CSS 3

网站：<https://www.w3schools.com/css/default.asp>

参考：<https://www.w3schools.com/cssref/css_functions.asp>

* CSS: 字母大小写都可以，包括id, clas, attribute .
* JQuery: 就比较复杂，例如 选择器 # 则大小写敏感, .class 则大小写不敏感。 而attr()，addClass(), removeClass() 操作则读取大小写不敏感， 而维护操作则大小写敏感。   
  例如：

<img class="fit yes world hello" good=1 src="tfs.png" width="800" height="200" border=2 />

alert($(".Fit").attr("Class"));

$(".FIT").removeClass("Hello Yes").addClass("Ok Bus"); - 删除失败, 大小写不符合

alert($(".Fit").attr("Class"));

alert($(".Fit").attr("Class", "Hao Ren")); - 修改失败， Class 应该小写

<img id="wo" class="fit yes world hello" good=1 src="tfs.png" width="800" height="200" border=2 />

alert($(".Fit").attr("Class"));

$(".FIT").removeClass("Hello Yes").addClass("Ok Bus"); -失败

$(".Fit").attr("CLASS", "blank NEW Class11"); - 失败

alert($(".Fit").attr("class")); - 失败

alert($(".Fit").attr("Class")); - 失败

$("#wo").attr("Good", "Hao Ren"); - 可以

alert($("#wo").attr("Good")); - 可以

alert($("#wo").attr("good")); - 可以

* initial, inherit 可以用于所有CSS 属性

inherit – 将属性的值将继承自它的父元素属性的值。

有些元素的属性: 可继承的 , 即如果不设置属性值，其值会自动继承自父元素的属性值。

例如： color

body { color: blue; }



body { color: blue; } p { color: orange; }



body { color: blue; } p { color: orange; } span { color: red; }



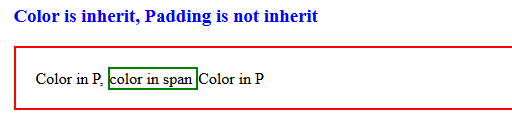
有些属性的属性： 是不可继承的，也就是说不管父元素的属性值是什么。自己如果不设置则取默认值，不会继承。

例如： padding

body { color: blue; padding: 20px; }

p { color: initial; padding: inherit; border: 2px solid red; }

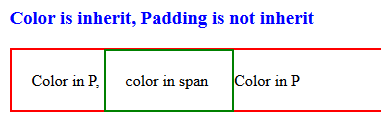
span { color: inherit; border: 2px solid green; }



body { color: blue; padding: 20px; }

p { color: initial; padding: inherit; border: 2px solid red; }

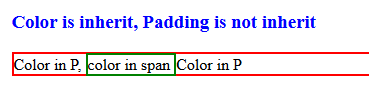
span { color: inherit; padding: inherit; border: 2px solid green; }



body { color: blue; padding: 20px; }

p { color: initial; border: 2px solid red; } - 注意： 没有设置, 默认padding: 0

span { color: inherit; padding: inherit; border: 2px solid green; } - 注意： 此时继承自父元素： padding :0 而不是 20px



initial – 将属性设置为默认值，属性如果没有明确指定某个值，都会有一个默认的值。

例如 ：

position : static; (按照DOM 的输出顺序的位置存放)

padding : 0

opacity: 1 (不透明)

font: 不确定(具体根据浏览器的设置有关)

**注意：**为何会有 initial 这个选项，因为有些元素是可继承的，对于可继承的属性，如果不设置，默认值就不起作用，这种情况我们需要使用initial 显式指定默认值。

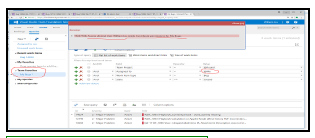
跟图片有关的属性：

* object-fit: fill|contain|cover|scale-down|none|initial|inherit;
* object-position: position|initial|inherit;

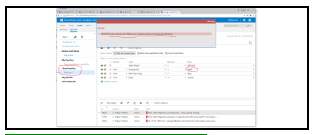
适用：<img> , <video>

用法：用于 <img> 或者<video> 如何填充，涉及到“伸展”,“按等比”，“部分剪切”等方式

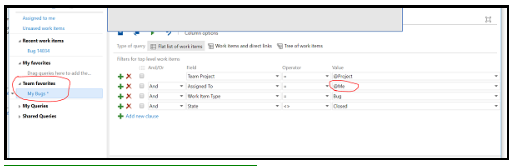
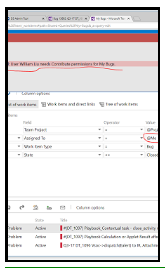
fill – 默认： 根据提供的宽与高，伸展填充。即: 图片变形填充



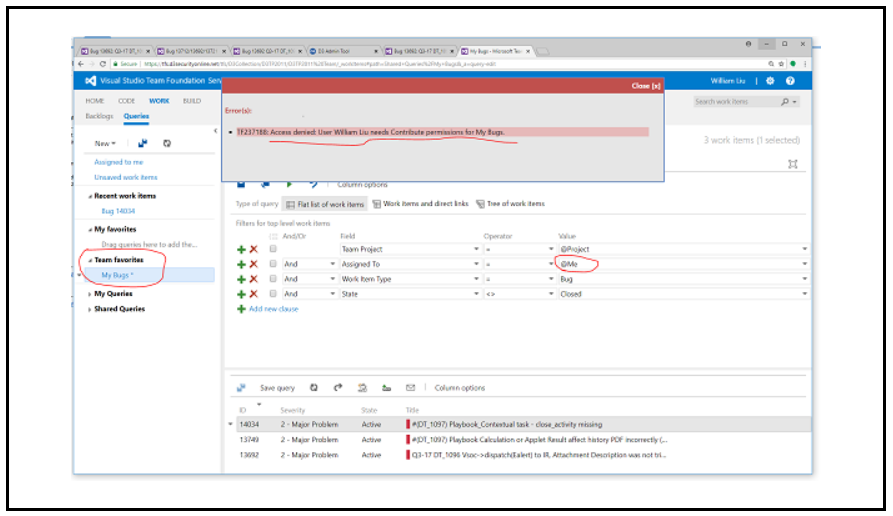
contain – 按等比，全图自适应填充。多余空间： 水平或者垂直居中，总是可以显示全图。 contain 会比较常用



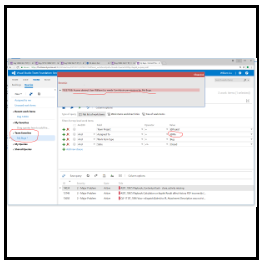
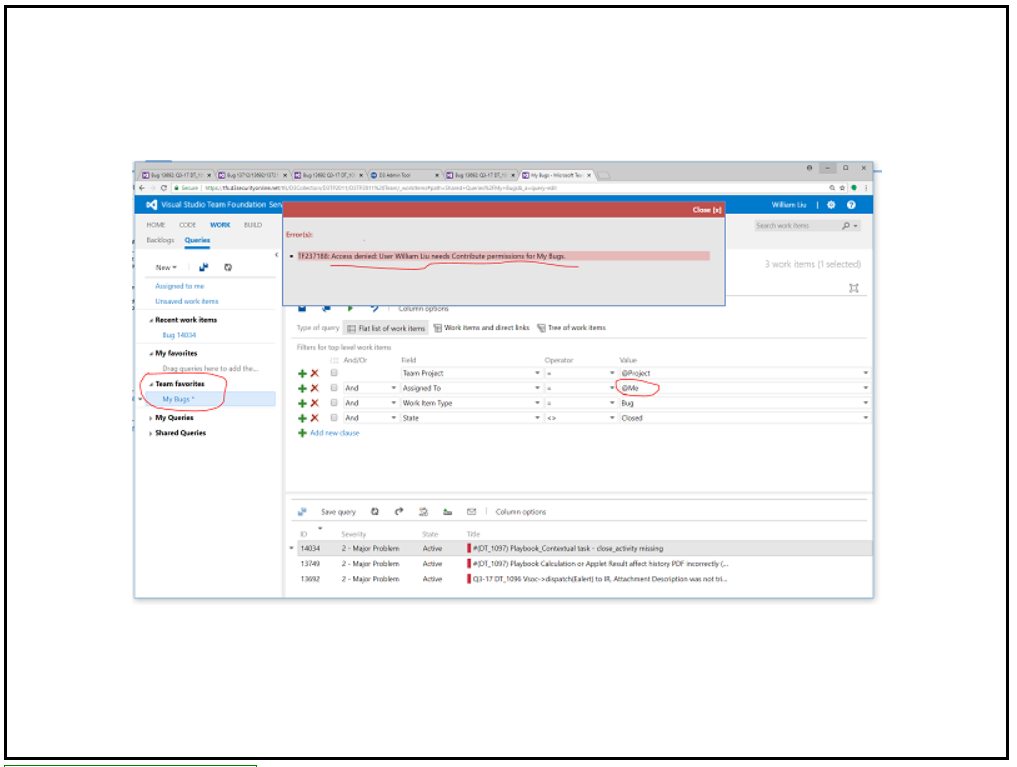
cover – 取宽和高的最大比例，等比伸展，然后宽或者高居中截取部分填充。

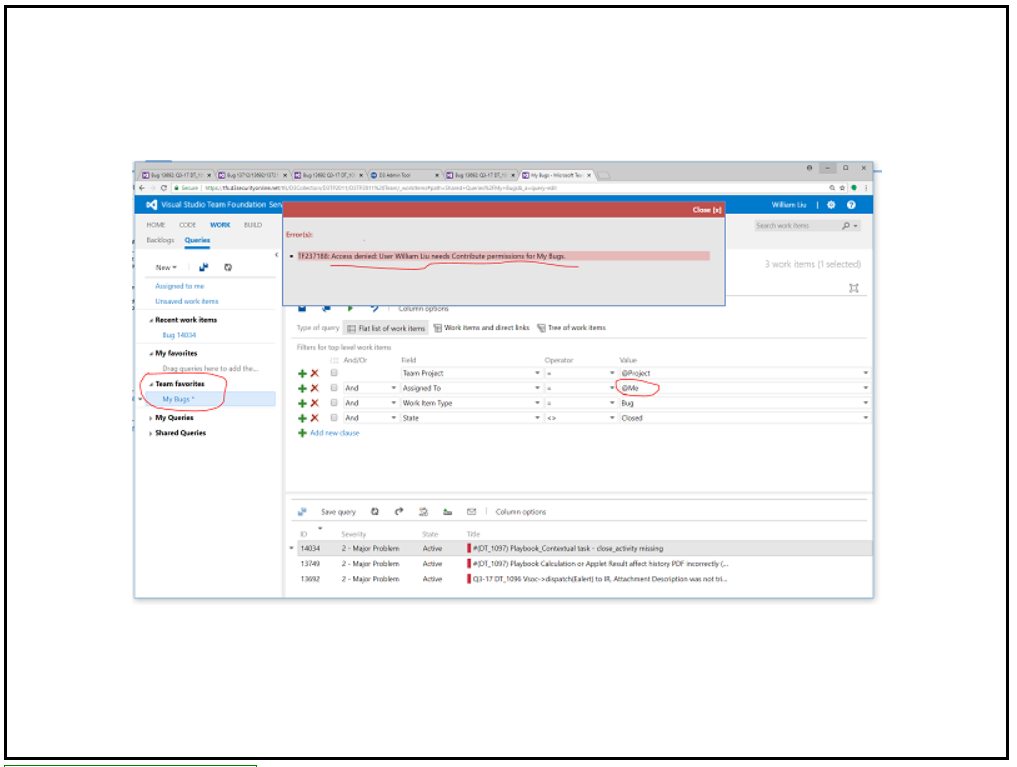
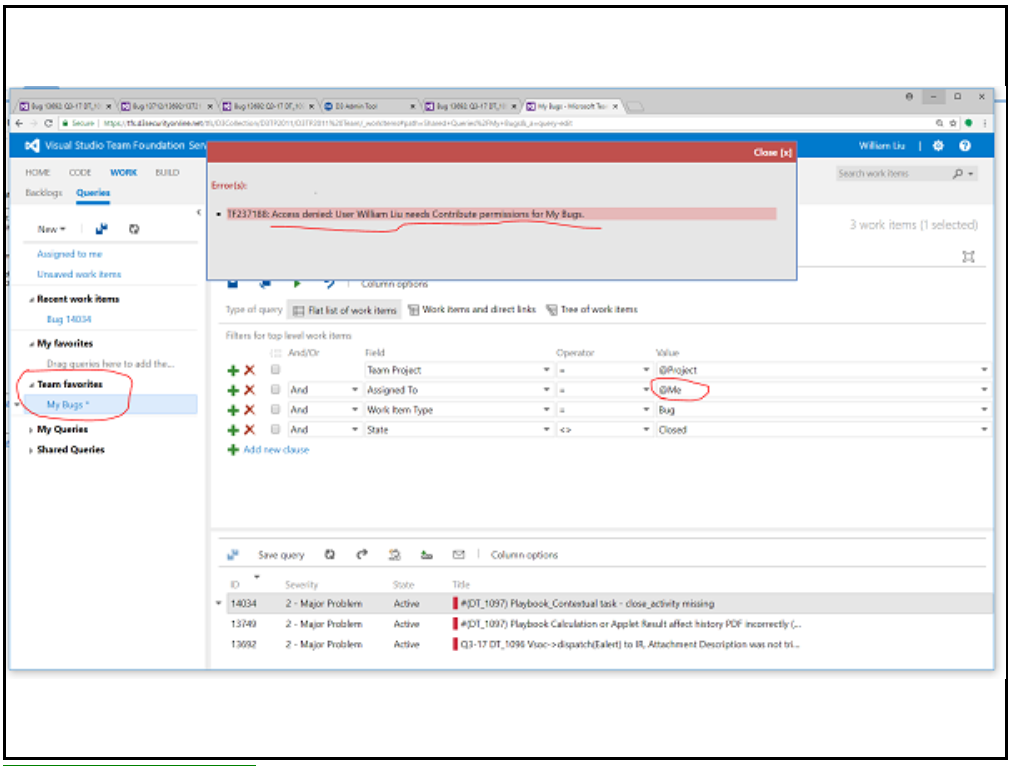
none - 按原图大小， 宽和高都居中填充。



scale-down - 按原图等比缩小，如果容器过大则不会等比放大，总是可以显示全图。与contain 的唯一区别就是不会等比放大。

scale-down 与 contain 的区别：当容器大小小于原图时，没有区别。 当容器大小超过原图大小时，才会有区别

跟动画有关的属性： <http://cubic-bezier.com> 贝塞尔函数曲线

* perspective: *length*|none; 默认值none

perspective 用来产生远景的3D 效果，定义物体离目视距离，越小值3D效果越明显。当我们给一个容器元素定义perspective，那么容器下的子元素将产生 3D 效果，而不是容器本身产生3D效果。

* perspective-origin: *x-axis y-axis*|initial|inherit;

定义站在哪个点位来观看3D物体

* transform: none|*transform-functions*|initial|inherit;

变形：涉及 形状， 位移，大小，旋转，扭曲等

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** |  |
| None | Defines that there should be no transformation |  |

|  |  |  |
| --- | --- | --- |
| matrix(*n,n,n,n,n,n*) | Defines a 2D transformation, using a matrix of six values |  |

|  |  |  |
| --- | --- | --- |
| matrix3d (*n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n*) | Defines a 3D transformation, using a 4x4 matrix of 16 values |  |
| translate(*x,y*) | Defines a 2D translation |  |

|  |  |  |
| --- | --- | --- |
| translate3d(*x,y,z*) | Defines a 3D translation |  |
| translateX(*x*) | Defines a translation, using only the value for the X-axis |  |

|  |  |  |
| --- | --- | --- |
| translateY(*y*) | Defines a translation, using only the value for the Y-axis |  |

|  |  |  |
| --- | --- | --- |
| translateZ(*z*) | Defines a 3D translation, using only the value for the Z-axis |  |
| scale(*x,y*) | Defines a 2D scale transformation |  |

|  |  |  |
| --- | --- | --- |
| scale3d(*x,y,z*) | Defines a 3D scale transformation |  |
| scaleX(*x*) | Defines a scale transformation by giving a value for the X-axis |  |

|  |  |  |
| --- | --- | --- |
| scaleY(*y*) | Defines a scale transformation by giving a value for the Y-axis |  |

|  |  |  |
| --- | --- | --- |
| scaleZ(*z*) | Defines a 3D scale transformation by giving a value for the Z-axis |  |
| rotate(*angle*) | Defines a 2D rotation, the angle is specified in the parameter |  |

|  |  |  |
| --- | --- | --- |
| rotate3d(*x,y,z,angle*) | Defines a 3D rotation |  |
| rotateX(*angle*) | Defines a 3D rotation along the X-axis |  |

|  |  |  |
| --- | --- | --- |
| rotateY(*angle*) | Defines a 3D rotation along the Y-axis |  |
| rotateZ(*angle*) | Defines a 3D rotation along the Z-axis |  |
| skew(*x-angle,y-angle*) | Defines a 2D skew transformation along the X- and the Y-axis |  |
| skewX(*angle*) | Defines a 2D skew transformation along the X-axis |  |
| skewY(*angle*) | Defines a 2D skew transformation along the Y-axis |  |

|  |  |  |
| --- | --- | --- |
| perspective(*n*) | Defines a perspective view for a 3D transformed element |  |
| initial | Sets this property to its default value. [Read about initial](https://www.w3schools.com/cssref/css_initial.asp) |  |
| inherit | Inherits this property from its parent element. [Read about inherit](https://www.w3schools.com/cssref/css_inherit.asp) |  |

注意：

Rotate(angle) – 顺时针旋转， -angle 逆时针旋转

RotateX(angle) – 沿X轴旋转，即垂直方向

RotateY(angle) – 沿Y轴旋转，即水平方向

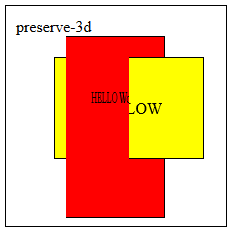
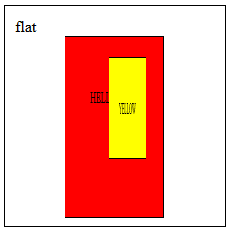
默认是旋转点是 元素的中心点，可以使用transform-origin 设置旋转点

* transform-style: flat|preserve-3d|initial|inherit;

flat – 使用二维平面

preserve-3d – 保留3d 位置

通过图形可以区别：

* transform-origin: *x-axis y-axis z-axis*|initial|inherit;

|  |  |
| --- | --- |
| 默认值：50% 50% 0 |  |

transform-origin: center top;

transform-origin: 30px 30px;

transform-origin: 30% 30%;

* transition: *property duration timing-function delay*|initial|inherit;

transition 转变过渡： 可以对所有属性的变化，指定过渡。

默认： all 0s ease 0s - 也就是说：默认所有 css 属性值得变化，都是即可就完成，即可生效的。

案例：

div {

width: 100px;

height: 100px;

background: red;

}

div:hover {

width: 300px;

background: yellow;

}

鼠标悬停，即刻长度和背景色都马上变化。

div {

width: 100px;

height: 100px;

background: red;

transition: all 2s ease 0s;

}

div:hover {

width: 300px;

background: yellow;

}

鼠标悬停，长度和背景色2秒内动画渐变成最终的值。

transition: width 2s ease 0s;

transition: width 2s ease 0s, background 2s ease 0s;

也可以分开来写：

transition-property: width, background;

transition-duration: 2s;

transition-delay: 1s;

transition-timing-function: linear;

* transition-timing-function: linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps(int,start|end)|cubic-bezier(*n*,*n*,*n*,*n*)|initial|inherit;

也可以使用贝塞尔函数：自定义

transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);

总结： 以下三个属性，即可自定动画效果

perspective - 3D效果

transform – 变形：大小，旋转等

transition – 自定义渐变过渡效果，可以应用到任何CSS 属性，当然也可以应用到transform

例子：

div.window {

display: block;

position: relative;

left: 50px;

top: 50px;

width: 400px;

height: 280px;

border: 2px solid purple;

perspective: 200px !important;

}

div.window > div {

border: 1px solid red;

transform: RotateX(0deg);

transform-style: preserve-3d;

transform-origin: 30px 30px;

transition-property: transform !important;

transition-duration: 2s;

transition-delay: 1s;

transition-timing-function: linear;

transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);

}

div.window:hover > div,

div.window.flip > div

{

transform: RotateX(-45deg);

}

<div class="window">

<div>

<img src="tsf.jpg" style="width:100%;height:100%;object-fit:contain" />

</div>

</div>

Case 1 Case 2 – css changed as below

div.window {

display: block;

position: relative;

left: 50px;

top: 50px;

width: 400px;

height: 280px;

border: 2px solid purple;

perspective: 150px !important;

perspective-origin: 30% 15%; -- 新增

}

div.window > div {

border: 1px solid red;

transform: RotateX(0deg);

transform-style: preserve-3d;

transform-origin: 0px 0px; -- 修改

transition-property: transform;

transition-duration: 2s;

transition-delay: 1s;

transition-timing-function: linear;

transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);

}

div.window:hover > div,

div.window.flip > div

{

transform: RotateX(-45deg);

}

如何制作翻转页面（正反面）:

Flip: <https://davidwalsh.name/demo/css-flip.php>

此方案 在 IE 上工作不正常，FireFox, Chrome 上工作正常

.flip-container {

display: block;

position: relative;

width: 300px;

height: 180px;

border: 2px solid navy;

perspective: 1000px;

}

.flip-container > .flipper {

display: block;

position: static;

width: 100%;

height: 100%;

transition: 1s; -主要是对 flipper 进行翻转

transform-style: preserve-3d;

}

.flip-container:hover > .flipper, - 用于鼠标盘旋控制翻转

.flip-container.flip > .flipper - 用于javascript 控制翻转

{

transform: rotateY(-180deg); -主要是对 flipper 进行翻转

}

function toggleFlip() { - 用于javascript 控制翻转

document.querySelector(".flip-container").classList.toggle("flip");

}

对于 front , back 页面内容，backface-visibility: hidden; 预设 front 翻转0度，而对back 翻转180度，

这样 front - visible, back - hidden

当 flipper 翻转180度以后：front 和 back 的情况刚好对调。

.flip-container > .flipper > .front,

.flip-container > .flipper > .back {

display: block;

position: absolute;

top: 0px;

left:0px;

bottom:0px;

right:0px;

backface-visibility: hidden;

}

.flip-container > .flipper > .front {

border: 1px solid purple;

transform: rotateY(0deg);

z-index: 2;

}

.flip-container > .flipper > .back {

border: 1px solid orange;

transform: rotateY(180deg);

}

<div class="flip-container">

<div class="flipper">

<div class="front">

<img src="tsf.jpg" style="width:100%;height:100%;object-fit:contain" />

</div>

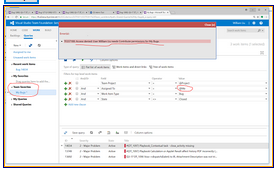
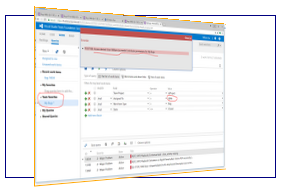
<div class="back">

<img src="tfs.png" style="width:100%;height:100%;object-fit:contain" />

</div>

</div>

</div>



使用transition简单而且在所有浏览器工作正常：

.flip-container {

display: block;

position: relative;

width: 300px;

height: 180px;

perspective: 1000px;

border: 2px solid navy;

}

.flip-container > .flipper {

display: block;

position: static;

width: 100%;

height: 100%;

transform: rotateY(0deg);

transition: transform 1s;

transform-style: preserve-3d;

}

.flip-container:hover > .flipper,

.flip-container.flip > .flipper

{

transform: rotateY(-180deg);

}

.flip-container:hover > .flipper > .front,

.flip-container.flip > .flipper > .front

{

opacity: 0;

}

.flip-container:hover > .flipper > .back,

.flip-container.flip > .flipper > .back

{

opacity: 1;

}

.flip-container > .flipper > .front,

.flip-container > .flipper > .back {

display: block;

position: absolute;

top: 0px;

left:0px;

bottom:0px;

right:0px;

}

.flip-container > .flipper > .front {

border: 1px solid purple;

z-index: 2;

transform: rotateY(0deg);

transition: opacity 1s;

opacity: 1;

}

.flip-container > .flipper > .back {

border: 1px solid orange;

z-index: 1;

transform: rotateY(180deg);

transition: opacity 1s;

opacity: 0;

}

重点：

flipper -> flipper:hover 1秒内实现翻转

front , back -> front:hover , back:hover 1秒内切换显示

总结： 以下三个属性，即可自定动画效果

perspective - 3D效果

transform – 变形：大小，旋转等

transition – 自定义渐变过渡效果，可以应用到任何CSS 属性，当然也可以应用到transform

除了以上可以定义动画以外，还有另外一种机制定义动画：

* animation: *name duration timing-function delay iteration-count direction fill-mode play-state*;

主要有 8 个属性：

* [animation-name](https://www.w3schools.com/cssref/css3_pr_animation-name.asp) - keyFrames name
* [animation-duration](https://www.w3schools.com/cssref/css3_pr_animation-duration.asp) - 默认0s, 即动画不执行， 动画持续时间
* [animation-timing-function](https://www.w3schools.com/cssref/css3_pr_animation-timing-function.asp) - ease(默认) , 也可以是贝塞尔函数cubic-bezier(.17,.67,.94,.06)
* [animation-delay](https://www.w3schools.com/cssref/css3_pr_animation-delay.asp) - 默认0s, 动画延时多少秒开始
* [animation-iteration-count](https://www.w3schools.com/cssref/css3_pr_animation-iteration-count.asp) - 默认1， 执行1次。动画执行重复多少次
* [animation-direction](https://www.w3schools.com/cssref/css3_pr_animation-direction.asp)

animation-direction: normal|reverse|alternate|alternate-reverse|initial|inherit;

reverse – 反向执行，即从结束点开始执行到开始点

alternate – 先正向执行，然后反向执行。 跟执行的次数有关系， 如果是 1 次， 则不会反向执行，

如果是奇数， 则最后一次没有反向执行的机会

* [animation-fill-mode](https://www.w3schools.com/cssref/css3_pr_animation-fill-mode.asp)

animation-fill-mode: none|forwards|backwards|both|initial|inherit;

none - 动画结束前后，不会应用@keyframes 里定义的CSS

forwards – 会保留@keyframes最后一帧动画的动画CSS

backwards – 动画开始时就执行 from 的第一帧，动画执行完毕，保留第一帧的CSS, 这个第一帧不是from (0%), 而是元素原始设置。

both – 动画开始就执行第一帧， 最后结束停留在最后一帧的CSS

forwards + delay : 在delay 期间保留元素原始的CSS.

backwards + delay , both + delay : delay 期间是使用from 的CSS

* [animation-play-state](https://www.w3schools.com/cssref/css3_pr_animation-play-state.asp)

animation-play-state: paused|running|initial|inherit;

animation 必须配合一起使用： @keyframes *animationname* {*keyframes-selector* {*css-styles;}*}

如果动画持续 100秒， 10%则对应 10秒的位置，75%则对应75秒的位置， 100%则对应100秒的位置

@keyframes mymove {

0% {top: 0px; background: pink;}

10% {top:50px; background: orange;}

15% {background: purple;}

20% {background: blue;}

25% {top: 200px; background: yellow;}

75% {top: 250px; background: red;}

100% {top: 400px; background: green;}

}

@keyframes mymove {

from {top: 0px; width: 20px; height:20px; background:pink;}

to {top: 200px; width: 300px; height: 300px; background:green; }

}

<style>

div {

width: 100px;

height: 100px;

background: red;

position: relative;

animation: mymove 3s;

animation-delay: 2s;

animation-fill-mode: both;

}

@keyframes mymove {

from {top: 0px; background-color: yellow;}

to {top: 200px; background-color: blue;}

}

</style>

animation 与 transition 的区别：

1. transition 的触发是自动触发的，transition当相关的属性发生变化, 自动发生渐变。toggle 时都是渐变的

animation的触发是可控制的， 触发的次数，最终的CSS设置都是可控的

1. transition 的渐变动画也是自动计算的，不能自定义属性如何渐变，可以自定义的是时间节奏。

Animation的动画可以灵活定义使用@keyframes, 也可以自定义时间节奏。

CSS3 有3种和动画相关的属性：transform, transition, animation。

其中 transform 描述了元素静态样式。而transition 和 animation 却都能实现动画效果。

所以三者之中transform 常常配合后两者使用，在页面实现酷炫的五毛（或五元）特效。

但后两者又有什么区别呢？

这篇文章做了比较好的总结：

<http://www.kirupa.com/html5/css3_animations_vs_transitions.htm>

我罗列其中的要点：

不同点：

1.  触发条件不同。transition通常和hover等事件配合使用，由事件触发。animation则和gif动态图差不多，立即播放。

2. 循环。 animation可以设定循环次数。

3. 精确性。 animation可以设定每一帧的样式和时间。tranistion 只能设定头尾。 animation中可以设置每一帧需要单独变化的样式属性， transition中所有样式属性都要一起变化。

4. 与javascript的交互。animation与js的交互不是很紧密。tranistion和js的结合更强大。js设定要变化的样式，transition负责动画效果，天作之合，比之前只能用js时爽太多。

结论：

1. 如果要灵活定制多个帧以及循环，用animation.

2. 如果要简单的from to 效果，用 transition.

3. 如果要使用js灵活设定动画属性，用transition.

背景

background: *bg-color bg-image position/bg-size bg-repeat bg-origin bg-clip bg-attachment* initial|inherit;

The properties that can be set, are:

* [background-color](https://www.w3schools.com/cssref/pr_background-color.asp) 默认 transparent
* [background-image](https://www.w3schools.com/cssref/pr_background-image.asp)
* [background-position](https://www.w3schools.com/cssref/pr_background-position.asp) 默认 0% 0%
* [background-size](https://www.w3schools.com/cssref/css3_pr_background-size.asp) 默认 auto
* [background-repeat](https://www.w3schools.com/cssref/pr_background-repeat.asp) 默认 repeat
* [background-origin](https://www.w3schools.com/cssref/css3_pr_background-origin.asp) 默认padding-box
* [background-clip](https://www.w3schools.com/cssref/css3_pr_background-clip.asp) 默认 border-box
* [background-attachment](https://www.w3schools.com/cssref/pr_background-attachment.asp) 默认scroll

background-attachment: scroll|fixed|local|initial|inherit;

fixed – 位置相对于整个浏览器的主页面固定，此时[background-position](https://www.w3schools.com/cssref/pr_background-position.asp)也是相对于页面的位置。

我们来看看例子，很好的理解 fixed, scroll, local 的区别：

<div class="bg">

Hello World

<br><br><br><br><br>

<br><br><br><br><br>

<br><br><br><br><br>

<br><br><br><br><br>

Hello World

<br><br><br><br><br>

<br><br><br><br><br>

<br><br><br><br><br>

<br><br><br><br><br>

Hello World

</div>

div.bg {

display:block;

width: 200px;

height: 200px;

border: 1px solid green;

overflow: auto;

background-image: url(like.png);

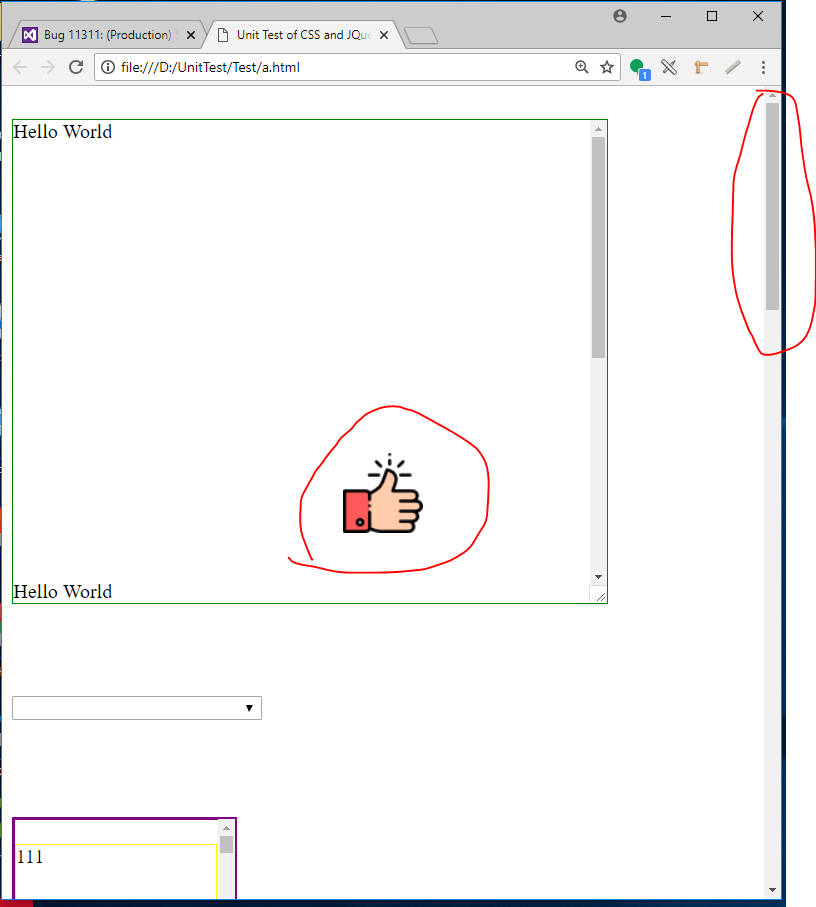
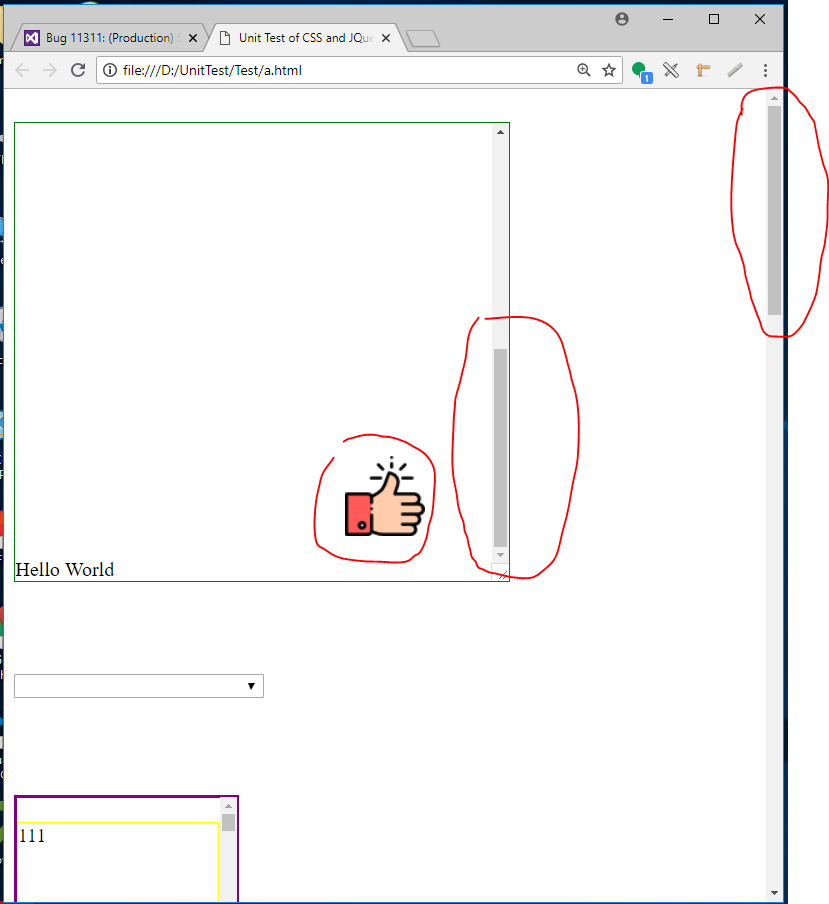
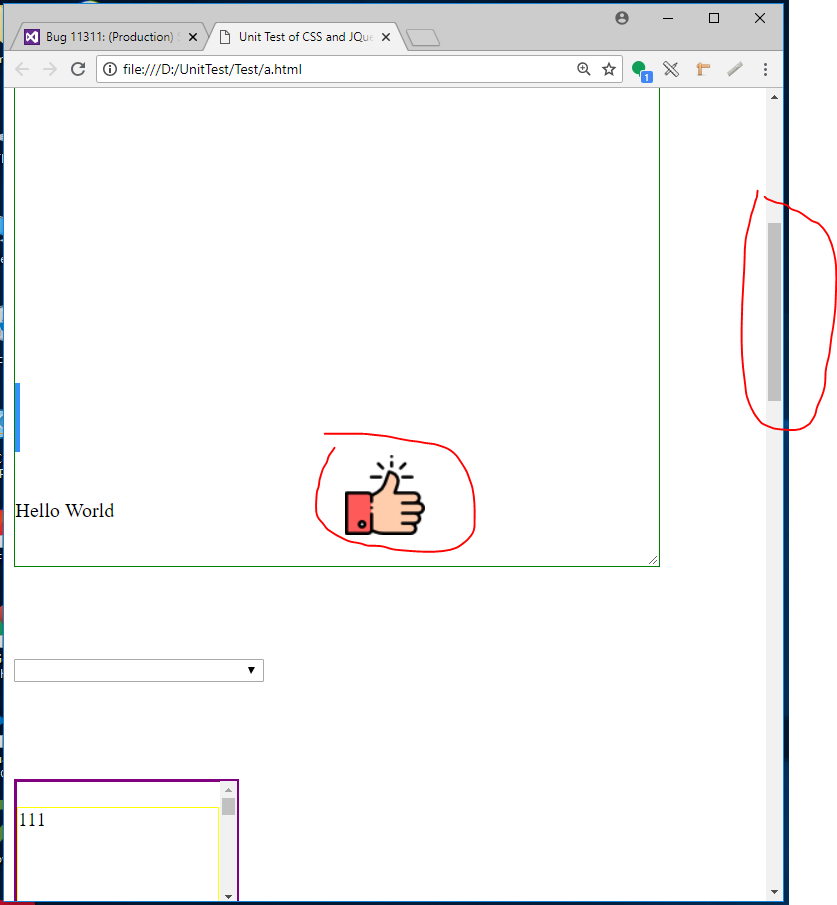
background-attachment:fixed; - fixed

background-position: center center; - center center

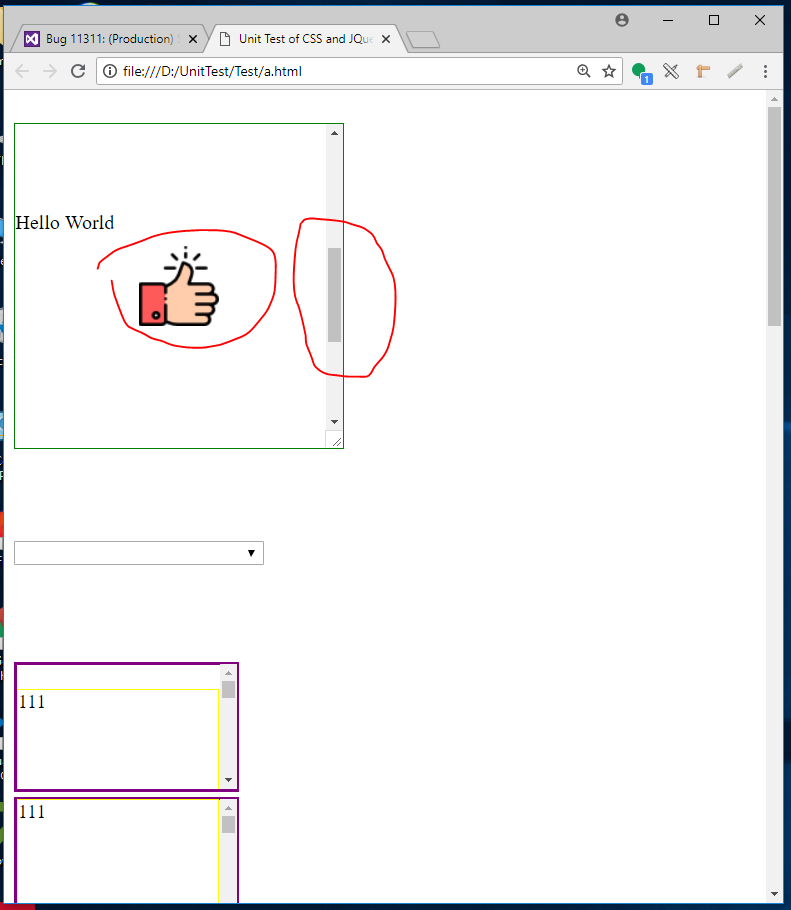
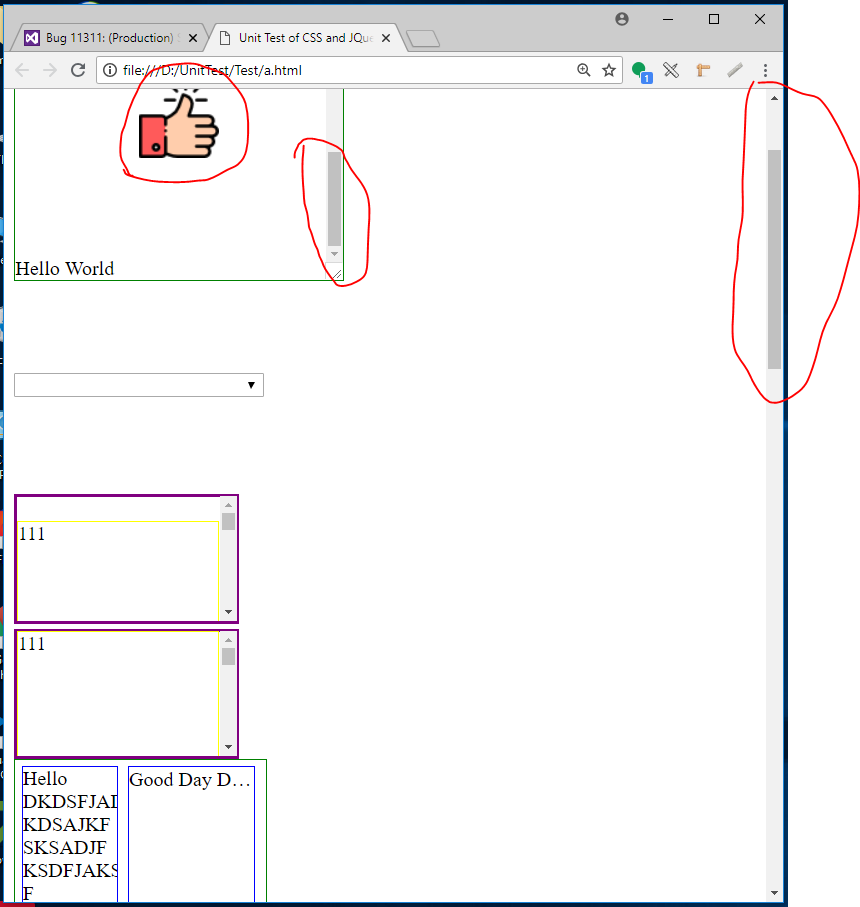
background-repeat: no-repeat;

resize: both;

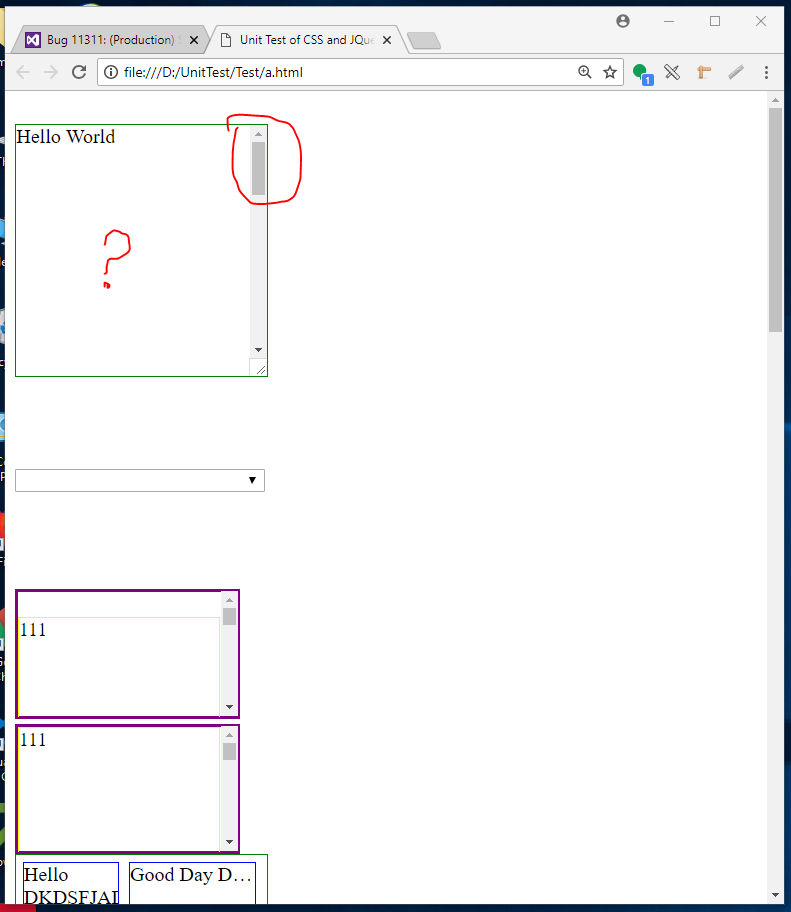
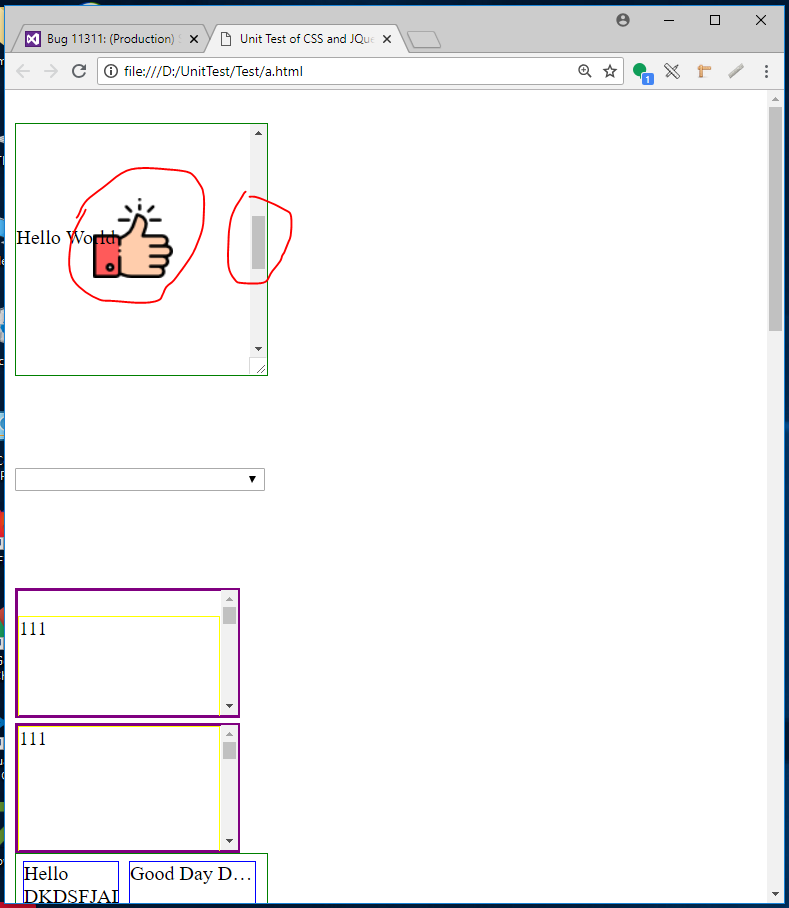
}

background-attachment:scroll; background-position: center center; – 位置相对于容器的位置。

background-attachment:local; background-position: center center; – 位置相对于容器里的内容大小的位置。

* [background-clip](https://www.w3schools.com/cssref/css3_pr_background-clip.asp) - 这是整个背景画布的大小布置方式

background-clip: border-box|padding-box|content-box|initial|inherit;

background-attachment:local; border-box无效，默认使用padding-box， 还可以使用content-box

background-attachment:fixed or scroll; 都可以使用

div.bg {

display:block;

width: 200px;

height: 200px;

border: 20px dotted grey;

padding: 30px;

overflow: auto;

background-image: url(like.png);

background-attachment:scroll;

background-repeat: no-repeat;

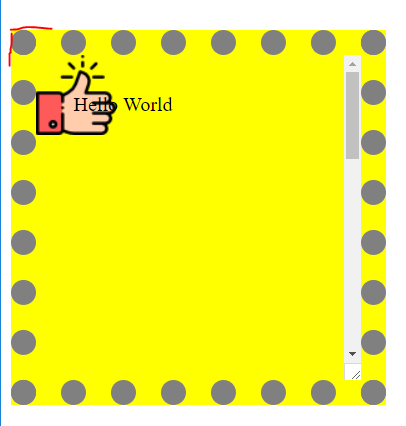
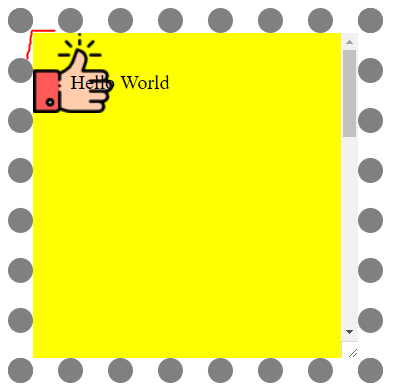
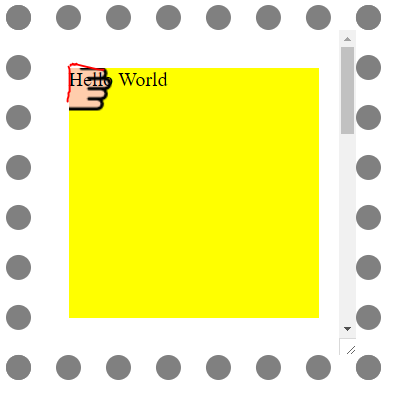
background-clip: border-box;

background-color: yellow;

resize: both;

}

border-box padding box content-box

[background-origin](https://www.w3schools.com/cssref/css3_pr_background-origin.asp) - 这是背景图片的布置位置方式

background-origin: padding-box|border-box|content-box|initial|inherit;

div.bg {

display:block;

width: 200px;

height: 200px;

border: 20px dotted grey;

padding: 30px;

resize: both;

overflow: auto;

background-image: url(like.png);

background-attachment:scroll;

background-repeat: no-repeat;

background-color: yellowgreen;

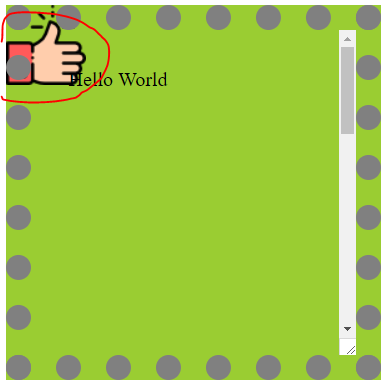
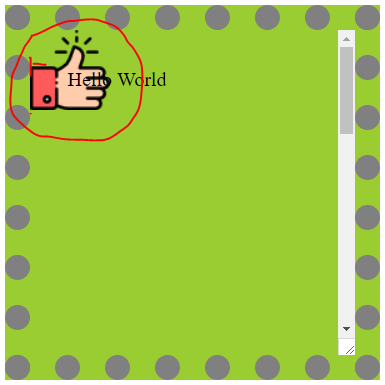
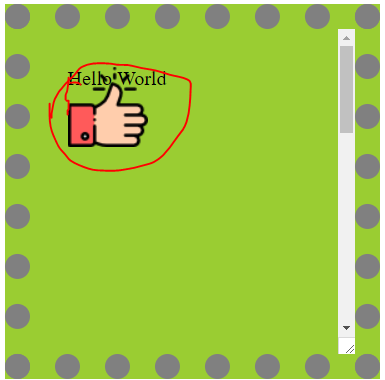
background-clip: border-box;

background-origin: content-box;

background-position:0px 0px;

}

padding-box border-box content-box

[background-size](https://www.w3schools.com/cssref/css3_pr_background-size.asp) - 用于控制背景图片大小

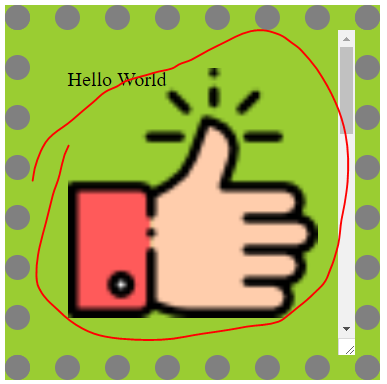
background-size: auto|*length*|cover|contain|initial|inherit;

cover – 图片伸展填充, 也要根据 border-box, padding-box 还是content-box 来决定填充的面积

contain- 自动伸缩保持最佳全图来填充

length – 可以是 % px , 可以等比 200px (按宽) ， auto 200px(按高) , 可以伸展 100px 200px;

background-size: auto 200px;



背景图片可以指定多个：

div.bg {

display:block;

width: 200px;

height: 200px;

border: 20px dotted grey;

padding: 30px;

resize: both;

overflow: auto;

background-color: yellowgreen; - 不支持多个

background-clip: border-box; - 不支持多个

background-image: url(like.png), url(envelope.png);

background-attachment:scroll;

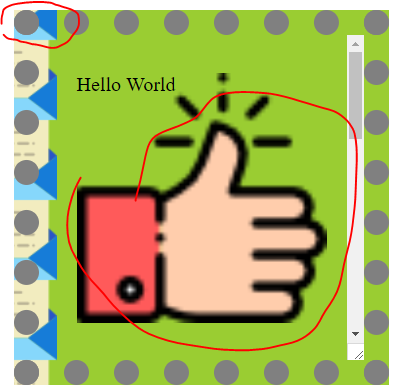
background-repeat: no-repeat, repeat-y;

background-origin: content-box, border-box;

background-position:0px 0px, -30px -40px;

background-size: auto 200px, auto;

}



[background-blend-mode](https://www.w3schools.com/cssref/css3_pr_background-origin.asp) 指定两个背景图片重叠时的叠加效果，默认normal

background-blend-mode: normal|multiply|screen|overlay|darken|lighten|color-dodge|saturation|color|luminosity;

background-blend-mode: lighten;

div.bg {

display:block;

width: 200px;

height: 200px;

border: 20px dotted grey;

padding: 30px;

resize: both;

overflow: auto;

background-color: yellowgreen;

background-clip: border-box;

background-image: url(like.png), url(envelope.png); - 如果重叠，前面的图片在最顶端

background-attachment:scroll;

background-repeat: no-repeat, repeat-y;

background-origin: content-box, border-box;

background-position:0px 0px, 20px 20px;

background-size: auto 200px, 100px;

background-blend-mode:lighten;

}



背景渐变色，不能使用background-color, 而是使用 background-image:

background-image: linear-gradient(red, yellow, green);

[border-image](https://www.w3schools.com/cssref/css3_pr_background-origin.asp)

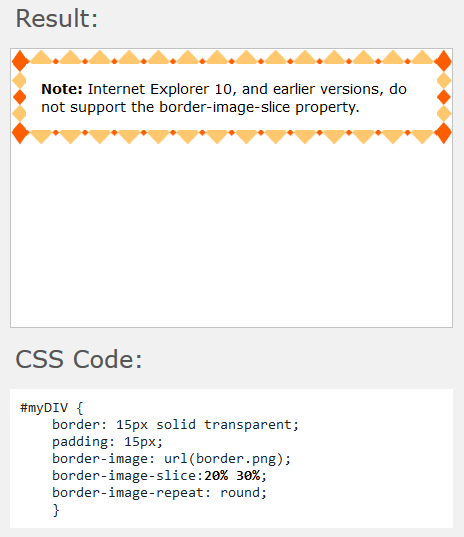
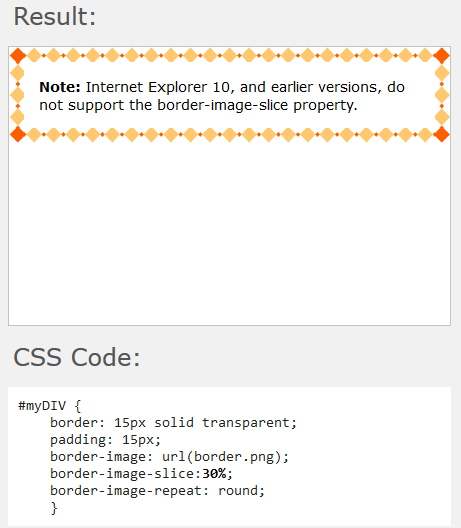
border-image: *source slice width outset repeat*|initial|inherit;

* [border-image-source](https://www.w3schools.com/cssref/css3_pr_border-image-source.asp)
* [border-image-slice](https://www.w3schools.com/cssref/css3_pr_border-image-slice.asp)

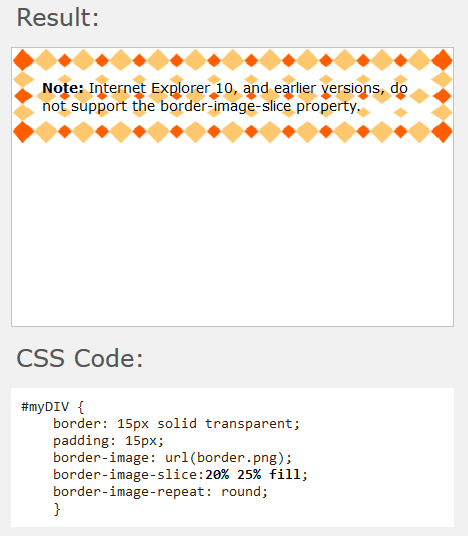
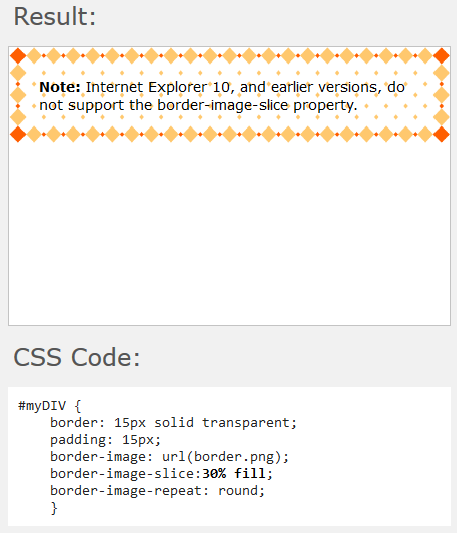
border-image-slice: *number*|*%*|fill|initial|inherit;

border-image-slice 属性指定了如何剪切图片，图片被切为 9方格: 4角，4边，1中心区

border-image-slice:20% 30%; border-image-slice:30%; border-image-slice:30;

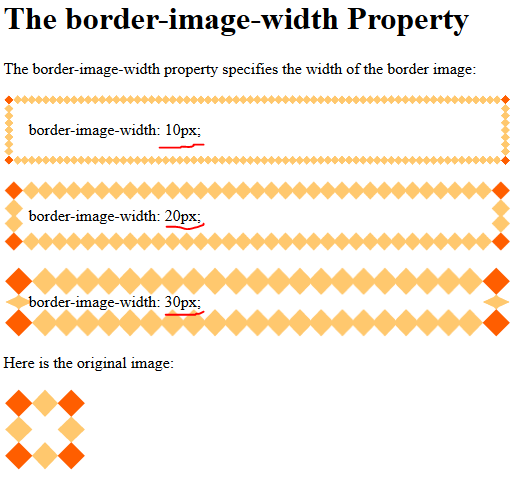
  

border-image-slice:20% 25% fill; border-image-slice:30% fill;

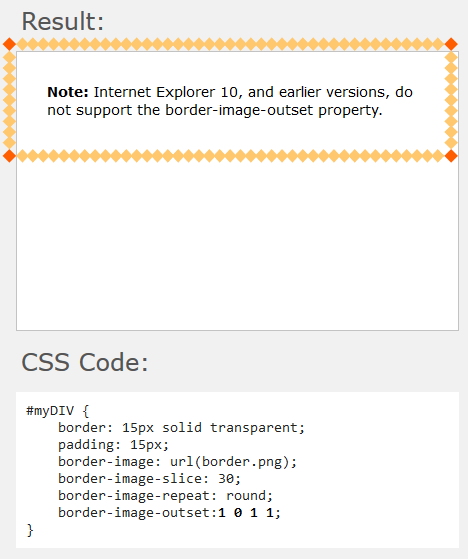
* [border-image-width](https://www.w3schools.com/cssref/css3_pr_border-image-width.asp)

border-image-width: *number*|*%*|auto|initial|inherit;



* [border-image-outset](https://www.w3schools.com/cssref/css3_pr_border-image-outset.asp)  
  border-image-outset: *length*|*number*|initial|inherit;

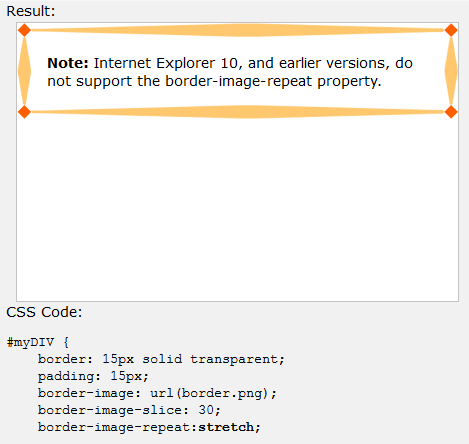
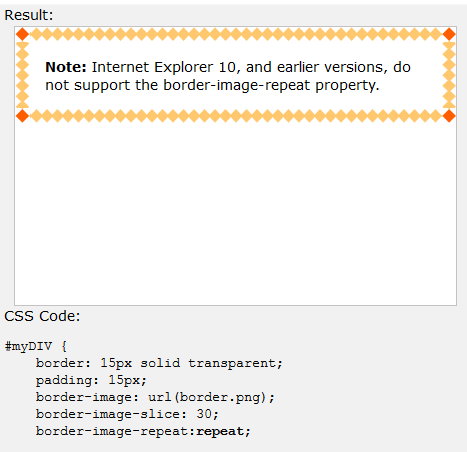
border-image-outset:10px 5px 15px 10px; border-image-outset:1 0 1 1;

* [border-image-repeat](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp) 默认stretch

border-image-repeat: stretch|repeat|round|initial|inherit;

stretch repeat round

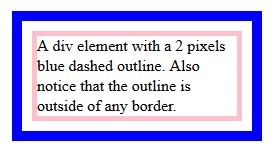
  

[outline](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp)

outline: *outline-width* *outline-style* *outline-color*|initial|inherit;

* [outline-width](https://www.w3schools.com/cssref/pr_outline-width.asp)
* [outline-style](https://www.w3schools.com/cssref/pr_outline-style.asp) (required)
* [outline-color](https://www.w3schools.com/cssref/pr_outline-color.asp)
* outline-offset: *length*|initial|inherit;

div {

 display: block;

width: 200px;

margin: auto;

border: 5px solid pink;

outline: 10px solid blue;

outline-offset: 10px;

}

outline 不占空间，所以注意会覆盖其他元素空间

[resize](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp) 前提 overflow: auto, hidden, scroll , overflow: visible; 将不工作

resize: none|both|horizontal|vertical|initial|inherit;

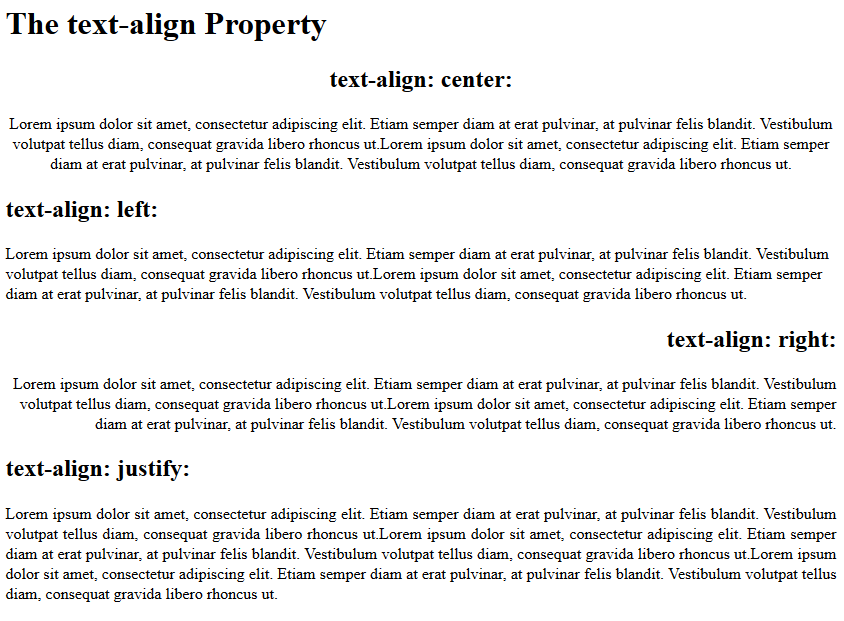
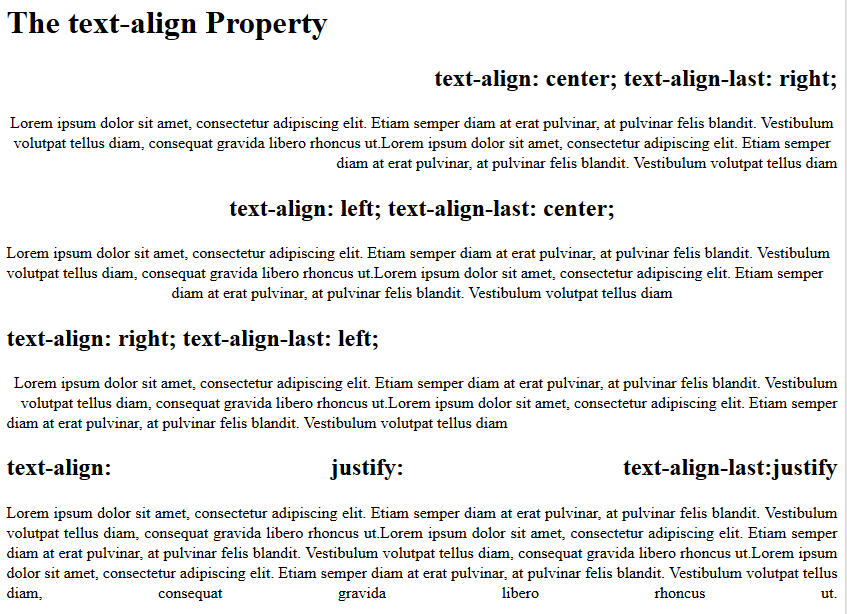
文字处理

[text-align](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp)

[text-align](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp)-last

text-align: left|right|center|justify|initial|inherit;

text-align-last: auto|left|right|center|justify|start|end|initial|inherit;

[direction](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp)

direction: ltr|rtl|initial|inherit;

direction 属性指定区块block里的文本的方向和书写方向.

div.rtl {

    display:    inline-block;

padding:    20px;

border:     1px solid orange;

direction:  rtl;

}

div.rtl > div {

    display:    inline-block;

padding:    5px;

width:      60xp;

height:     60px;

line-height:60px;

border:     1px solid green;

}

<div class="rtl">

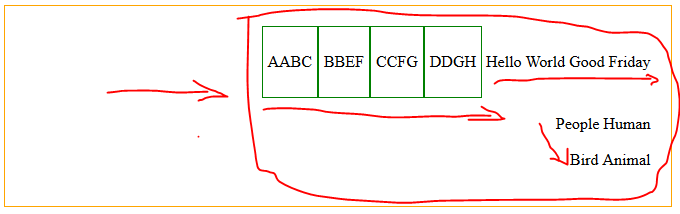
<div>AABC</div><div>BBEF</div><div>CCFG</div><div>DDGH</div>

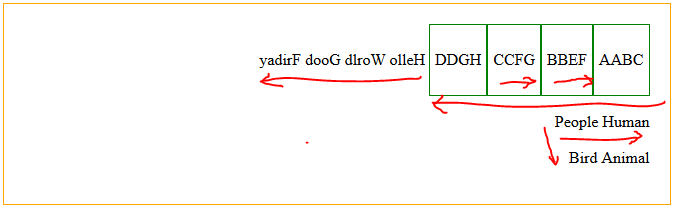
<span>Hello</span> <span>World</span> <span>Good</span> <span>Friday</span>

<p>People Human</p><p>Bird Animal</p>

</div>

direction:  rtl; text-align:  right;

unicode-bidi 必须结合direction 一起使用

unicode-bidi: normal|embed|bidi-override|initial|inherit;

direction: rtl;

unicode-bidi: bidi-override;

text-decoration

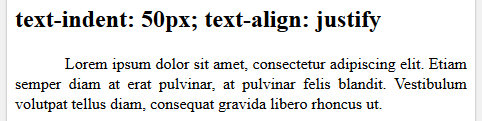
text-decoration: text-decoration-line text-decoration-color text-decoration-style|initial|inherit;

* text-decoration-line (required) - none|underline|overline|line-through|initial|inherit;
* text-decoration-color - *color*|initial|inherit;
* text-decoration-style - solid|double|dotted|dashed|wavy|initial|inherit;

text-decoration: overline underline line-through orange wavy;



text-indent: *length*|initial|inherit;



text-justify: auto|inter-word|inter-character|none|initial|inherit;

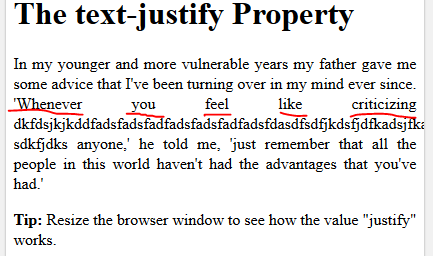
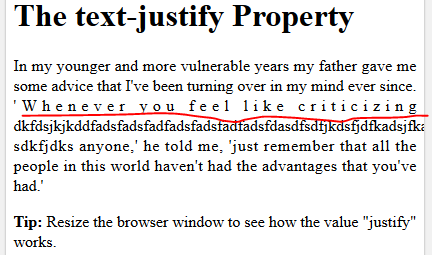
div {

text-align: justify;

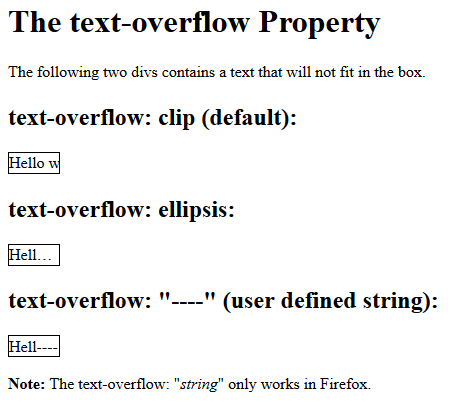
text-justify: inter-word;

}

text-justify: inter-word; text-justify: inter-character;

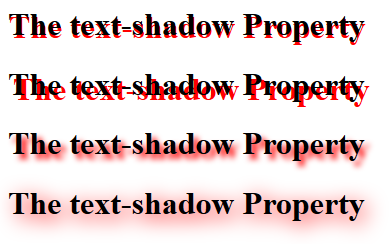
 

text-overflow: clip|ellipsis|*string*|initial|inherit;



text-shadow: *h-shadow v-shadow blur-radius color*|none|initial|inherit;

h1.a {

 text-shadow: 1px 3px #FF0000;

}

h1.b {

text-shadow: 5px 5px #FF0000;

}

h1.c {

text-shadow: 5px 5px 5px #FF0000;

}

h1.d {

text-shadow: 5px 5px 20px #FF0000;

}

<h1 class="a">The text-shadow Property</h1>

<h1 class="b">The text-shadow Property</h1>

<h1 class="c">The text-shadow Property</h1>

<h1 class="d">The text-shadow Property</h1>

text-transform: none|capitalize|uppercase|lowercase|initial|inherit;

word-break: normal|break-all|keep-all|break-word|initial|inherit;

word-wrap: normal|break-word|initial|inherit;

div {

text-align: justify;

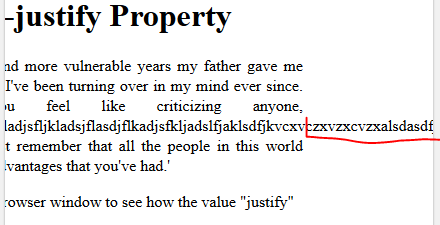
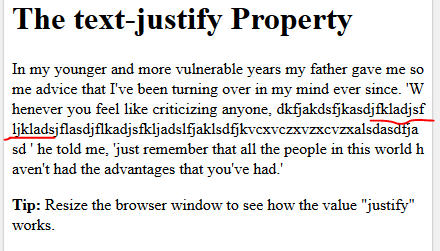
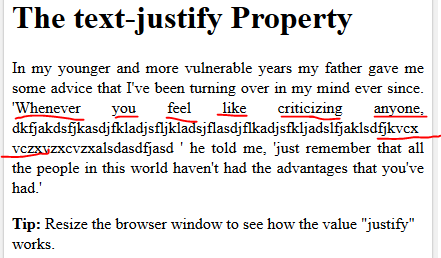
text-justify: inter-word;

word-break: break-word; - break-all 不完美

word-wrap: break-word; - 完美解决方案：word-break + word-wrap : break\_word;

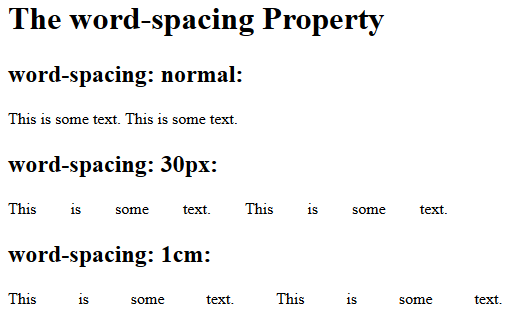
}

word-break: break-word; word-break: break-all; word-break:break-word; word-wrap:break-word;

word-spacing: normal|*length*|initial|inherit;

letter-spacing: normal|*length*|initial|inherit;



margin

margin: *length*|auto|initial|inherit;

div {

    display: inline-block;

width: 300px;

margin: auto;

border: 3px solid purple;

}

display:inline-block; margin:auto;未居中 display:block; margin:auto; block + auto 可以实现居中效果

float

float: none|left|right|initial|inherit;

注意 使用float时， position:absolute; float不起作用, float: top left..是相对位移，margin也可以用。float 对于父容器来说仍然占据空间，这点position:absolute 对于父容器来说是不占空间的。

table 有关的属性

table-layout: auto|fixed|initial|inherit;

caption-side: top|bottom|initial|inherit;

border-collapse: separate|collapse|initial|inherit;

border-spacing: *length*|initial|inherit;

empty-cells: show|hide|initial|inherit;

## HTML Table Tags

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<table>](https://www.w3schools.com/tags/tag_table.asp) | Defines a table |
| [<th>](https://www.w3schools.com/tags/tag_th.asp) | Defines a header cell in a table |
| [<tr>](https://www.w3schools.com/tags/tag_tr.asp) | Defines a row in a table |
| [<td>](https://www.w3schools.com/tags/tag_td.asp) | Defines a cell in a table |
| [<caption>](https://www.w3schools.com/tags/tag_caption.asp) | Defines a table caption |
| [<colgroup>](https://www.w3schools.com/tags/tag_colgroup.asp) | Specifies a group of one or more columns in a table for formatting |
| [<col>](https://www.w3schools.com/tags/tag_col.asp) | Specifies column properties for each column within a <colgroup> element |
| [<thead>](https://www.w3schools.com/tags/tag_thead.asp) | Groups the header content in a table |
| [<tbody>](https://www.w3schools.com/tags/tag_tbody.asp) | Groups the body content in a table |
| [<tfoot>](https://www.w3schools.com/tags/tag_tfoot.asp) | Groups the footer content in a table |

<table border="1" cellspacing="30" cellpadding="30" >

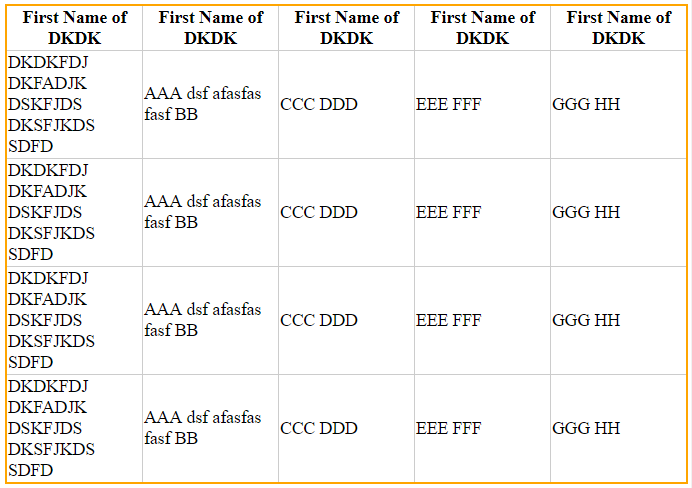
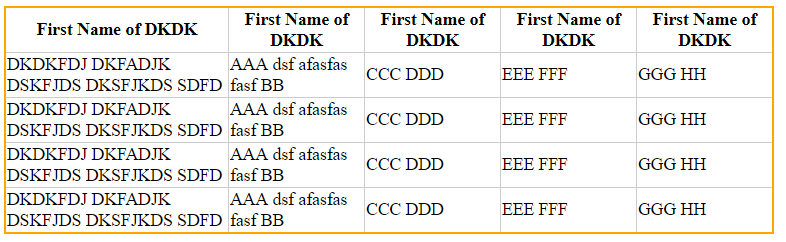
border-spacing 优先级高于 cellspacing

table > td padding 优先级高于 cellpadding

table-layout: fixed vs auto

fixed: 当table 的大小有多余或者被压缩时，各个列的宽度一致

auto: 当table 的大小有多余或者被压缩时， 各个列宽度根据内容自动分配，并且文字只会wrap不出现ellipsis

当table 的大小可以自适应时， fixed和auto 的效果是一样的

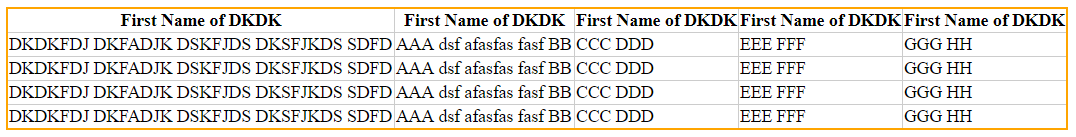


table {

border-collapse: collapse;

table-layout: fixed;

border: 2px solid orange;

width: 100%; - width 100% fixed 才会对列均分宽度

min-width: 400px;

overflow: auto;

}

table th,

table td {

min-width: 40px;

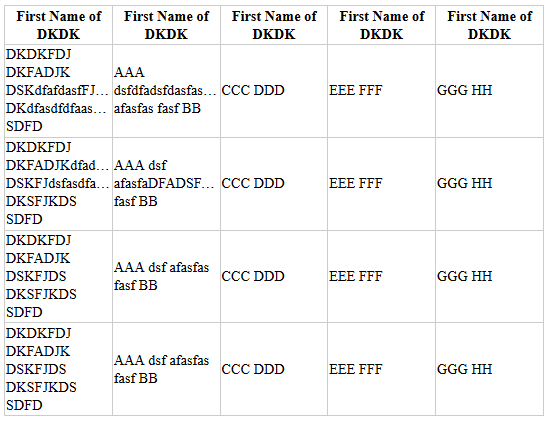
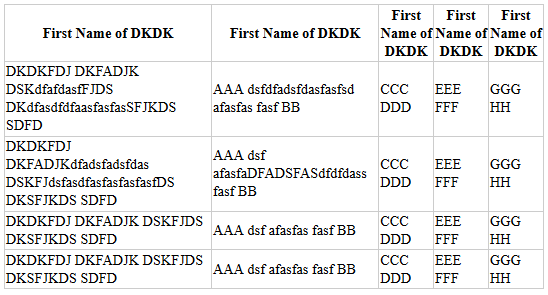
overflow: hidden;

text-overflow: ellipsis;

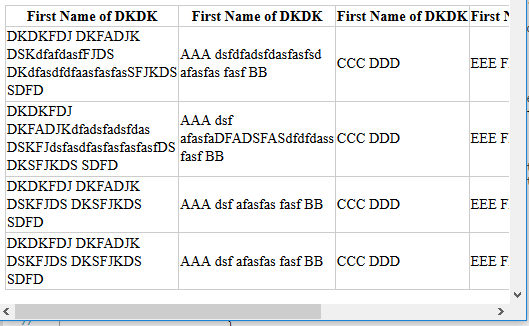
border: 1px solid #cccccc;

}

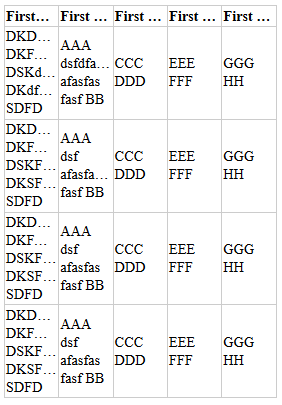
fixed auto

auto: 最多出现文字wrap, 不会出现 ellipsis, 其最宽度取决于内容的最紧宽度。



fixed:



比较常用的设置：

table {

border-collapse: collapse;

table-layout: auto;

border: 2px solid orange;

display: block; - 如果table超宽则会出现水平滚动条

width: 100%; - 可有可无

min-width: 300px; - 设置一个最小宽度

overflow: auto; - 入股超宽则出现滚动条

}

table th,

table td {

min-width: 40px;

overflow: hidden;

text-overflow: ellipsis;

border: 1px solid #cccccc;

}

关于table 也完全可以通过 display: 属性来定义

display: *value*;

|  |  |  |
| --- | --- | --- |
| table | Let the element behave like a <table> element |  |

|  |  |  |
| --- | --- | --- |
| table-caption | Let the element behave like a <caption> element |  |
| table-column-group | Let the element behave like a <colgroup> element |  |
| table-header-group | Let the element behave like a <thead> element |  |
| table-footer-group | Let the element behave like a <tfoot> element |  |
| table-row-group | Let the element behave like a <tbody> element |  |
| table-cell | Let the element behave like a <td> element |  |
| table-column | Let the element behave like a <col> element |  |
| table-row | Let the element behave like a <tr> element |  |

position

position: static|absolute|fixed|relative|sticky|initial|inherit;

static – 默认： 按照在文档页面中出现的顺序, 注意 static 不象 relative, 它不受 top, left, bottom, right 的影响。

top, left, right, bottom 不起作用。

absolute – 绝对位置， 相对于第一个（position**非static**）祖先元素的位置。

fixed – 相对于浏览器窗口的位置

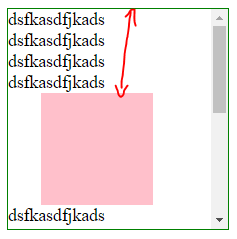
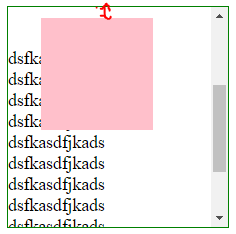
relative – 相对于正常位置

sticky – 位置在 relative 与 fixed 之间切换，主要取决于滚动的位置

sticky 取决于父元素的滚动位置：

position: sticky; top: 10px;

滚动到距离顶端 10px 的位置， 变为fixed

注意： sticky 在 edge 和 IE 上不能正常工作，可以使用javascript

案例：

<https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_sticky_header>

<script>

window.onscroll = function() {myFunction()};

var header = document.getElementById("myHeader");

var sticky = header.offsetTop;

function myFunction() {

if (window.pageYOffset > sticky) {

header.classList.add("sticky");

} else {

header.classList.remove("sticky");

}

}

</script>

.top-container {

background-color: #f1f1f1;

padding: 30px;

text-align: center;

}

.header {

padding: 10px 16px;

background: #555;

color: #f1f1f1;

}

.content {

padding: 16px;

}

.sticky {

position: fixed;

top: 0;

width: 100%;

}

.sticky + .content {

padding-top: 102px;

}

width, height 固定宽度和高度

top, left, bottom, right - 如果width, height 固定长度， top, left 也设置值，则 bottom, right 不起作用。

width, height 没有设置

top, left, bottom, right - 可以决定元素的大小, 通常用于贴边变化大小，尤其对于 absolute, fixed 。

可以通过此方法，设置自适应大小的区块元素。

[display:](https://www.w3schools.com/cssref/css3_pr_border-image-repeat.asp)

display: *value*;

|  |
| --- |
| **Value** |
| inline |

|  |  |  |
| --- | --- | --- |
| block | Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width |  |
| contents | Makes the container disappear, making the child elements children of the element the next level up in the DOM |  |
| flex | Displays an element as a block-level flex container |  |
| grid | Displays an element as a block-level grid container |  |
| inline-block | Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values |  |
| inline-flex | Displays an element as an inline-level flex container |  |
| inline-grid | Displays an element as an inline-level grid container |  |
| inline-table | The element is displayed as an inline-level table |  |
| list-item | Let the element behave like a <li> element |  |
| run-in | Displays an element as either block or inline, depending on context |  |
| table | Let the element behave like a <table> element |  |
| table-caption | Let the element behave like a <caption> element |  |
| table-column-group | Let the element behave like a <colgroup> element |  |
| table-header-group | Let the element behave like a <thead> element |  |
| table-footer-group | Let the element behave like a <tfoot> element |  |
| table-row-group | Let the element behave like a <tbody> element |  |
| table-cell | Let the element behave like a <td> element |  |
| table-column | Let the element behave like a <col> element |  |
| table-row | Let the element behave like a <tr> element |  |
| none | The element is completely removed |  |
| initial | Sets this property to its default value. [Read about initial](https://www.w3schools.com/cssref/css_initial.asp) |  |
| inherit | Inherits this property from its parent element. [Read about inherit](https://www.w3schools.com/cssref/css_inherit.asp) |  |

重点讲讲 display: flex 弹性布局

<https://www.cnblogs.com/xuyuntao/articles/6391728.html>

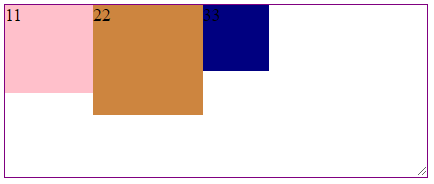
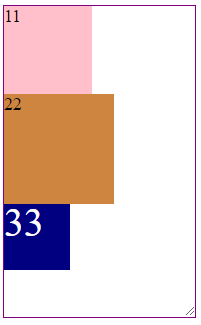
<https://www.cnblogs.com/myzhibie/p/4318904.html>

**Flex是Flexible Box的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。**设为Flex布局以后，子元素的float、clear和vertical-align属性将失效。

容器属性：

* flex-direction: row|row-reverse|column|column-reverse|initial|inherit;

row (默认) column

* justify-content: flex-start|flex-end|center|space-between|space-around|initial|inherit;

项目在主轴上的对齐方式(主轴究竟是哪个轴要看属性flex-direction的设置了)

flex-start：在主轴上由左或者上开始排列，默认值

flex-end：在主轴上由右或者下开始排列

center：在主轴上居中排列

space-between：在主轴上左右两端或者上下两端开始排列, 两端贴边，中间项目等距排列

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

\*沿主轴方向的对齐方式是没有 baseline 的，只有在主轴的垂直方向上才有baseline

<div style="display: flex; justify-content:flex-start;">

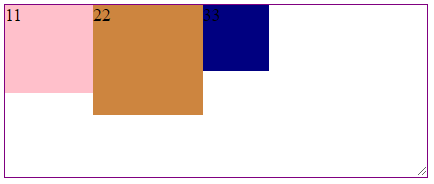
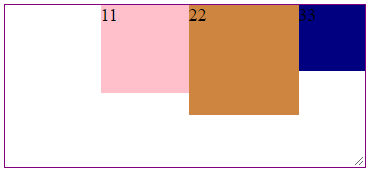
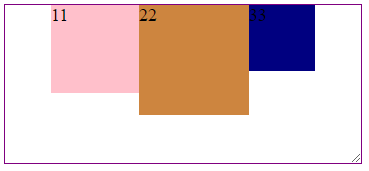
<div style="display:block;width:80px;height:80px;background:pink;">11</div>

<div style="display:block;width:100px;height:100px;background:peru;">22</div>

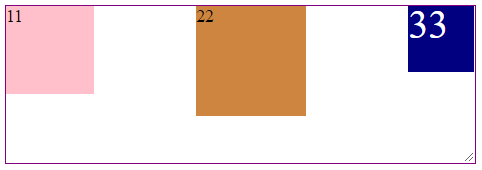
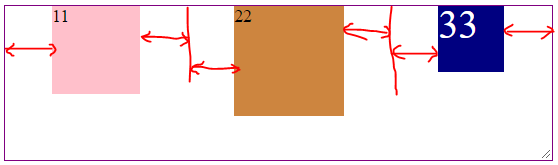
<div style="display:block;width:60px;height:60px;background:navy;">33</div>

</div>

flex-start flex-end center

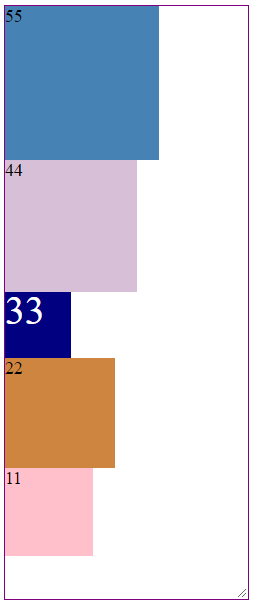
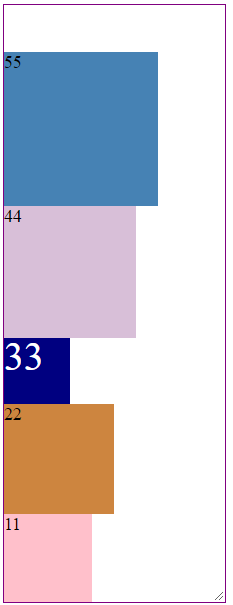
space-between space-around

flex-direction 与 justify-content 的组合使用：

flex-direction:column-reverse; flex-direction:column-reverse;

justify-content:flex-end; (贴最后一个) justify-content:flex-start;(贴第一个)

* align-items: stretch|center|flex-start|flex-end|baseline|initial|inherit;

项目在侧轴(与主轴垂直)上的对齐方式(侧轴究竟是哪个轴要看属性flex-direction的设置了)

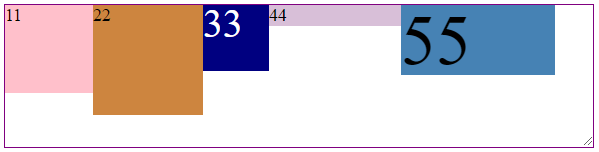
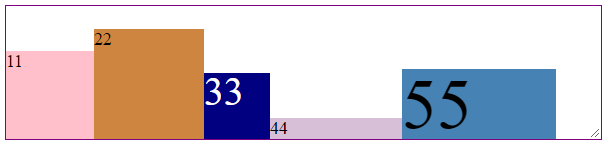
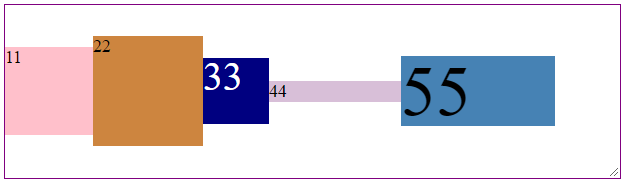
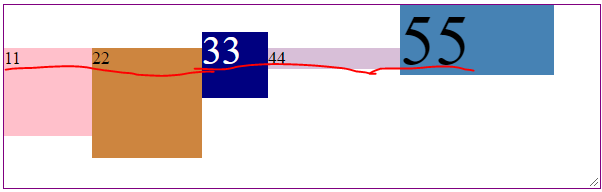
stretch – 自动伸缩, 默认值，

如果项目在相应的侧轴上，设置了高度(垂直侧轴)， 或者宽度(水平侧轴), 则不会自动填充

baseline – 字体的下端对齐

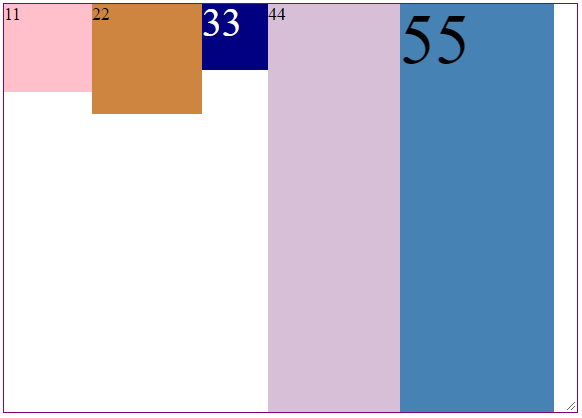
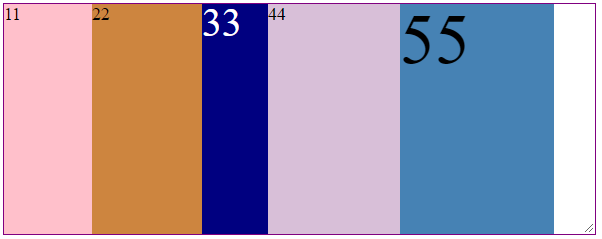
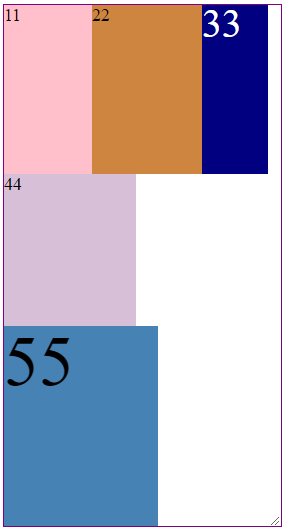
\*没有 space-between , space-around 因为有且只有一行的情况，不用考虑wrap 的情况

flex-start flex-end center baseline

stretch - 主轴如果是水平, 那么容器里的项目可以不用设置height. 如果设置则height固定，否则就是自动伸缩。

如 44, 55 没有设置高度 都没有设置高度

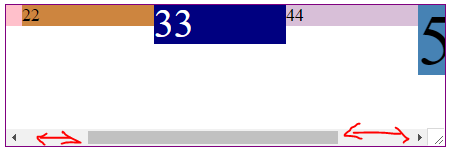
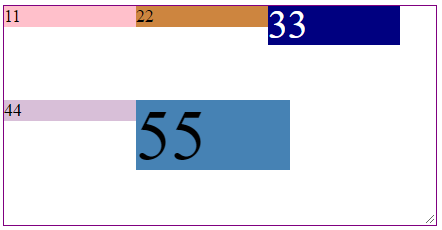
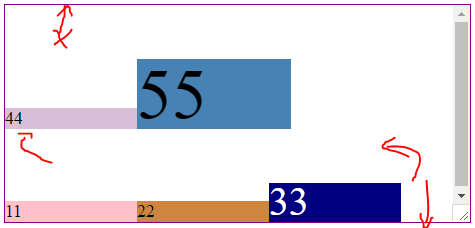
* flex-wrap: nowrap|wrap|wrap-reverse|initial|inherit;

属性可选值的范围为nowrap(默认)不换行、wrap换行（第一行在上方）和wrap-reverse（你懂的~）

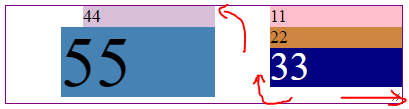
正常的换行： 水平换行是往下, 垂直换行是往右。

wrap-reverse : 默认 水平主轴会下对齐，往上换行。垂直主轴会右对齐，往左换行。

nowrap wrap wrap-reverse

column + wrap-reverse



* align-content: stretch|center|flex-start|flex-end|space-between|space-around|initial|inherit;

align-content 是与wrap 一起工作的，如果没有换行形成多行，此属性则不起作用.

案例：

div.flex {

display: flex;

flex-direction: row;

justify-content: flex-start;

align-items: stretch;

flex-wrap: wrap;

align-content: flex-start;

border: 1px solid purple;

overflow: auto;

resize: both;

}

div.flex > div {

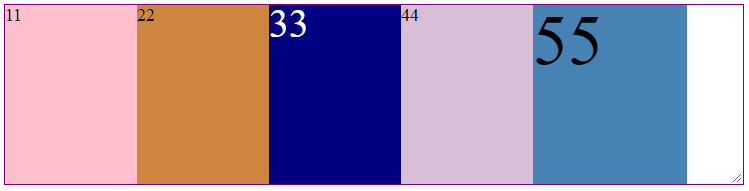
display: block;

width: 120px;

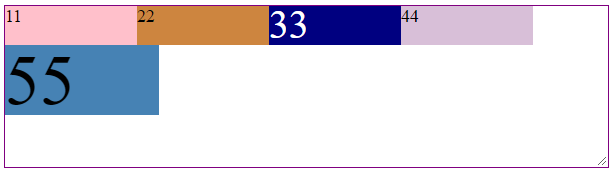
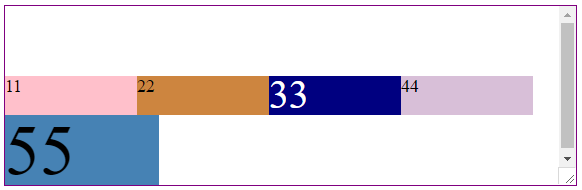
flex-shrink: 0;

}

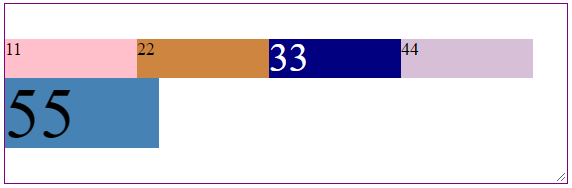
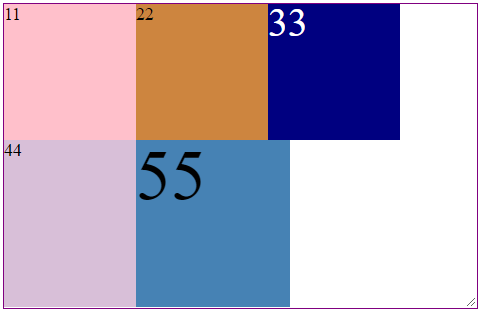
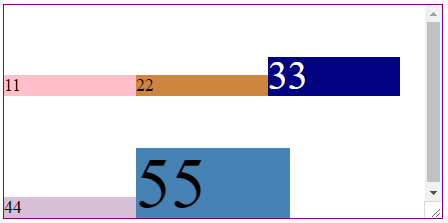
开始状态： 注意： align-items:stretch;



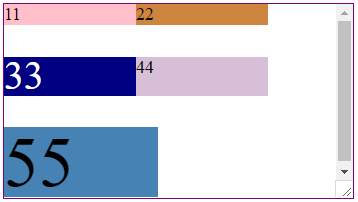
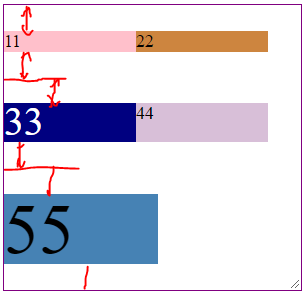
flex-start 换行以后项目并没有自动伸缩 flex-end

center stretch – 取决于align-items:stretch; 如果是其他flex-end

space-between space-around

* align-items vs align-content

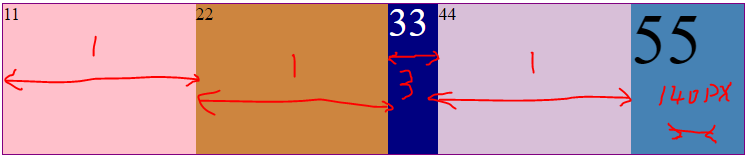
align-content 是与wrap 一起工作的，形成多行以后，此属性起作用，如果没有形成换行，align-content的设置不起作用，

在没有形成换行的情况下，align-items 属性的设置起作用。

项目属性：

* order - 项目的排列顺序。数值越小，排列越靠前，默认为0。
* flex-shrink - 项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。

数值越大，收缩越快， 3 是 1 的 3 倍收缩量

、

* flex-grow - 项目的放大比例，默认为0，即如果存在剩余空间，也不放大。

数值越大, 放大越快， 3 是 1 的 3 倍收缩量

* flex-basic - 在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。

如果允许换行，则主轴上换行的条件大小由flex-basic 来决定

div.flex { display: flex; flex-wrap: wrap; }

div.flex > div {

display: block;

width: 200px;

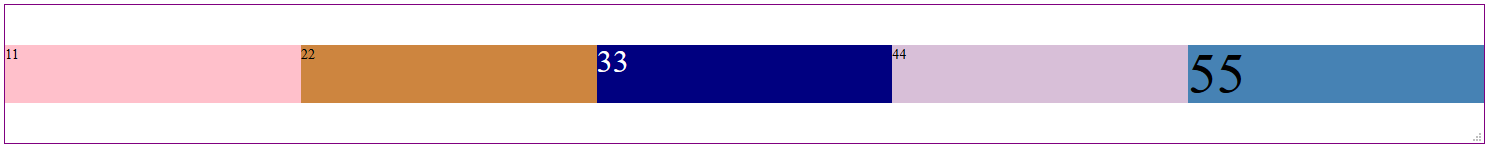
flex-shrink: 1;

flex-grow: 1;

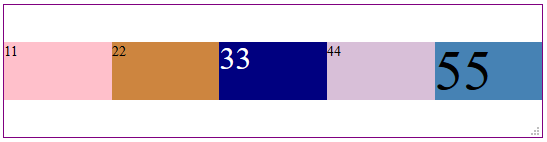
flex-basis: 120px;

}

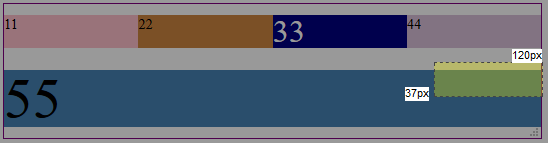
自动伸展



可以收缩



可以收缩到了120px; flex-basis:120px; 则换行



div.flex > div {

display: block;

width: 80px;

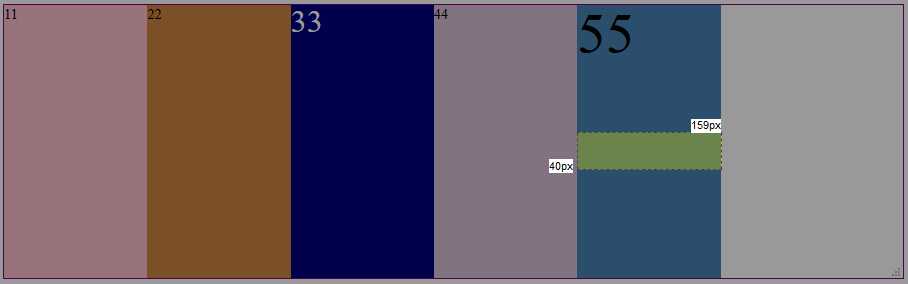
flex-shrink: 0;

flex-grow: 0;

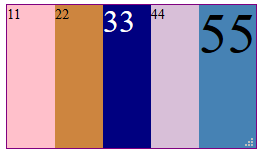
flex-basis: 160px;

}

不允许收缩与伸展： 实际宽度是 160px 而不是 80px



flex-shrink:1; flex-grow:0;



* align-self

align-self: auto|stretch|center|flex-start|flex-end|baseline|initial|inherit;

默认是auto - 继承它父元素 align-items 的设置，如果没有父元素那么默认就是“stretch”

44: align-self: center;



* 主轴上项目的大小收缩与伸展，是由 flex-basic, flex-grow, flex-shrink 来控制的

Flex 的一个实际应用的例子：

一共3个tag 在容器里，只允许最后一个Div的宽度可以弹性

<style>

h3 {

display:flex;

flex-direction:row;

justify-content: flex-start;

align-items: center;

flex-wrap: nowrap;

box-sizing: border-box;

border: 1px solid red;

overflow: hidden;

resize: both;

}

h3 > div {

border: 1px solid green;

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

flex-shrink: 1;

flex-grow: 1;

}

h3 > div:first-of-type {

border: 5px solid orange;

flex-shrink: 0;

flex-grow: 0;

}

h3 > div:nth-of-type(2) {

text-align: right;

}

</style>

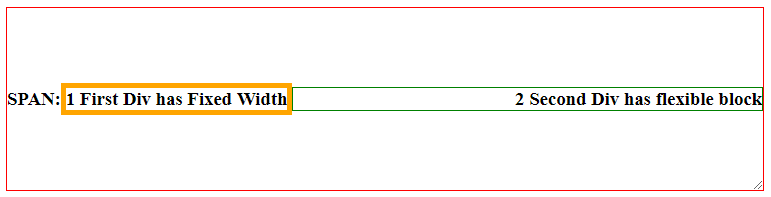
<h3>

<span>SPAN:</span>

<div>1 First Div has Fixed Width</div>

<div>2 Second Div has flexible block</div>

</h3>



调整宽度以后的效果：前面两个元素不可变，最后一个div宽度随之调整，并且文字超界限省略号



CSS 小技巧：

* 如何通过CSS 删除掉排列元素之间的空格 “ ”，由于回车和多个空格(如果不是&nbsp;) 处理是当作一个空格“ ”。

看看以下例子：

ul {

list-style: none;

border:    3px solid purple;

padding:    0px;

}

ul > li {

    display:    inline-block;

border:     3px solid orange;

}

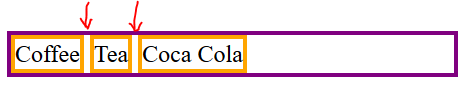
<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>



ul {

list-style: none;

border:    3px solid purple;

padding:    0px;

font-size: 0px; - 字体大小为 0，则空格不占空间

}

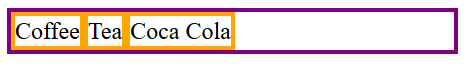
ul > li {

    display:    inline-block;

border:     3px solid orange;

font-size: initial; - 字体大小重新设置新值，或者使用默认值

}



* 如何在区块元素里将其他元素，水平居中，垂直居中

水平居中比较容易实现： text-align

方案一：

div.block {

    display:    block;

position:   relative;

margin:     auto;

width:      400px;

height:     400px;

border:     3px solid orange;

text-align: center;

}

div.block > span,

div.block > div

{

    display:    inline-block;

position:   relative;

    border:     2px solid green;

top:        50%;

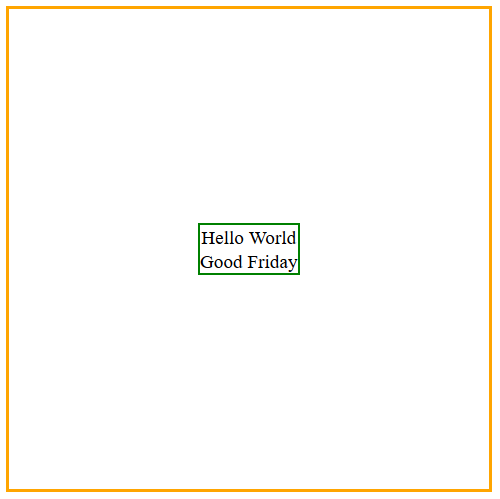
transform: translateY(-50%);

}

<div class="block">

<div>Hello World<br>Good Friday</div>

</div>



方案二：

div.block {

    display:    block;

position:   relative;

margin:     auto;

width:      400px;

height:     400px;

border:     3px solid orange;

}

div.block > span,

div.block > div

{

    display:    inline-block;

position:   absolute; - 也可以使用 absolute

    border:     2px solid green;

top:        50%; - 这是 container 高度的 50%

left:       50%; - 这是 container 宽度的 50%

transform: translate(-50%, -50%); - 这是元素本身高度和宽度的 50% 的位移量

}

结果如上图：

方案三：

<div **align="center"** style="**display:table-cell; vertical-align:middle;** width:300px;height:260px;border:1px solid green;">

<div style="display:block; text-align:initial; width:150px; height:180px; border:1px solid blue;">Hello</div>

</div>

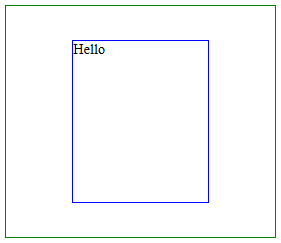
方案四：

<div style="**display:flex; justify-content:center; align-items:center;** width:300px;height:260px;border:1px solid green;">

<div style="display:block; width:150px; height:180px; border:1px solid blue;">Hello</div>

</div>

结果如图：



@charset "charset";

@charset "UTF-8";

@media

@media not|only *mediatype* and *(media feature* and|or|not *mediafeature)* { *CSS-Code;*}

根据不同的媒体加载不同的样式表：

<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">  
<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">

MediaType：

