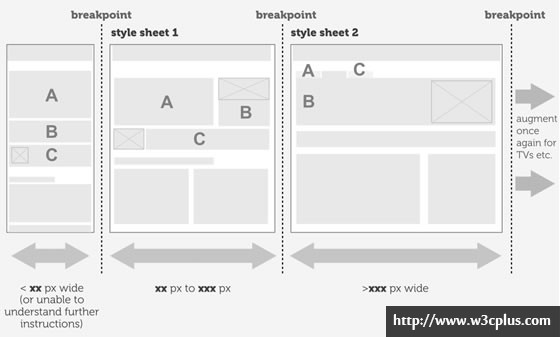
2. 弹性图片：图片宽高不固定（可设置min-width: 100%）

3. 媒体查询：让网页在不同的终端上面展示效果相同（用户体验相同让用户用着更爽） 在不同的设备（大小不同 分辨率不同）上面均展示合适的页面

4. 主要断点： 设备宽度的临界点

大小的区别 ---》 宽度不同 ---》 根据不同宽度展示不同的样式

响应式网页开发主要是在css样式上面进行操作



### 媒体查询

媒体查询是向不同设备提供不同样式的一种方式，它为每种类型的用户提供了最佳的体验。

css2: media type

media type(媒体类型)是css 2中的一个非常有用的属性，通过media type我们可以对不同的设备指定特定的样式，从而实现更丰富的界面。

css3: media query

media query是CSS3对media type的增强，事实上我们可以将media query看成是media type+css属性(媒体特性Media features)判断。

媒体查询不占用权重,css内部引用媒体查询时一般会把媒体查询放在最后,否则相同元素设置样式后面的会覆盖媒体查询的样式

如何使用媒体查询?

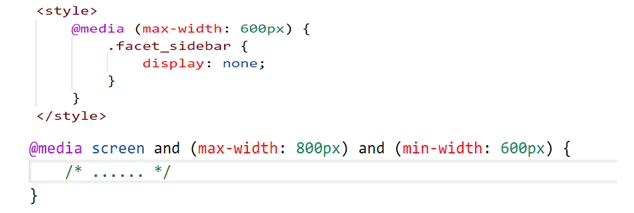
媒体查询的引用方法有很多种：  
link标签  
@import url(example.css) screen and (width:800px);  
css3新增的@media



媒体类型（Media Type): all(全部)、screen(屏幕)、print(页面打印或打印预览模式)

媒体特性（Media features): width(渲染区宽度)、device-width(设备宽度)...

Media Query是CSS3 对Media Type的增强版，其实可以将Media Query看成Media Type(判断条件)+CSS(符合条件的样式规则)



媒体类型(media type):



媒体特性(media features):



逻辑操作符:

合并多个媒体属性 and

@media screen and (min-width: 600px) and (max-width:100px) ；

合并多个媒体属性或合并媒体属性与媒体类型, 一个基本的媒体查询，即一个媒体属性与默认指定的screen媒体类型。

没有or关键词可用于指定备用的媒体功能。相反，可以将备用功能以逗号分割列表的形式列出；

@media screen and (min-width: 769px), print and (min-width: 6in)

这会将样式应用到宽度超过769像素的屏幕或使用至少6英寸宽的纸张的打印设备。

@media not screen and (monochrome)

要指定否定条件，可以在媒体声明中添加关键字not，不能在单个条件前使用not。该关键字必须位于声明的开头，而且它会否定整个声明。所以，上面的示例会应用于除单色屏幕外的所有设备。

向早期浏览器隐藏媒体查询

media="only screen and (min-width: 401px) and (max-width: 600px)"

媒体查询规范还提供了关键字only，它用于向早期浏览器隐藏媒体查询。类似于not，该关键字必须位于声明的开头。

Only指定某种特定的媒体类型 为了兼容不支持媒体查询的浏览器

早期浏览器应该将以下语句media="screen and (min-width: 401px) and (max-width: 600px)"

解释为media="screen"： 换句话说，它应该将样式规则应用于所有屏幕设备，即使它不知道媒体查询的含义。

无法识别媒体查询的浏览器要求获得逗号分割的媒体类型列表，规范要求，它们应该在第一个不是连字符的非数字字母字符之前截断每个值。所以，早期浏览器应该将上面的示例解释为：media="only" 因为没有only这样的媒体类型，所以样式表被忽略。Query -🡪 css3

device-width/device-height 是设备的宽度（如电脑手机的宽度 不是浏览器的宽度）

width/height使用documentElement.clientWidth/Height即viewport的值。渲染(视口)宽度/高度

### 单位值

Rem：rem是CSS3新增的一个相对单位（root em，根em）相对的只是HTML根元素。

Em：em是相对长度单位。相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。

Px： px像素（Pixel）。相对长度单位。像素px是相对于显示器屏幕分辨率而言的。

Vw：相对于视口的宽度。视口被均分为100单位的vw

Vh：相对于视口的高度。视口被均分为100单位的vh

Vmax： 相对于视口的宽度或高度中较大的那个。其中最大的那个被均分为100单位的vmax

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

渐进增强 ---》 iphone6 向上兼容 兼容最新设备

优雅降级 ---》 开发通用版本 再兼容老版本 向下兼容

先移动端 ---》 pc端

先iphone6为初始原型 开发 ---》 兼容其他的设备 ====》 渐进增强