

第四篇 交互动效设计

Contents

01.变换

transform

02.动画

animation

03.动效案例

case

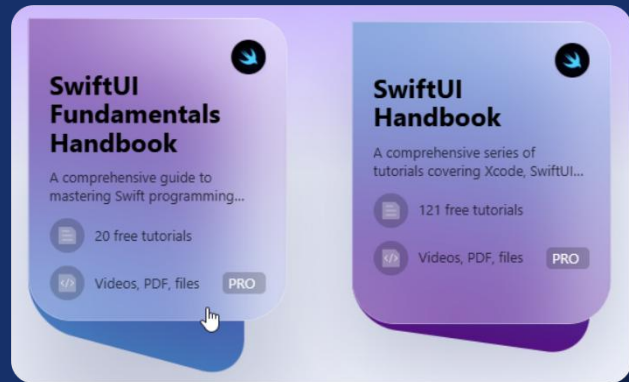
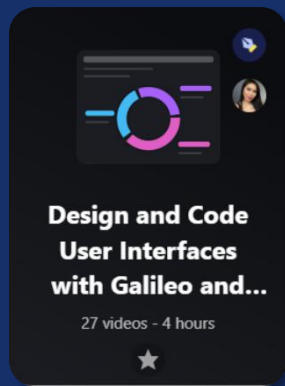
变换

transform~

CSS `transform` 是元素进行 2D/3D 变换 的核心属性，支持平移、旋转、缩放、倾斜等效果，且**不破坏**原有文档流布局。

场景：

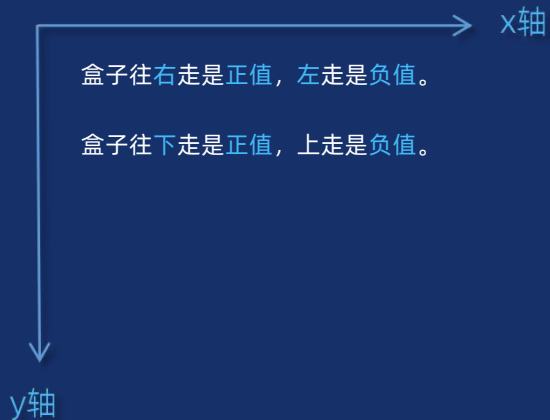
应用非常广泛，通过简洁的语法实现复杂的视觉动态效果。



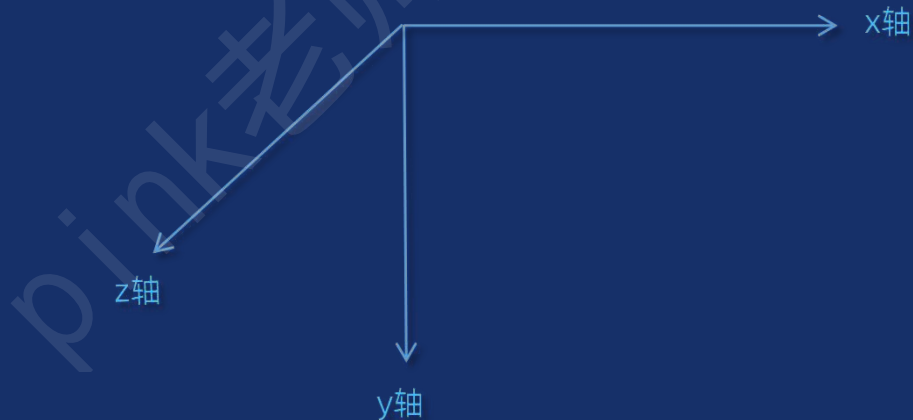
坐标系

x y z ~

变换的时候很多情况需要按照坐标轴来变化，介绍2D坐标系和3D坐标系。



2D坐标系



3D坐标系

基础变换函数-2D

transform~

transform 是用于对元素进行 2D/3D 变换 的核心属性，支持平移、旋转、缩放、倾斜等效果，且**不破坏**原有文档流布局。

变换类型	函数语法	核心描述	应用场景
平移	<code>translate(x, y)</code> <code>translateX(x)</code> <code>translateY(y)</code>	沿 X/Y 轴移动元素，参数可为像素或百分比（百分比相对于元素自身尺寸）	微调元素位置、悬停偏移效果
旋转	<code>rotate(angle)</code>	以元素中心为基点顺时针旋转，需加单位 deg（负值为逆时针）	图标旋转、卡片翻转
缩放	<code>scale(sx, sy)</code> <code>scaleX(sx)</code> <code>scaleY(sy)</code>	X/Y 轴按比例缩放，单参数时等比例缩放（>1 放大，<1 缩小，支持百分比）	悬停放大、焦点突出
倾斜	<code>skew(x-angle, y-angle)</code> <code>skewX(angle)</code> <code>skewY(angle)</code>	沿 X/Y 轴扭曲元素形状，参数为倾斜角度（支持负值）	斜切导航栏、动态图表

体验网站：<https://css-transform.moro.es/>

1. 变换函数- 2D平移

transform~

平移 (translate) 沿 X/Y 轴移动元素位置, 不改变元素的实际布局 (原位置仍保留空白)

场景:

1. 悬停元素微调。鼠标放入元素上下或者左右移动 (添加过渡更优雅~)
2. 元素居中。元素实现水平垂直居中。

语法:

```
transform: translate(20px, 30px); /* x y 同时移动 */  
transform: translateX(50%); /* X 轴移动 */  
transform: translateY(-50%); /* Y 轴移动 */
```

- ❑ 平移不影响页面布局, 仅是视觉上的移动。
- ❑ 添加鼠标经过元素移动, 优先transform而不是通过left、top等, 性能更佳。
- ❑ 如果单位是百分比, 相对于元素自身尺寸, 而非父容器
结合 top: 50%; left: 50% 与 translate(-50%, -50%) 实现元素居中



Redmi 14C

【持久续航】5160mAh 大电池

499元起



王者荣耀

MOBA | 国民MOBA手游大作



和平精英

射击 | 反恐军事竞技体验类国产手游

小米上新
米家净水器Pro 1200G
创新双龙头 净饮更健康

活动价 ¥2599

立即查看



1.变换函数- 2D平移

transform~

案例：



要求：

1. 鼠标经过显示绿色边框。
2. 底部找相似绿色盒子升上来。

思路：

1. 原先盒子有2像素透明边框，鼠标经过边框改绿色。
2. 底部找相似绿色盒子定位到盒子最下面看不见，鼠标经过transform升上来。
3. 注意，鼠标经过的是 li，但是做移动的是绿色盒子。

2.变换函数-2D旋转

transform~

旋转 (rotate) 通过改变元素在平面或空间中的角度实现视觉效果。

场景：

1. 悬停动画（如按钮旋转）
2. 加载动画（无限循环旋转）

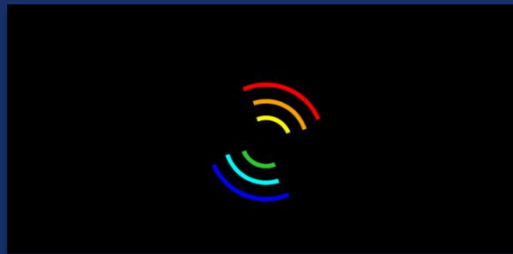
语法：

```
transform: rotate(360deg);/* 旋转 */
```

- 参数单位：deg（度），正值是顺时针，负值是逆时针

```
transform-origin: left top;/* 旋转的基点 */
```

- transform-origin 设置旋转中心点。
- 属性值支持left、top等，也可以支持数字比如像素和百分比等。



2.变换函数-2D旋转

transform~

旋转 (rotate) 通过改变元素在平面或空间中的角度实现视觉效果。

注意：

行内元素的布局特性（无法设置宽高、盒模型限制、transform 基准异常）会破坏旋转效果的稳定性和精确性，所以文字类要转换行内块或者块级元素。比如字体图标需要转换。

3. 变换函数-2D缩放

transform~

缩放 (`scale`) 用于调整元素尺寸, 且**不改变**元素在文档流中的原始占位。

场景:

1. 悬停放大等

语法:

```
transform: scale(1.5); /* 放大 */  
transform: scale(1.5, 1); /* 放大 */
```

- ❑ 单参数: 同时作用于 X 和 Y 轴 (如 `scale(2)` 表示**整体放大 2 倍**)。
- ❑ 双参数: 第一个控制 X 轴, 第二个控制 Y 轴 (如 `scale(0.5, 1.5)` 横向缩小 50%、纵向放大 50%)

<https://game.qq.com/web202406/#/pc/join-us>

探索vivo



专业设计师在**花瓣**发现、收集灵感

节日热点

行业灵感

设计趋势



4. 变换函数-2D倾斜

transform~

倾斜 ([skew](#)) 用于对元素进行 二维倾斜变换, 通过沿 X 轴或 Y 轴扭曲元素的几何形状。

场景:

1. 鼠标经过元素倾斜效果。

语法:

```
transform: skew(30deg, 30deg); /* 倾斜 */  
transform: skewX(30deg); /* 倾斜 */  
transform: skewY(30deg); /* 倾斜 */
```

- ❑ 如果只写一个参数则Y轴默认为0
- ❑ transform-origin 设置倾斜中心点

查看所有英雄 >



过渡进阶

transition~

transition 完整写法:

transition: 过渡属性 持续时间 速度曲线 延迟时间;

```
transition: all 1s linear 1s;
```

所有属性添加过渡效果, 过渡持续1秒, 匀速, 延迟1秒执行


1.cubic-bezier.com: 可视化编辑贝塞尔曲线。


2.easings.net: 获取常用缓动函数的预设值


3.谷歌浏览器调试





速度曲线参数	效果描述
ease	默认值, 慢速开始 → 加速 → 慢速结束, 适合大多数自然过渡 (如按钮悬停)。
linear	匀速运动, 无加速/减速, 适合机械感效果 (如进度条)。
ease-in	慢速开始 → 逐渐加速, 适合元素从静止启动 (如弹窗出现)。
ease-out	快速开始 → 逐渐减速, 适合元素退出 (如关闭动画)。
ease-in-out	慢速开始和结束, 中间加速, 适合对称性动画 (如页面切换)。
cubic-bezier(x1,y1,x2,y2)	贝塞尔曲线。通过四值 (0-1范围内) 自定义速度曲线, 实现弹跳、骤停等创意效果 (如按钮点击反馈)。


ease


linear


ease-in


ease-out


ease-in-out

变换函数- 复合写法

transform~

在 CSS 中, transform 属性的复合写法 (多个变换函数组合使用)

语法:

```
transform: A() B() C() ;
```

顺序:

核心规则: 从右到左的执行顺序

实际的执行顺序是: 先执行最右侧的 C(), 然后是 B(), 最后是左侧的 A()。

变换函数- 汽车案例

transform~

分析：



变换函数- 汽车案例

transform~

分析：



变换函数- 汽车案例

transform~

分析：



变换函数- 汽车案例

transform~

分析：



1. 汽车盒子包含汽车图片和轮子图片，轮子调整大小利用定位放到对应位置

变换函数- 汽车案例

transform~

分析：



1. 汽车盒子包含汽车图片和轮子图片，轮子调整大小利用定位放到对应位置

变换函数- 汽车案例

transform~

分析：



1. 汽车盒子包含汽车图片和轮子图片，轮子调整大小利用定位放到对应位置
2. 汽车盒子做平移效果，轮子图片做旋转效果

3D 变换与透视

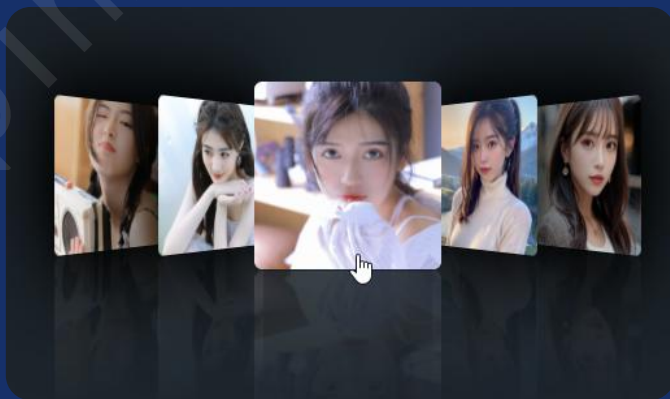
3D~

CSS 3D 效果通过将二维元素在三维空间中进行变换，为网页添加立体感和动态交互体验。

场景：

1. 卡片翻转效果。
2. 3D 幻灯片/轮播图。
3. 数据可视化。立体图表（如3D柱状图）等

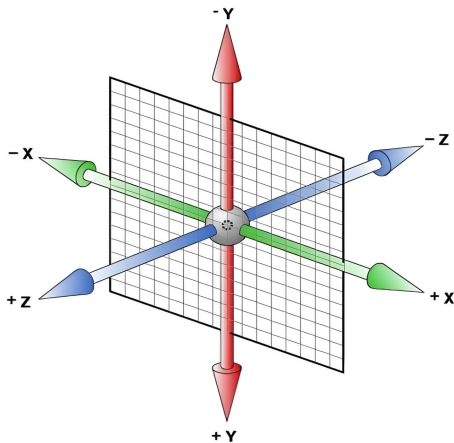
相比JavaScript或WebGL，CSS 3D利用GPU加速，动画更流畅，性能更高效。



三维坐标系

x y z~

网页中的三维坐标系：



坐标数值：

X轴代表左右方向。右边是正值，左边是负值。

Y轴代表上下方向。下方是正值，上方是负值。

Z轴代表远离/接近屏幕的方向。接近屏幕是正值，远离屏幕是负值。

可以利用 左手法则 记忆。

默认情况下，元素在二维平面上呈现，Z轴值为0。

当应用3D变换时，元素便可以在三维空间中自由变换。

3D旋转

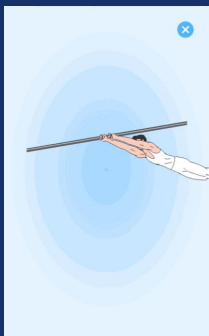
rotate~

旋转 (rotate) 通过改变元素在3D空间中的角度实现视觉效果。

语法：

```
transform: rotateX(45deg); /* 围绕X轴旋转 */  
transform: rotateY(45deg); /* 围绕Y轴旋转 */  
transform: rotateZ(45deg); /* 围绕Z轴旋转 */
```

- 参数单位：deg（度），正值是顺时针旋转，负值是逆时针旋转
- rotateZ和二维里面的rotate() 一样的



rotateX



rotateY

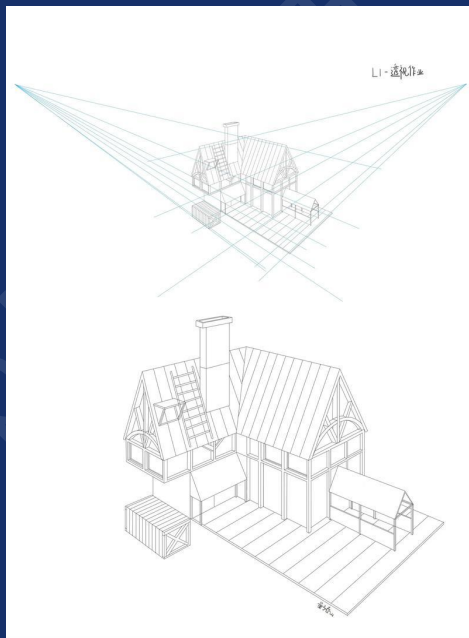
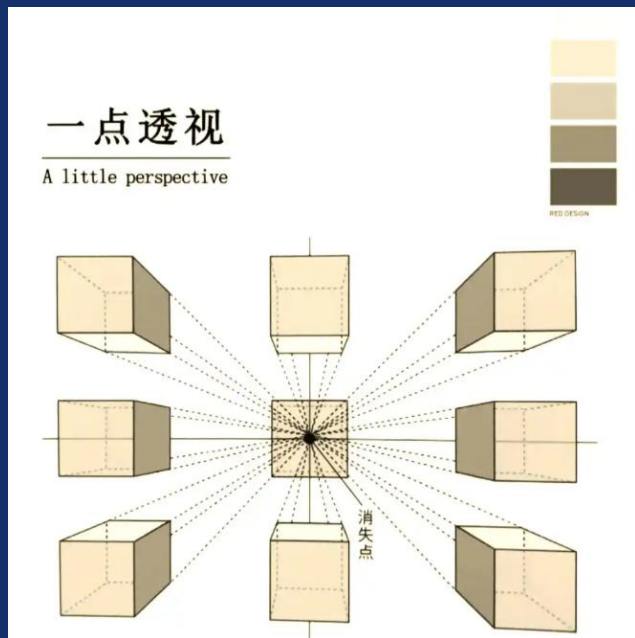


rotateZ

透视 (Perspective)

Perspective~

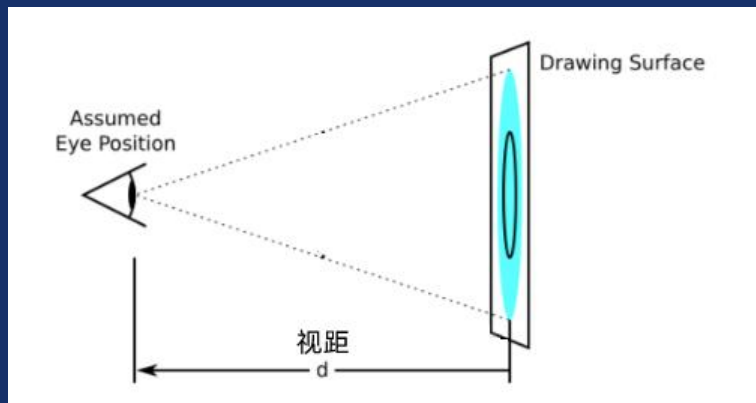
“透视” (perspective) 一词源于拉丁文 “perspicere” (看透)，指在平面上描绘物体的空间关系的方法或技术。



透视 (Perspective)

Perspective~

在 CSS 中，透视效果 (Perspective) 用于模拟人眼观察 3D 空间时的近大远小效果。



语法：

```
perspective: 1000px; /* 透视效果 */
```

- ❑ 数值越小，透视效果越强。
- ❑ 给父元素添加，里面所有子元素都会添加透视效果。（常用）
- ❑ 给子元素添加，当前元素添加透视效果。

```
transform: perspective(1000px) rotateX(45deg); /* 同时使用多个属性 */
```

- ❑ 注意：perspective() 必须作为 transform 属性的第一个函数（否则无效）

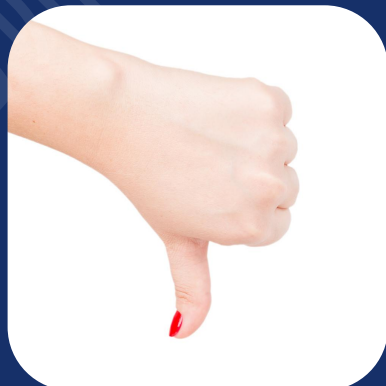
透视 (Perspective)

Perspective~

3D旋转方向： 还是左手



`rotateX(360deg)` 正值是四指指向的方向
负值反方向



`rotateY(360deg)` 正值是四指指向的方向
负值反方向

3D旋转-两面翻转的盒子



3D旋转-两面翻转的盒子



box父盒子



3D旋转-两面翻转的盒子



box父盒子



3D旋转-两面翻转的盒子



语法：

```
backface-visibility: hidden;
```

- 控制元素背面的可见性（默认镜像显示），常用于隐藏背面（如扑克牌翻转效果）

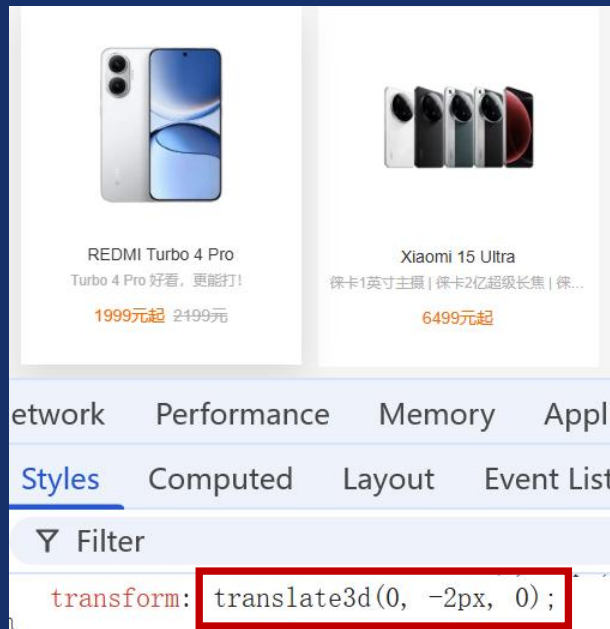
3D位移

CSS 3D 平移函数 `translateZ()` 与 `translate3d()`。

语法：

```
transform: translate3d(x, y, z);
```

- ❑ 在三维空间中同时沿 X、Y、Z 轴平移元素，支持更复杂的空间定位
- ❑ 位移建议通过 3D 变换强制启用 GPU 渲染，提升动画流畅度



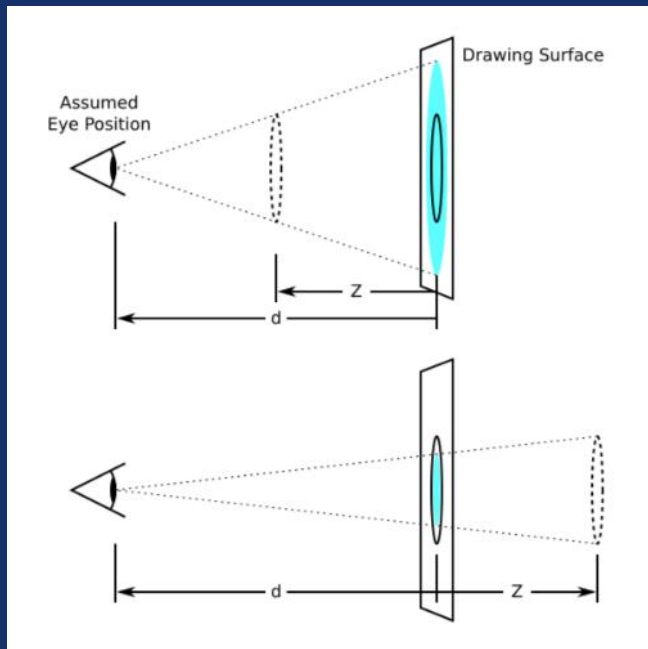
3D位移

CSS 3D 平移函数 `translateZ()` 与 `translate3d()`。

语法：

```
transform: translateZ(100px);
```

- 沿 Z 轴（垂直于屏幕方向）平移元素，实现近大远小的立体效果
- 正值元素靠近观察者（放大），负值远离观察者（缩小）
- 需父容器设置 `perspective` 属性才能生效，否则无视觉变化



开启3D空间

Perspective~

父容器需设置 `transform-style: preserve-3d;` 使子元素保留3D位置（如 3D 卡片翻转）。

语法：

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

❑ 默认`flat`会将子元素压平

Contents

01.变换

transform

02.动画

animation

03.动效案例

case

动画

animation~

CSS3 的 `animation` 是一种通过定义关键帧和动画属性来实现元素动态效果的技术。。

优势：

- ❑ 性能高效：通过浏览器原生支持，利用 GPU 加速渲染，避免频繁的 JavaScript 计算开销。
- ❑ 代码简洁：解决了传统 CSS 过渡（transition）只能定义两帧状态的局限性。可以利用关键帧实现更复杂动画。
- ❑ 交互增强：支持与用户操作（如悬停、点击）结合，提升网页的动态表现力和用户体验。



动画

animation~

CSS3 的 `animation` 是一种通过定义关键帧和动画属性来实现元素动态效果的技术。。

关键帧：

关键帧 (Keyframe) 是动画和视频制作中用于定义动作或状态变化的核心节点，它决定了动画的起始、转折和结束状态。中间帧 (插帧) 通过算法自动生成，形成连贯的动画效果。

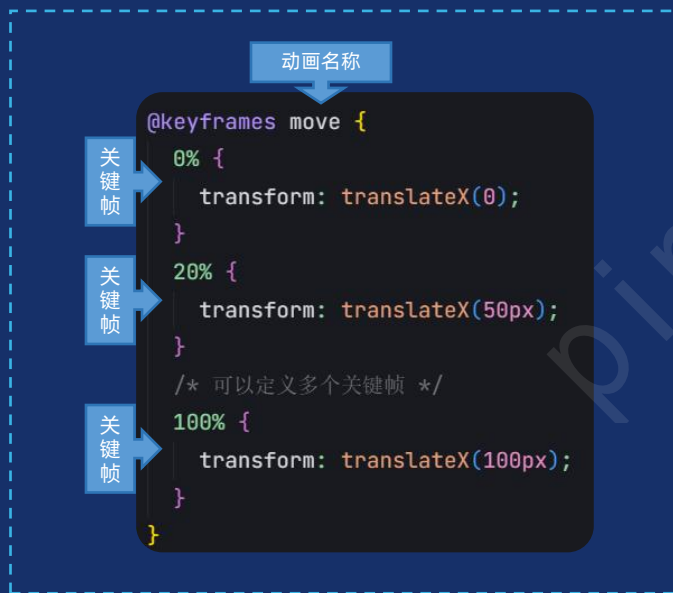


动画

animation~

动画使用：先定义动画，后使用动画。

1. 定义动画：



动画

animation~

动画使用：先定义动画，后使用动画。

2.使用动画：

animation: 动画名称 动画时长;

□ 动画属性要写到目标元素里面。

```
animation: move 1s;
```

```
@keyframes move {  
  from {  
    transform: translateX(0);  
  }  
  
  to {  
    transform: translateX(100px);  
  }  
}
```

动画

animation~

animation属性完整写法：

animation: 动画名称 动画时长 速度曲线 延迟时间 播放次数 播放方向 执行完毕状态;

- ❑ 动画名称和动画时长是必写，其余可以省略，但是要保证书写顺序。
- ❑ 动画属性要写到目标元素里面。

```
animation: move 1s;
```

```
@keyframes move {  
  from {  
    transform: translateX(0);  
  }  
  
  to {  
    transform: translateX(100px);  
  }  
}
```

子属性	默认值	说明
播放次数	1	播放次数，若需无限循环写 infinite
播放方向	normal	播放方向，可选 reverse（反向）、 alternate （交替）
执行完毕状态	none	动画结束后状态（如 forwards 保留最后一帧， backwards 回到第一帧）

动画

animation~

动画使用：单写属性

动画子属性	默认值	说明
animation-timing-function	ease	动画 速度曲线 ，支持 linear、cubic-bezier() 等
animation-delay	0s	动画 延迟时间 ，可设为负值（跳过部分动画）
animation-iteration-count	1	播放次数 ，若需无限循环需显式声明 infinite
animation-direction	normal	播放方向 ，可选 reverse（反向）、alternate（交替）
animation-fill-mode	none	执行完毕状态 ，动画结束后是否保留样式（如 forwards 保留最后一帧）
animation-play-state	running	暂停或者继续动画 ，需单独设置 paused 暂停动画

动画

animation~

案例：



动画

animation~

steps 逐帧动画：

速度曲线 除了常见的 linear、ease、cubic-bezier等，还包含 `steps()` 函数，`steps()` 是 CSS 动画中用于控制动画分段执行的计时函数，它通过将动画分割为离散的步骤，实现类似传统帧动画（逐帧动画）的跳跃效果。



<https://shouji.360.cn/v6/index.html>

动画

animation~

steps 逐帧动画：

速度曲线 除了常见的 linear、ease、cubic-bezier等，还包含 `steps()` 函数，`steps()` 是 CSS 动画中用于控制动画分段执行的计时函数，它通过将动画分割为离散的步骤，实现类似传统帧动画（逐帧动画）的跳跃效果。

语法：

```
animation: move 1s steps(8) infinite;
```

- `steps(步数)`，步数为正整数。
- 经常和背景图片(精灵图)来实现动画效果。



动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

逐帧动画原理：



1. 移动图片。其实是通过变化背景位置，移动距离就是图片宽度。
2. 图片有N个不同图形， steps(步数) 就写N

动画

animation~

案例: <https://shouji.360.cn/v6/index.html>



盒子大小:

宽度: 548px, 高度 513px

背景图片大小:

宽度 13700px, 高度 513px

背景图片一共有 25张 小图片组成



Contents

01.变换

transform

02.动画

animation

03.动效案例

case

动效案例1

case~

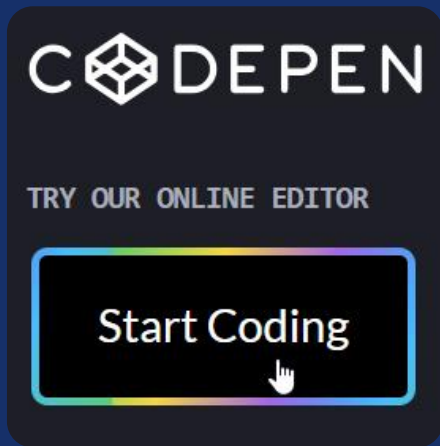
流光渐变边框效果：



动效案例1

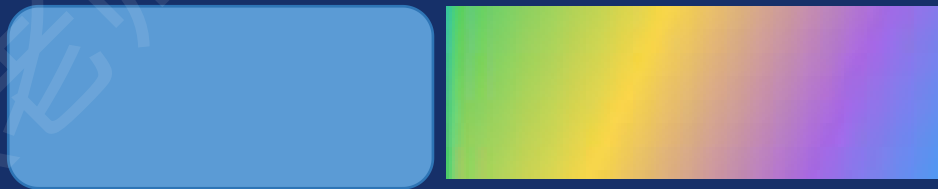
case~

流光渐变边框效果：



<https://codepen.io/>

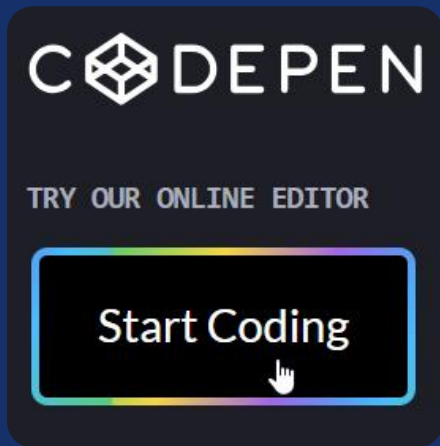
原理：



动效案例1

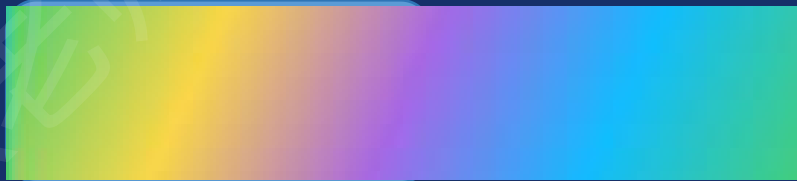
case~

流光渐变边框效果：



<https://codepen.io/>

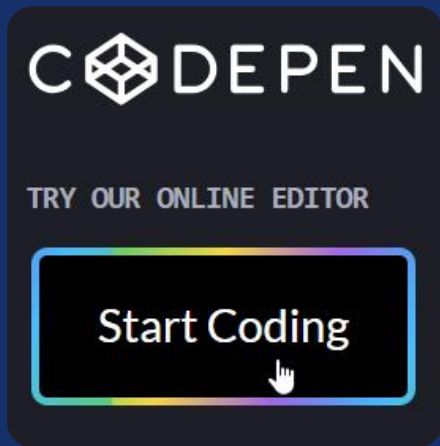
原理：



动效案例1

case~

流光渐变边框效果：



<https://codepen.io/>

原理：



3个元素：

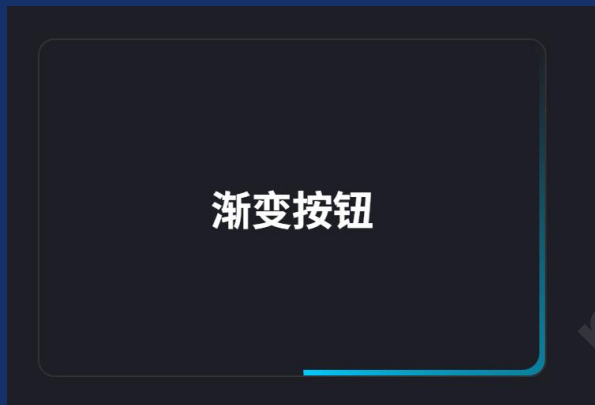
1. 大按钮（当爸爸）
2. 彩色盒子（做动画）
3. 文字盒子（做遮挡）

```
background: linear-gradient(115deg, #4fcf70, #fad648, #a767e5, #12bcfe, #44ce7b);
```

动效案例1

case~

流光渐变边框效果：



原理：



渐变按钮

动效案例1

case~

流光渐变边框效果：

原理：



4个元素：

1. 大盒子（当爸爸）
2. 径向渐变盒子（做动画）
3. 黑色遮挡盒子
4. 内容盒子（可能内容较多，不适合遮挡）

background-image: conic-gradient(transparent, transparent, transparent, #00ccff);

动效案例1

case~

流光渐变边框效果：



新知识：

`inset: 3px;` 针对于定位，写法更简单

等价于 `top: 3px; left: 3px; right: 3px; bottom: 3px;`

`z-index: -1;` 可以被标准流盒子压住。

动效案例1

case~

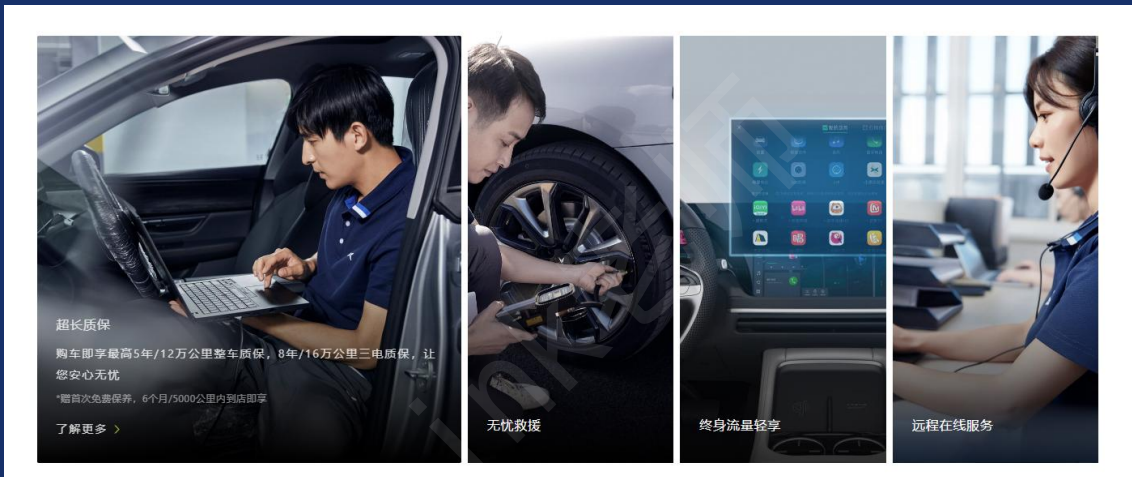
作业：

单边流光效果

霓虹描边效果

动效案例2

case~



<https://www.xiaopeng.com/customerservice.html>

动效案例2

case~



实现思路:

1. 布局可以使用 `grid` 分成几等份。
2. 鼠标经过某个子盒子, 修改份数。

新知识:

`has()` 选择器;

`:has()` 允许你“根据子元素的特征反向选择父元素”也被称为“父选择器”或“存在选择器”

```
.box ul:has(.item:nth-child(1):hover) {  
  grid-template-columns: 2fr 1fr 1fr 1fr;  
}
```

动效案例3

case~

核心:

1. 多个元素, 利用定位分别移动到目标位置。
2. 元素开始是按照 rotateX Y 拉伸 (或者斜切也可以)
3. 鼠标经过, 每个盒子去掉倾斜效果, 利用平移微调位置即可。

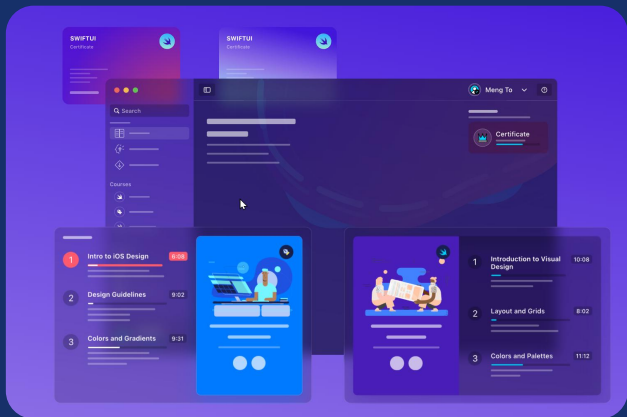
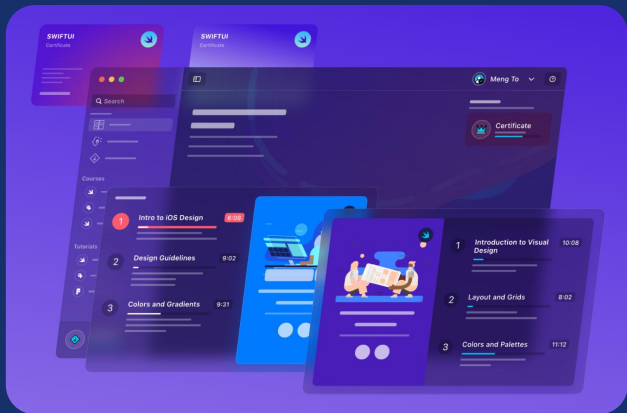
新知识:

```
backdrop-filter: blur(10px);
```

backdrop-filter 滤镜函数对元素背后的内容 (如背景图像、文本或其他元素) 进行实时处理, 而不会改变元素本身的样式。

通过 blur() 实现半透明背景的模糊效果, 数值越大越模糊, 比如毛玻璃效果

```
transition: 2s cubic-bezier(0.075, 0.82, 0.165, 1);
```



动效案例4

case~

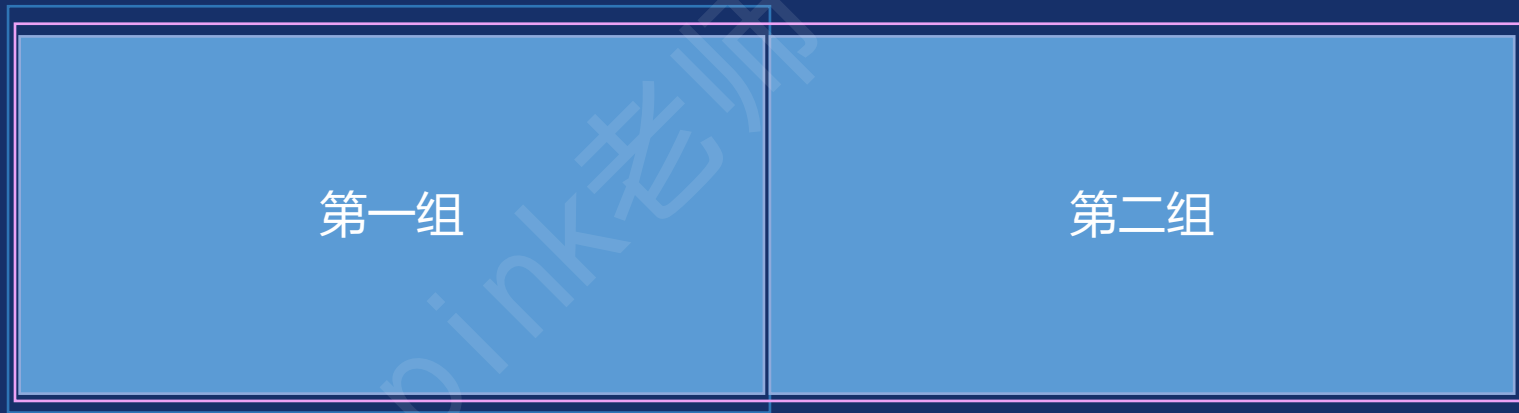


<https://retail.ele.me/?spm=a2f95.17632747.0.0.501a30d76wiyStl>

动效案例4

case~

二号盒子，负责移动

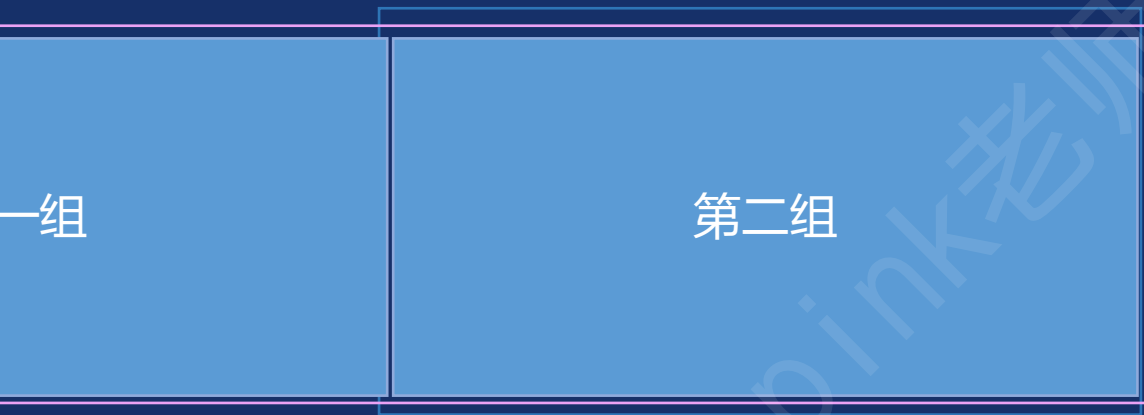


box 父盒子

动效案例4

case~

二号盒子，负责移动

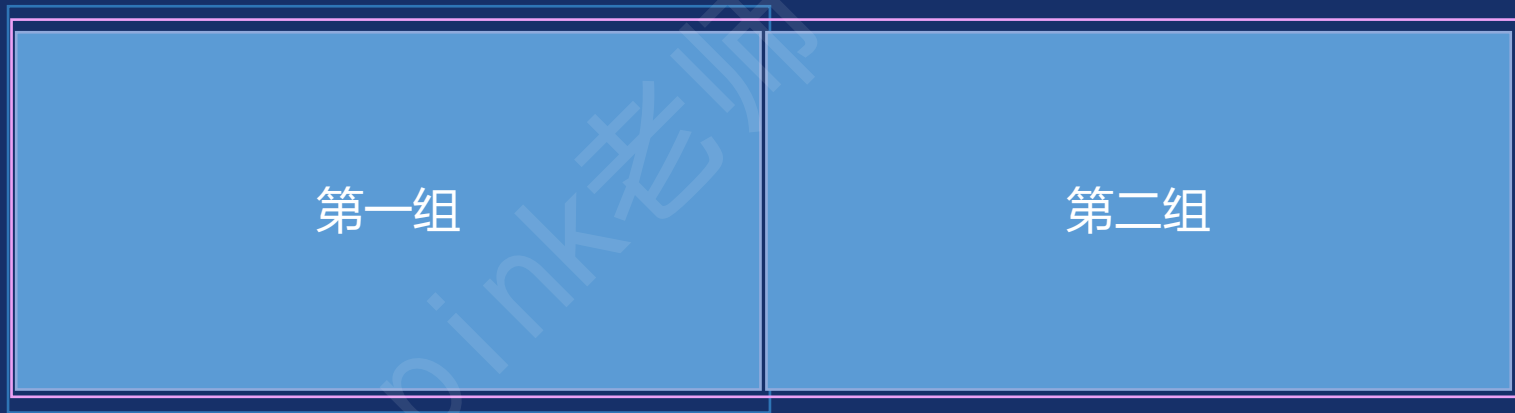


box 父盒子

动效案例4

case~

二号盒子，负责移动



box 父盒子

动效案例5

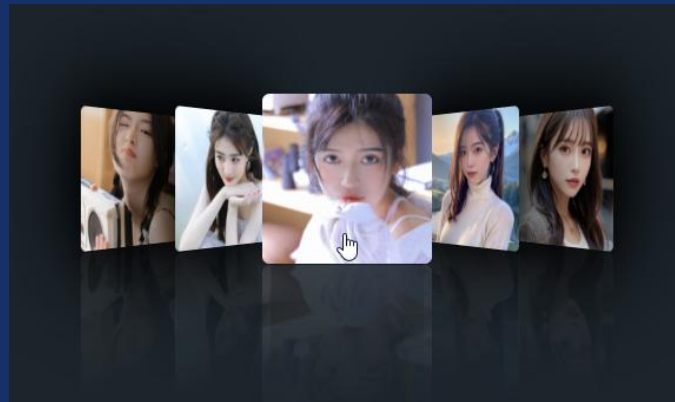
case~



<https://www.vivo.com.cn/vivo/s20pro/>

动效案例5

case~



/* 添加倒影 */

```
-webkit-box-reflect: below 1px linear-gradient(transparent, □#0002);
```

新知识：

`box-reflect` 是CSS 中用于为元素创建倒影效果。

语法：

`box-reflect:` 倒影方向 倒影距离 倒影图像;

①倒影方向（必写）：

- `above`：倒影在元素上方（顶部）。
- `below`：倒影在元素下方（底部，最常用）。
- `left`：倒影在元素左侧。
- `right`：倒影在元素右侧。

② 倒影距离（必写）：

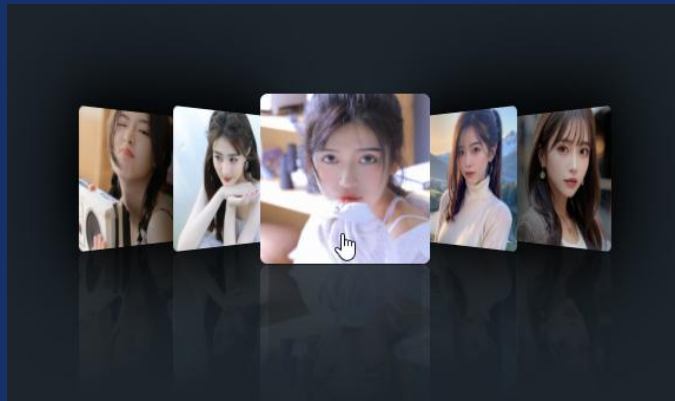
③ 倒影图像（可选）：

常用线性渐变（`linear-gradient`）实现从清晰到透明的过渡，避免倒影边缘生硬

注意：因为兼容性问题，前面需要加 `-webkit-`

动效案例5

case~



新知识:

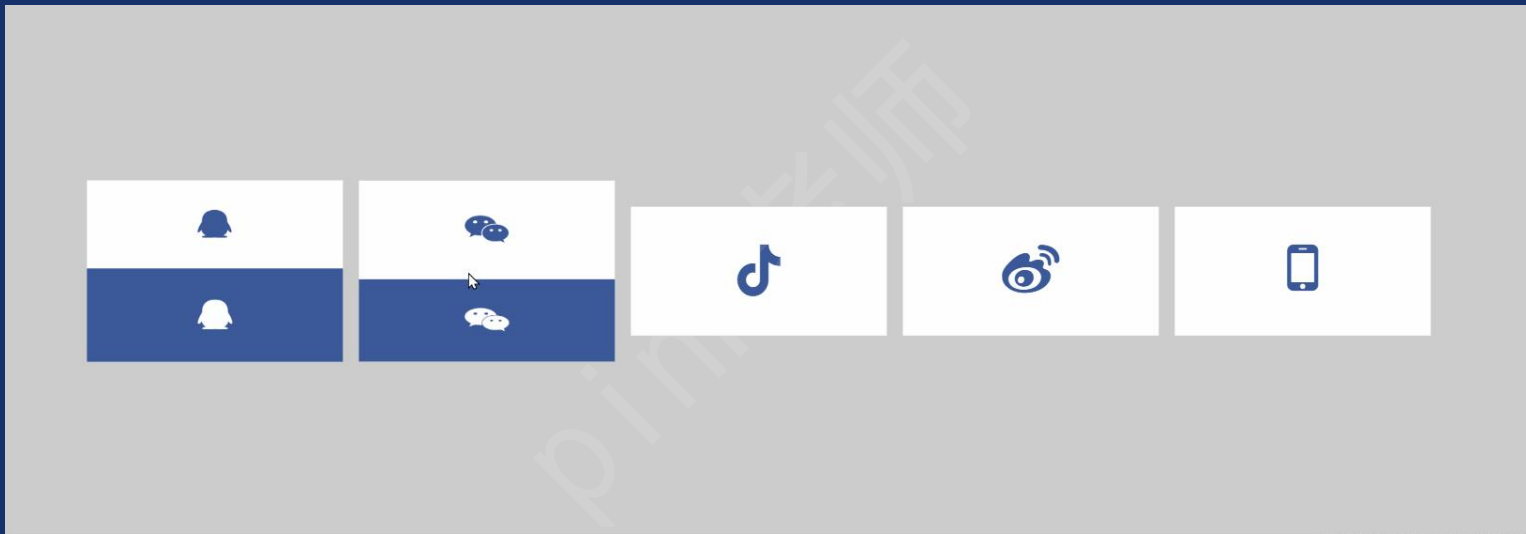
:not(); 否定选择器

作用是“排除”符合某些条件的元素，让样式作用于剩余元素。

```
.box:hover>.item:not(:hover) {  
  margin: 0 -20px;  
  transform: perspective(500px) rotateY(45deg) scale(0.95);  
}
```

动效案例6

case~



3D导航栏

动效案例6

case~

前面盒子: `translateZ 20px`

底面盒子: `translateY 20px` 和 `rotateX -90度`

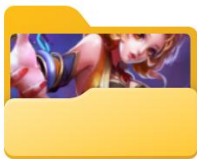
大的盒子: `rotateX 90度`

作业安排

case~



01-翻转卡片



05-王者翻转框



06-悬停侧边栏-代码



02-流光线条的效果.html



03-翻转的盒子.html



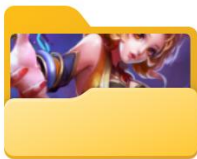
04-悬停导航栏.html

作业安排

case~



01-翻转卡片



05-王者翻转框



06-悬停侧边栏-代码



02-流光线条的效果.html



03-翻转的盒子.html



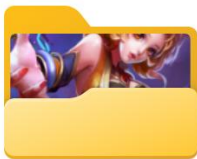
04-悬停导航栏.html

作业安排

case~



01-翻转卡片



05-王者翻转框



06-悬停侧边栏-代码



02-流光线条的效果.html



03-翻转的盒子.html



04-悬停导航栏.html



T H A N K Y O U

行无止境