div:not([id="footer"]){

background: orange;

}

除了footer以外

:empty选择器表示的就是空。

用来选择没有任何内容的元素，这里没有内容指的是一点内容都没有，**哪怕是一个空格**。

#brand:target p{

background: orange;

color: #fff;

}

对应id=”brand”的活动的P元素

选择器：子类的选择

ol > li:nth-child(odd){

background: green;

}

ol > li:nth-child(2n) ol > li:nth-child(1) ol > li:nth-child(even)

.wrapper > p:last-of-type {

background: orange;

}

最后一个p元素 :last-of-type :nth-of-type(odd/even/2n)

div:nth-last-of-type(5) 倒数第五个div

:only-child 唯一的一个子元素

p:only-of-type 唯一的一个p类型的子元素（可以有其他类型子元素）

input[type="text"]:disabled{

box-shadow: none;

}

“:disabled”选择器刚好与“:enabled”选择器相反，用来选择不可用表单元素。要正常使用“:disabled”选择器，需要在表单元素的HTML中设置“disabled”属性。

input[type="radio"]:checked + span {

opacity: 1;

}

表示选中状态

::selection

突出显示文本

“**:read-only**”伪类选择器用来指定处于只读状态元素的样式。简单点理解就是，元素中设置了“**readonly=’readonly’**

**::input-placeholder{color: blue} input提示文字的颜色**

**:read-write** 选择器来设置不是只读控件的文本框样式。与read-only相反

**旋转rotate(**20deg**) 函数**通过指定的角度参数使元素相对原点进行旋转

扭曲skew(20deg)函数能够让元素**倾斜显示**

**transform: scale(1.5,1); scaleX() 缩放**

**translate()函数**可以将元素向指定的方向移动，类似于position中的**relative**。或以简单的理解为，使用translate()函数，可以把元素从原来的位置移动，而不影响在X、Y轴上的任何Web组件。

transform:translate(-50%,-50%); 居中

transform:translate(-50px,-50px);

transform: matrix(1,0,0,1,100,100); 矩阵

transform-origin改变元素原点位置

div {

width: 200px;

height: 200px;

transition-property: width,height;

transition-duration:.5s;

transition-timing-function: ease-in;

transition-delay:.18s;

}

div:hover {

width: 400px;

height:800px;

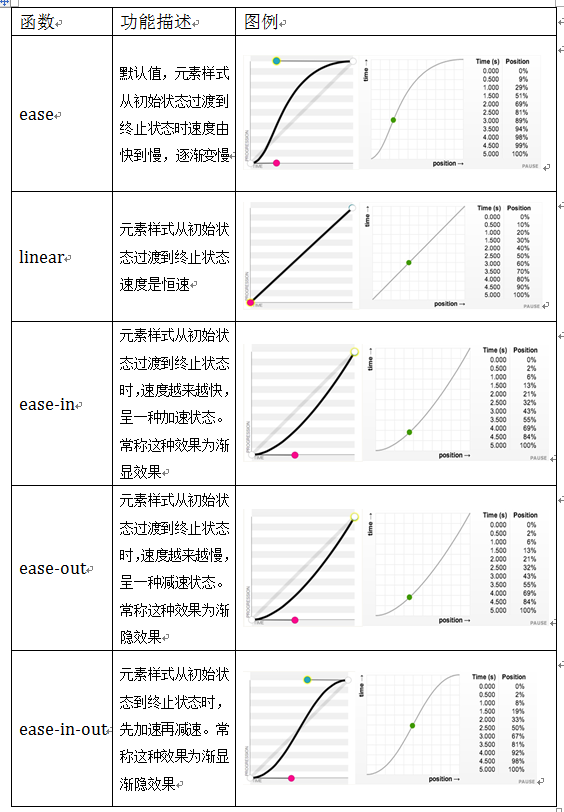
}

transition-property: 指定过渡或动态模拟的CSS属性

transition-duration: 指定完成过渡所需的时间

transition-timing-function: 指定过渡函数

transition-delay: 指定开始出现的延迟时间



transition-delay属性和transition-duration属性极其类似，不同的是transition-duration是用来设置过渡动画的持续时间，而transition-delay主要用来指定一个动画开始执行的时间，也就是说当改变元素属性值后多长时间开始执行。

有时我们想改变两个或者多个css属性的transition效果时，只要把几个transition的声明串在一起，用逗号（“，”）隔开，然后各自可以有各自不同的延续时间和其时间的速率变换方式。但需要值得注意的一点：第一个时间的值为 transition-duration，第二个为transition-delay。

例如：a{ transition: background 0.8s ease-in 0.3,color 0.6s ease-out 0.3;}

Keyframes被称为关键帧，其类似于Flash中的关键帧。在CSS3中其主要以“@keyframes”开头，后面紧跟着是动画名称加上一对花括号“{…}”，括号中就是一些不同时间段样式规则。

@keyframes changeColor{

0%{

background: red;

}

100%{

background: green;

}

}

div:hover {

animation-name: changeColor; 调用的动画

animation-duration: 5s; 设置动画完成时

animation-timing-function: ease-out; 动画播放模式

animation-delay: .1s; 开始时间

animation-iteration-count: infinite; 播放次数

animation-direction: alternate; 动画方向交替，normal向前

animation-play-state: running; 播放状态，running和paused

animation-fill-mode: both; 动画时间外属性

}

animation-fill-mode属性定义在动画开始之前和结束之后发生的操作。主要具有四个属性值：none、forwards、backwords和both。其四个属性值对应效果如下：

|  |  |
| --- | --- |
| 属性值 | **效果** |
| none | 默认值，表示动画将按预期进行和结束，在动画完成其最后一帧时，动画会反转到初始帧处 |
| forwards | 表示动画在结束后继续应用最后的关键帧的位置 |
| backwards | 会在向元素应用动画样式时迅速应用动画的初始帧 |
| both | 元素动画同时具有forwards和backwards效果 |

在默认情况之下，动画不会影响它的关键帧之外的属性，使用animation-fill-mode属性可以修改动画的默认行为。简单的说就是告诉动画在第一关键帧上等待动画开始，或者在动画结束时停在最后一个关键帧上而不回到动画的第一帧上。或者同时具有这两个效果。

columns: 200px 2; 多列布局模块，宽度和显示列数

column-width: 500px; auto | <length> 宽度

column-count：auto | <integer> 最大列数，auto默认1

column-gap: normal | <length> 列与列的间距，normal默认1em

column-rule:<column-rule-width>|<column-rule-style>|<column-rule-color>



column-span: none | all 垮列设置

box-sizing: content-box | border-box | inherit

## 容器的flexbox属性

* flex-direction flex容器的主轴被定义为与文本方向相同
* flex-wrap 这个属性主要是设置container中的items是否会换行。
* flex-flow
* justify-content 主要用设置flex items在容器里面严着主轴的排列方式
* align-items 当flex items在主轴上只有一排时，align-items属性主要用于设置交叉轴上flex items的排列方式
* align-content 当flex items在主轴上有多排(只有一排时此属性不起作用)时，align-content属性主要用于设置交叉轴上flex items的排列方式。
*  

## 项目的flexbox属性

* Order 编号
* flex-grow 比例放大
* flex-shrink 比例缩小
* flex-basis auto | <width> 制定大小 auto默认 1
* Flex flex-grow shrink basis 的缩写 none|auto|[<flex-shrink><-shrink>?||<flex-basis>] none默认 1 1 auto ； auto默认 0 0 auto
* align-self 不想跟别人一样 align-self: auto | flex-start | flex-end | center | baseline | stretch;

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>flexbox-model</title>

<style>

.flex-item1 {

width: 100px;

height: 100px;

background-color: white;

margin: 20px;

order: 1;

flex-grow: 1;

}

.flex-item2 {

width: 100px;

height: 100px;

background-color: white;

margin: 20px;

order: 0;

flex-grow: 2;

}

.flex-item3 {

width: 100px;

height: 100px;

background-color: white;

margin: 20px;

order: 3;

flex-grow: 3;

}

.flex-container {

background-color: #FECE3F;

width: 500px;

height: 420px;

display: flex;

-webkit-flex-direction: row;

flex-wrap: wrap;

align-items: ;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item1">1</div>

<div class="flex-item2">2</div>

<div class="flex-item3">3</div>

</div>

</body>

</html>

测试代码：

@media是CSS3中新引进的一个特性，被称为媒体查询。在页面中也可以通过这个属性来引入媒体类型。@media引入媒体类型和@import有点类似也具有两方式。

1. 在样式文件中引用媒体类型：

@media screen {

选择器{/\*你的样式代码写在这里…\*/}

}

1. 使用@media引入媒体类型的方式是在<head>标签中的<style>中引用。

<head>

<style type="text/css">

@media screen{

选择器{/\*你的样式代码写在这里…\*/}

}

</style>

</head>