mask

خاصية القناع أو الحاجز، تقوم بقص خلفية العنصر، حسب المناطق الملونة والمناطق الشفافة في صورة معينة أو تدرج لوني معين، حيث تكون الأجزاء الملونة من صورة القناع هي المناطق المقصوصة على أساسها الخلفية أما المناطق الشفافة فتظهر شفافة في خلفية العنصر ليظهر أجزاء من العنصر الذي خلفه:

ملف html

```
<di> </div>
```

ملف css

```
div{
    width: 180px;
    height: 220px;
    padding: 20px;
    margin: 20px;
    background:url(roses.jpg) center/50px;
    mask: url(pic.png) center/contain ;
    }
```

background

mask-image

mask

لاحظ أن خلفية العنصر أصبحت مقصوصة عن طريق الأجزاء ذات اللون الأسود بينما الأجزاء الشفافة التي تظهر باللون الأبيض اختفت من خلفية العنصرِ.

والخاصية mask تعتبر اختصاراً لعدة خصائص فرعية أخرى وقيمتها تجمع قيم كل هذه الخصائص أو بعضها وهذه الخصائص كالتالي:

mask-image

الصورة التي تستخدم في قص خلفية عنصر آخر، والصورة نوعان، الأول يسمي image-source حيث يمكن أن تكون الصورة بتنسيق png أو svg أو أي تنسيق يحتوي مناطق شفافة كما أنها قد تكون تدرج لوني، ملون في بعض أجزائه وشفافاً في أجزاء أخرى ، والنوع الثاني لصورة القناع يسمى mask-source حيث تكون صورة القناع عنصر mask داخل العنصر svg.

mask-position

خاصية تحدد موقع أو مكان صورة القناع أو التدرج اللوني على المحورين الأفقي والرأسي لخلفية العنصر، والخاصية غير وراثية ولها قيمة افتراضية center وتقبل قيمتين، الأولى تحدد موقع صورة القناع على المحور الأفقي والثانية تحدد موقعها على المحور الرأسي ويمكن الاكتفاء بقيمة واحدة فقط إذا تساوت القيمتين، والقيم التي تقبلها الخاصية كالتالي:

على المحور الأفقي: right | center | left | percentage | value

على المحور الرأسي: top | center | bottom| percentage | value

وقد شرحنا تأثير وسلوك هذه القيم في الخاصية background-position.

mask-size

خاصية تحدد حجم صورة القناع أو حجم التدرج اللوني والخاصية غير وراثية ولها قيمة افتراضية auto وتقبل عدة أنواع من القيم كالتالي:

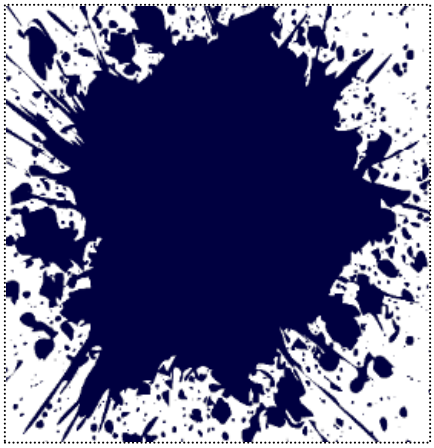
auto | cover | contain | percentage | value

وقد شرحنا تأثير وسلوك هذه القيم في الخاصية background-size.

قد تحتاج لتعيين الخصائص الثلاث السابقة، خاصة أبعاد صورة القناع في الغالب لن تتماشى مع أبعاد العنصر الذي ستقوم بقصه، لذل يجب تجهيز صورة القناع من حيث مكانها بالنسبة للعنصر وحجمها الذي سيغطي العنصر كالتالي:

لاحظ في الكود التالي أن قمنا بتعيين الثلاث خصائص

```
mask-image: url(stain.png);
mask-position: right  bottom;
mask-size: contain;
```



لاحظ أن خلفية العنصر تم قصها حسب المناطق الملونة في صورة القناع بينما المناطق الشفافة ظهرت باللون الأبيض (خلفية الصفحة).

ويمكن لصورة القناع أن تكون تدرج لوني من أي نوع كالتالي:

```
mask-image: linear-gradient(to right,black,transparent);
```

linear

radial

conic

أما النوع الثاني لصورة القناع وهو ما يطلق عليه mask-source فقد تكون عنصر mask داخل العنصر svg كالتالي:

ملف html

```
<di> </div>
<svg>
    <mask id="mymask">
       <polygon points="200 25,375 200,200 375,0 200"
                fill="white"/>
    </mask>
</svg>
```

ملف css

```
div{
    width: 180px;
    height: 220px;
    background: url(wood.jpg) center/cover;
    mask-image: url(#mymask);
    mask-position: center;
    mask-size: contain;
    }
```

background

mask

mask-type

خاصية توضح نوع الشفافية المستخدمة في صورة القناع من نوع mask-source (العنصر mask للعنصر svg) والخاصية غير وراثية وتقبل عدة أنواع من القيمتين كالتالي:

ملف html

```
<di> </div>
<svg>
    <mask id="mymask" style="mask-type: alpha">
       <polygon points="200 25,375 200,200 375,0 200"
                fill="green"/>
    </mask>
</svg>
```

ملف css

```
div{
    width: 180px;
    height: 220px;
    background: url(wood.jpg) center/cover;
    mask-image: url(#mymask);
    mask-position: center;
    mask-size: contain;
    }
```

– *luminance* : وهي القيمة الافتراضية وتعني أن نسبة إضاءة اللون في صورة القناع هي الأساس في تحديد القناع.

– *alpha* : نسبة الشفافية في صورة القناع هي الأساس في تحديد القناع.

original luminance alpha

mask-repeat

خاصية تحدد إمكانية أن تتكرر صورة القناع أو التدرج اللوني على المحورين الأفقي والرأسي وحتى تتكرر صورة القناع يجب أن تكون أبعادها أصغر من أبعاد خلفية العنصر التي ستقوم بقصها، والخاصية غير وراثية وتقبل قيمتين، الأولى تمثل نوع التكرار على المحور الأفقي، والثانية تمثل نوع التكرار على المحور الرأسي، والخاصية تقبل عدة أنواع من القيم كالتالي:

no-repeat | repeat-x | repeat-y |round |space

وقد شرحنا تأثير وسلوك هذه القيم في الخاصية background-repeat.

```
mask-position: right  bottom;
mask-size: 20%;
mask-repeat: space space;
```

ويمكن الاكتفاء بقيمة واحدة إذا كانت القيمتين متشابهتين.

repeat repeat-x repeat-y

والقيمة الافتراضية للخاصية هي repeat repeat.

والقيمة repeat-x هي اختصار القيمتين no-repeat repeat.

والقيمة repeat-y هي اختصار القيمتين repeat no-repeat.

mask-clip

خاصية تحدد المنطقة التي يغطيها القناع دون تغيير في أبعاده، والخاصية غير وراثية ولها قيمة افتراضية border-box كما أنها تقبل عدة أنواع من القيم كالتالي:

border-box | padding-box | content-box

```
mask-clip: content-box;
```

لاحظ أن صورة القناع احتفظت بكامل حجمها ولكن تم قصها حسب القيم السابقة.

mask-origin

خاصية تحدد المنطقة التي تحتوي داخلها القناع، والخاصية غير وراثية ولها قيمة افتراضية border-box كما أنها تقبل عدة أنواع من القيم كالتالي:

border-box | padding-box | content-box

```
mask-origin: content-box;
```

لاحظ أن صورة القناع قد تغيرت أبعادها، بحيث تصبح محتواة بالكامل داخل المنطقة المطلوبة من العنصر، حسب القيم السابقة.

mask-mode

خاصية تحدد نوع شفافية صورة القناع المستخدمة في إظهار خلفية العنصر، والخاصية غير وراثية وتقبل عدة أنواع من القيم كالتالي:

match-source | luminance | alpha

```
mask-mode: match-source;
```

background

mask-image

– *match-source* : القيمة الافتراضية وتعني استخدام صورة القناع للقيمة alpha إذا كانت من النوع image-source، أو استخدام القيمة luminance إذا كانت من النوع الثاني mask-source.

– *alpha* : قيمة تعني استخدام طبقة الشفافية لتشكيل صورة القناع.

– *luminance* : قيمة تعني استخدام إضاءة الألوان لتشكيل صورة للقناع.

alpha

luminance

mask-composite

خاصية تحدد العلاقة بين صورتين (طبقتين) للقناع أحدهما فوق الأخرى، والخاصية غير وراثية، ولها قيمة افتراضية add والخاصية تقبل عدة أنواع من القيم كالتالي:

ملف html

```
<di> </div>
```

ملف css

```
div{
    width: 600px;
    height: 220px;
    background: url(wood.jpg) center/cover;
    mask-image: url(circle.svg), url(circle.svg);
    mask-position: left, right;
    mask-size: contain;
    }
```

original layer 1 layer 2

```
   mask-composite: subtract;
```

- ***add*** : شكل القناع يتكون نتيجة إضافة (مجموع) طبقة القناع الثانية للطبقة الأولى.
- ***subtract*** : شكل القناع يتكون نتيجة طرح طبقة القناع الثانية من الطبقة الأولى.
- ***intersect*** : شكل القناع هو منطقة التقاطع بين الطبقتين.
- ***exclude*** -: شكل القناع يتكون نتيجة استبعاد منطقة التقاطع من مجموع الطبقتين.

add subtract intersect exclude

كما سبق أن ذكرنا أن الخاصية mask هي اختصار للخصائص الفرعية السابقة وقيمتها تجمع قيم كل الخصائص السابقة أو بعضها على أن تتضمن هذه القيم بالضرورة الخاصية mask-image وباقي الخصائص وجودها في القيمة اختياري وتكون قيمة الخاصية كالتالي:

ملف html

```
<di> </div>
```

ملف css

```
div{
    font: 800 100px arial;
    text-align: center;
    color: white;
    width: 650px;
    height: 400px;
    border: 20px solid green;
    padding: 10px;
    padding-top: 100px;
    box-sizing: border-box;
    background: url(wood.jpg) center/cover;
    mask: url(../../images/circle.svg) left/contain
          no-repeat padding-box exclude,
          url(../../images/circle.svg) right/contain
          no-repeat;
    }
```

original

mask

كما سبق أن أوضحنا قيمة الخاصية mask هي اختصار للخصائص الفرعية السابقة ولذلك فإننا نجد أن قيمتها قد احتوت على قيم الخصائص الفرعية كالتالي:

property	value
mask-image	url(../../images/circle.svg)
mask-position	left, right
mask-size	contain
mask-repeat	no-repeat
mask-origin/clip	padding-box
mask-composite	exclude

ويمكن حذف قيم أي من القيم السابقة أو إضافة أي قيم خصائص فرعية أخرى من القيم السابقة حسب التعديل المطلوب على القيم الافتراضية للخصائص الفرعية حيث إن قيم الخصائص التي لا يتم إضافتها في الخاصية mask يتم تعيينها عن طريق القيم الافتراضية لهذه الخصائص.

الخاصية mask وخصائصها الفرعية مدعومة في متصفح فايرفوكس أما المتصفحات المشتقة من متصفح chormium فيجب إضافة البادئة -webkit- قبل اسم هذه الخصائص.

mask-border

خاصية تقوم بعمل حدود للقناع، والخاصية هي اختصار لعدة خصائص فرعية واسم الخاصية غير مدعوم في أي متصفح، والاسم البديل هو mask-box-image وهو مدعوم في المتصفحات المشتقة من متصفح chormium (مثل جوجل كروم وإيدج)، لذلك يضاف قبل اسم الخاصية وقبل اسم الخصائص الفرعية البادئة -webkit- كالتالي:

ملف html

```
<di> </div>
```

ملف css

```
div{
    width: 400px;
    height: 400px;
    padding: 20px;
    margin: 20px;
    background: url(wood.jpg) center/50px;
    -webkit-mask: url(circle.svg) center/contain;
    }
```

original

mask

mask-box-image

mask-border-source

والاسم المدعوم المستخدم هو -webkit-mask-box-image-source وهي خاصية تحدد مصدر صور حدود القناع كالتالي:

```
-webkit-mask-box-image-source: url(circle.svg);
```

وصورة القناع قد تكون تدرج لوني كالتالي:

```
-webkit-mask-box-image-source: linear-gradient(to right,
```

```
                                    black, white);
```

image

linear-gradient

mask-border-repeat

والاسم المدعوم المستخدم هو -webkit-mask-box-image-repeat وهي خاصية تقوم بتحديد طريقة تكرار الصورة مصدر حدود القناع لتحديد طريقة ملء حدود القناع والخاصية غير وراثية وتقبل عدة أنواع من القيم كالتالي:

– ***stretch*** : القيمة الافتراضية وتعني استطالة الجزء المقسم من صورة حدود القناع لتملأ حدود القناع دون أي تكرار من الجزء المقسم من صورة حدود القناع.

– ***repeat*** : قيمة تعني أن صورة حدود القناع تتكرر بنفس أبعادها بدون تغيير فيها مما يترتب عليه وجود وحدة مكررة غير مكتملة من صورة حدود القناع.

– ***round*** : قيمة تعني أن صورة حدود القناع تتكرر مع امتداد الصورة لتملأ الفراغ الناتج عن الفرق بين أبعاد الحدود ومجموع أبعاد الواحدات المتكررة من صورة حدود القناع.

– ***space*** : قيمة تعني أن صورة حدود القناع تتكرر مع ترك مسافة فارغة بدلاً من وحدة التكرار الغير مكتملة.

```
-webkit-mask-box-image-repeat: round;
```

repeat

round

space

mask-border-slice

والاسم المدعوم المستخدم هو -webkit-mask-box-image-slice وهي خاصية تحدد الجزء حدود القناع الذي سيتم ملؤه بصورة حدود القناع وتقبل نوعين من القيم كالتالي:

– ***number*** : قيمة تعني أن الجزء المقطوع من صورة القناع والذي سيتم ملؤه بصورة حدود القناع يساوي قيمة مطلقة من عرض وارتفاع عنصر القناع (وليس صورة القناع) حسب اتجاه الجزء المقطوع.

– ***percentage*** : قيمة تعني أن الجزء المقسم من صورة القناع والذي سيتم ملؤه بصورة حدود القناع عبارة عن نسبة مئوية من عرض وارتفاع صورة القناع (وليس عنصر القناع) حسب اتجاه الجزء المقطوع.

```
-webkit-mask-box-image-slice: 20 15 13 17;
```

لاحظ أن قيمة الخاصية مكونة من أربع قيم، حيث تمثل القيمة الأولى عرض الجزء العلوي والثانية عرض الجزء الأيمن والثالثة عرض الجزء السفلي، والرابعة عرض الجزء الأيسر.

وقد تقبل الخاصية ثلاث قيم إذا كان عرض الجزء الأيمن يساوي عرض الجزء الأيسر، فنستغني عن القيمة الرابعة ونكتفي بالثلاث قيم الأولى.

وقد تقبل الخاصية قيمتين إذا كان عرض الجزء العلوي يساوي عرض الجزء السفلي والجزء الأيمن يساوي الجزء الأيسر.

ويمكن الاكتفاء بقيمة واحدة، إذا تساوى عرض كل الأجزاء من كل الاتجاهات.

mask-border-width

والاسم المدعوم المستخدم هو -webkit-mask-box-image-width وهي خاصية تحدد عرض حدود القناع والخاصية غير وراثية وتقبل عدة أنواع من القيم كالتالي:

– ***auto*** : قيمة تعني أن عرض الحد يتحدد تلقائياً على أساس عرض وارتفاع تقسيمات الحدود في الخاصية mask-border-slice.

– ***value*** : قيمة تعني أن عرض الحدود يكون قيمة مطلقة.

– ***percentage*** : قيمة تعني أن عرض الحدود يكون نسبة مئوية من عرض القناع للحدين العلوي والسفلي، ونسبة من ارتفاع القناع للحدين الأيمن والأيسر.

```
-webkit-mask-box-image-width: 20px 15px 10px 25px;
```

10px 20px 30px

ومثل القيمة السابقة، قيمة هذه الخاصية يمكن أن تتكون من قيمة واحدة إلى أربع قيم، حسب الشروط والترتيب السابق دراستهم.

mask-border-outset

والاسم المدعوم المستخدم هو -webkit-mask-box-image-outset وهي خاصية تحدد المسافة بين حدود العنصر وحدود القناع وهي غير وراثية وتقبل نوعين من القيم كالتالي:

– ***length*** : قيمة تعني أن المسافة تساوي قيمة مطلقة.

– ***number*** : قيمة تعني أن المسافة تساوي أجزاء أو مضاعفات عرض وارتفاع حدود القناع، فالقيمة 2 تعني أن المسافة ضعف عرض الحدود أو ارتفاعها حسب اتجاه حد القناع.

```
-webkit-mask-box-image-outset: 25px 10px 15px 30px;
```

0px 10px 20px

ومثل القيمتين السابقتين قيمة هذه الخاصية يمكن أن تتكون من قيمة واحدة إلى أربع قيم، حسب الشروط والترتيب السابق دراستهم.

mask-border-mode

وهي خاصية تحدد إذا كانت نسبة الشفافية في صورة حدود القناع أو نسبة إضاءة الألوان هي المسئولة عن شكل حدود القناع والخاصية غير وراثية وغير مدعومة في أي متصفح وتقبل نوعين من القيم كالتالي: luminance | alpha

وقد وضحنا سلوك وتأثير كل قيمة في الخاصية mask-mode.

```
-webkit-mask-box-image-mode: luminance;
```

وكما سبق أن أوضحنا أن الخاصية –webkit-mask-box-image- تعتبر اختصاراً للخصائص الفرعية السابقة كالتالي:

```
-webkit-mask-box-image: url(bord.png) 30/20px/5px round;
```

والخصائص الفرعية وقيمها بالترتيب الآتي:

property	value
mask-box-image-source	url(bord.png)
mask-box-image-slice	30
mask-box-image-width	20px
mask-box-image-outset	5px
mask-box-image-repeat	round

transition

خاصية تقوم بتسجيل خطوات انتقال العنصر من وضع إلى وضع آخر سواء نتيجة تغير في قيم خصائصه مثل الحركة أو الألوان أو الشفافية أو الحدود أو الهامش وغيرها، والخاصية هي اختصار لعدد من الخصائص الفرعية التي تتحكم في كيفية الانتقال transition.

ملف html

```
<div>Transition</div>
```

ملف css

```
div{
    font: 800 10px arial;
    text-align: center;
    width: 75px;
    height: 75px;
    background: orange;
    border: 1px solid;
    margin: 40px;
    padding-top: 25px;
    box-sizing: border-box;
    }
div:hover{
    background: blue;
    color: white;
    transform: rotate(45deg) scale(2);
    border: 5px solid red;
    border-radius: 50%;
    }
```

original div

div:hover

الكود السابق سينتج عنصر div له خصائص معينة وعند المرور عليه بالمؤشر أو ما يسمى hover ستتغير خصائص العنصر ويتحول إلى الشكل الثاني ولكن هذا التحول يكون مفاجي وهنا يأتي دور خاصية transition أو تتحكم في هذا التغير والتحول بطريقة ما تجعله انسيابياً أو مفاجئاً، سريعاً أو بطيئاً، أو التحكم في خصائص دون أخرى.

والخصائص الفرعية كالتالي:

transition-duration

خاصية تحدد الفترة الزمنية التي تتم خلالها عملية الانتقال، وتكون قيمتها عبارة رقم يتبعها رمز الفترة الزمنية s (second ثانية) أو ms (millisecond ملي ثانية) وإذا كانت الخصائص أكثر من خاصية فيكون لكل خاصية الفترة الزمنية الخاصة بها بالترتيب ويفصل بينها بفاصلة ويمكن الاكتفاء بقيمة واحدة لكل الخصائص إذا تساوت القيمة لها جميعاً، والخاصية غير وراثية ولها قيمة افتراضية 0s.

أضف الكود التالي للعنصر div ولاحظ التغير في فترة الانتقال قبل وبعد الكود:

```
transition-duration: 3s;  /* equals 3000ms */
```

transition-property

خاصية تحدد الخصائص التي سيتم التحكم في انتقالها وهي خاصية غير وراثية وتقبل عدة أنواع من القيم كالتالي:

- *property name*: قيمة تعني تحديد اسم الخاصية التي سيتم التحكم في عملية انتقالها وقد تكون أكثر من خاصية يفصل بينها بفاصلة.
- *all* : القيمة الافتراضية وتعني التحكم في انتقال كل الخصائص التي تغيرت.
- *none* : قيمة تعني إلغاء التحكم في انتقال أي خاصية.

أضف الكود التالي للعنصر div ولاحظ أن كل الخصائص تنتقل في الفترة الزمنية المحددة لها بينما لون الخلفية واستدارة الحدود تنتقل فجائياً وذلك لتجاهلها من قبل الخاصية transition-property لها كالتالي:

```
transition-duration: 4s, 8s, 1s;
transition-property: transform, border, color;
```

transition-delay

خاصية تحدد الفترة الزمنية التي تسبق عملية بدء الانتقال أو هي فترة التأخير في بدء عملية الانتقال، وتقبل قيمة عبارة عن رقم يتبعه رمز الفترة الزمنية وقد تحتوي القيمة على أكثر من فترة تأخير، يفصل بينها فاصلة إذا احتوت الخاصية transition-property على أكثر من خاصية ويمكن الاكتفاء بفترة تأخير واحدة لكل الخصائص والخاصية غير وراثية ولها قيمة افتراضية 0s.

أضف الكود التالي للعنصر div ولاحظ التأخر في بدء انتقال كل خاصية حسب الفترة المحددة لها كالتالي:

```
transition-duration: 2s, 4s, 1s;
transition-property: transform, border, color;
transition-delay: 3s, 4s, 2s;
```

transition-timing-function

خاصية تزامن الانتقال وهي تحدد مراحل التباطؤ والتسارع أثناء عمليه الانتقال وهذه الخاصية غير وراثية وتقبل عدة أنواع من القيم كالتالي:

- ***ease*** : القيمة الافتراضية وتعني أن السرعة تزداد من البداية إلى منتصف زمن الانتقال، ثم تتباطأ من منتصف الفترة إلى نهايتها.
- ***linear*** : قيمة تعني أن السرعة متساوية في كل مراحل فترة الانتقال.
- ***ease-in*** : قيمة تعني أن سرعة الانتقال تبدأ بطيئة وتتزايد حتى نهاية فترة الانتقال.
- ***ease-out*** : قيمة تعني أن سرعة الانتقال تبدأ سريعة وتتباطأ حتى نهاية الانتقال.
- ***ease-in-out*** : قيمة تعني أن سرعة الانتقال تبدأ بطئية ثم تتسارع ثم تتباطأ في نهاية فترة الانتقال.

استخدام الكود التالي لتوضيح الفرق بين القيم الخمس السابقة:

ملف html

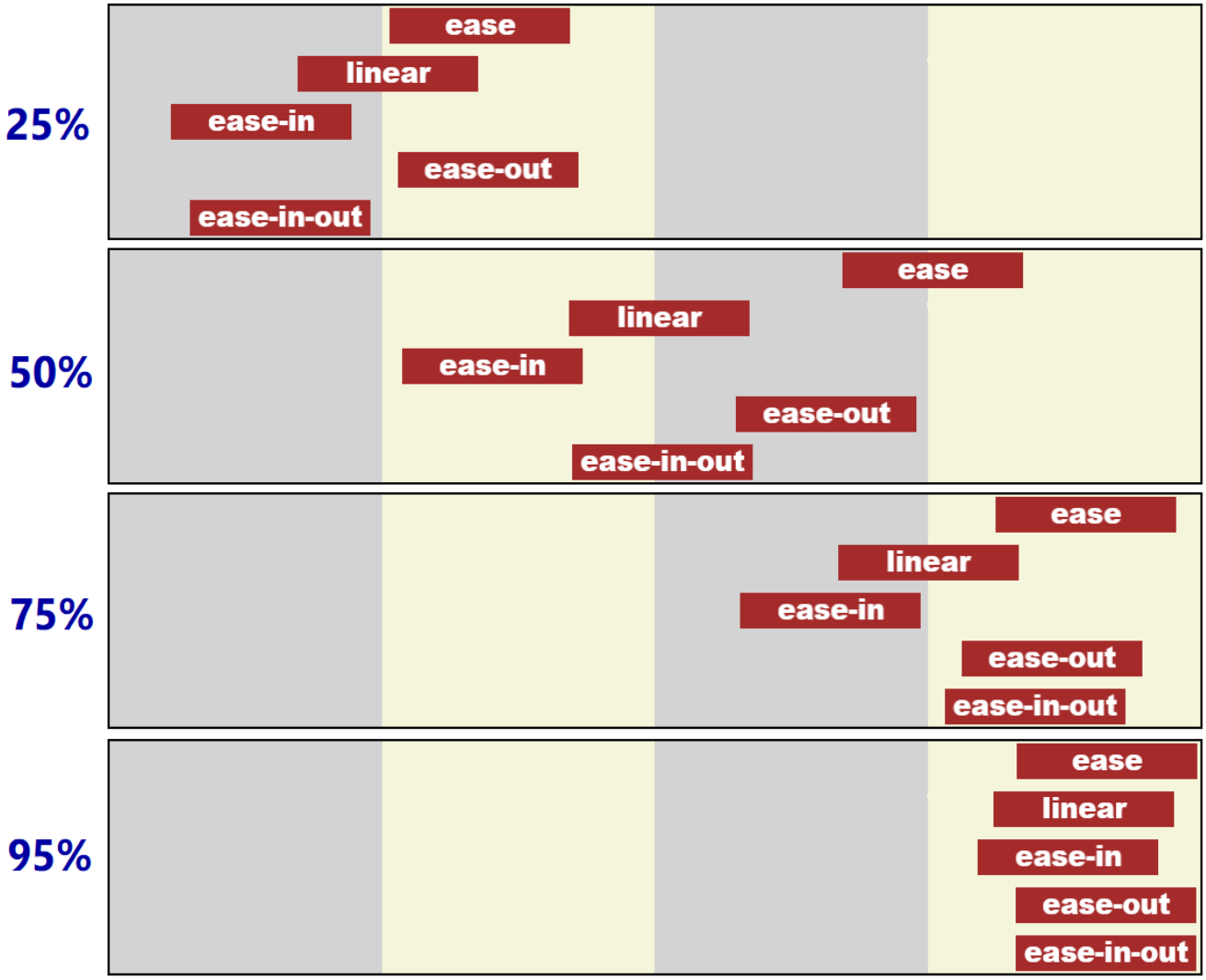
```
<div class="parent">
     <div class="ease">ease</div>
     <div class="linear">linear</div>
     <div class="ease-in">ease-in</div>
     <div class="ease-out">ease-out</div>
     <div class="ease-in-out">ease-in-out</div>
</div>
```

ملف css

```
.parent{
       width: 480px;
       height: 88px;
       padding: 1px;
       border: 1px solid;
       background: repeating-linear-gradient(to right,
                   lightgray 0% 25%, beige 25% 50%);
        }
.ease, .linear, .ease-in, .ease-out, .ease-in-out{
      font: 800 12px arial;
      color: white;
      text-align: center;
      width: 80px;
      height: 16px;
      background:brown;
      margin-bottom: 2px;
      }
.parent:hover > div{
      transition-duration: 12s;
      transform: translate(400px);
      }
.ease{
      transition-timing-function: ease;
      }
.linear{
      transition-timing-function: linear;
```

```
        }
.ease-in{
      transition-timing-function: ease-in;
      }
.ease-out{
      transition-timing-function: ease-out;
      }
.ease-in-out{
      transition-timing-function: ease-in-out;
     }
```

لاحظ الشكل التالي الناتج عن تطبيق الكود، والاختلاف بين مراحل انتقال العنصر عند الفترات 25% و 50% و 75% و 90% من فترة الانتقال كالتالي:



– ***steps()*** : دالة تقوم بتقسيم عملية الانتقال على عدد معين من الخطوات، فإذا كانت فترة الانتقال هي 5s وكان عدد خطوات الانتقال هو 5 خطوات فهذا يعني أن فترة الانتقال مقسمة إلى 5 مراحل، وكل مرحلة عبارة عن خطوة للانتقال وهي: 20% و 40% و 60% و 80% و 100%.

وسوف نستعمل مصطلح القفزة، والقفزة لا تدخل ضمن الفترة الزمنية لعملية الانتقال، فإذا كانت مدة الانتقال هي 5s وكان عدد خطوات الانتقال هو 5 خطوات فهذا يعني أن كل خطوة تستغرق 1s إلا إذا بدأ العنصر عملية الانتقال بقفزة فتكون الخطوة الأول التي تمثل القفزة خارج الفترة الزمنية، وتقسم الفترة الزمنية (5s) على الخطوات الأربع المتبقية، وإذا أنهى العنصر عملية الانتقال بقفزة، فهذا يعني أن الخطوة الأخيرة خارج الفترة الزمنية وأن الفترة الزمنية سوف تقسم على الخطوات الأربع الأولى.

وهذه الدالة تقبل معاملين، الأول هو عدد الخطوات، والثاني نوع الخطوات:

– jump-start : تقسم الخطوات بحيث تبدأ أول قفزة مع بداية عملية الانتقال وهي تساوي قيمة أخرى هي start.

– jump-end : تقسم الخطوات بحيث تنتهي آخر قفزة مع نهاية عملية الانتقال وهي تساوي قيمة أخرى هي end.

– jump-none : لا توجد قفزة في البداية ولا في النهاية.

– jump-both: تقسم الخطوات بحيث تبدأ أول قفزة مع بداية عملية الانتقال وآخر قفزة مع نهاية عملية الانتقال.

– step-start : تختصر كل الخطوات في خطوة واحدة تبدأ وتنتهي مع بداية عملية الانتقال، وهذه القيمة اختصار للدالة steps(1, jump-start).

– step-end : تختصر كل الخطوات في خطوة واحدة تبدأ وتنتهي مع نهاية عملية الانتقال، وهذه القيمة اختصار للدالة steps(1,jump-end).

استخدم الكود التالي لتوضيح الفرق بين القيم السابقة كالتالي:

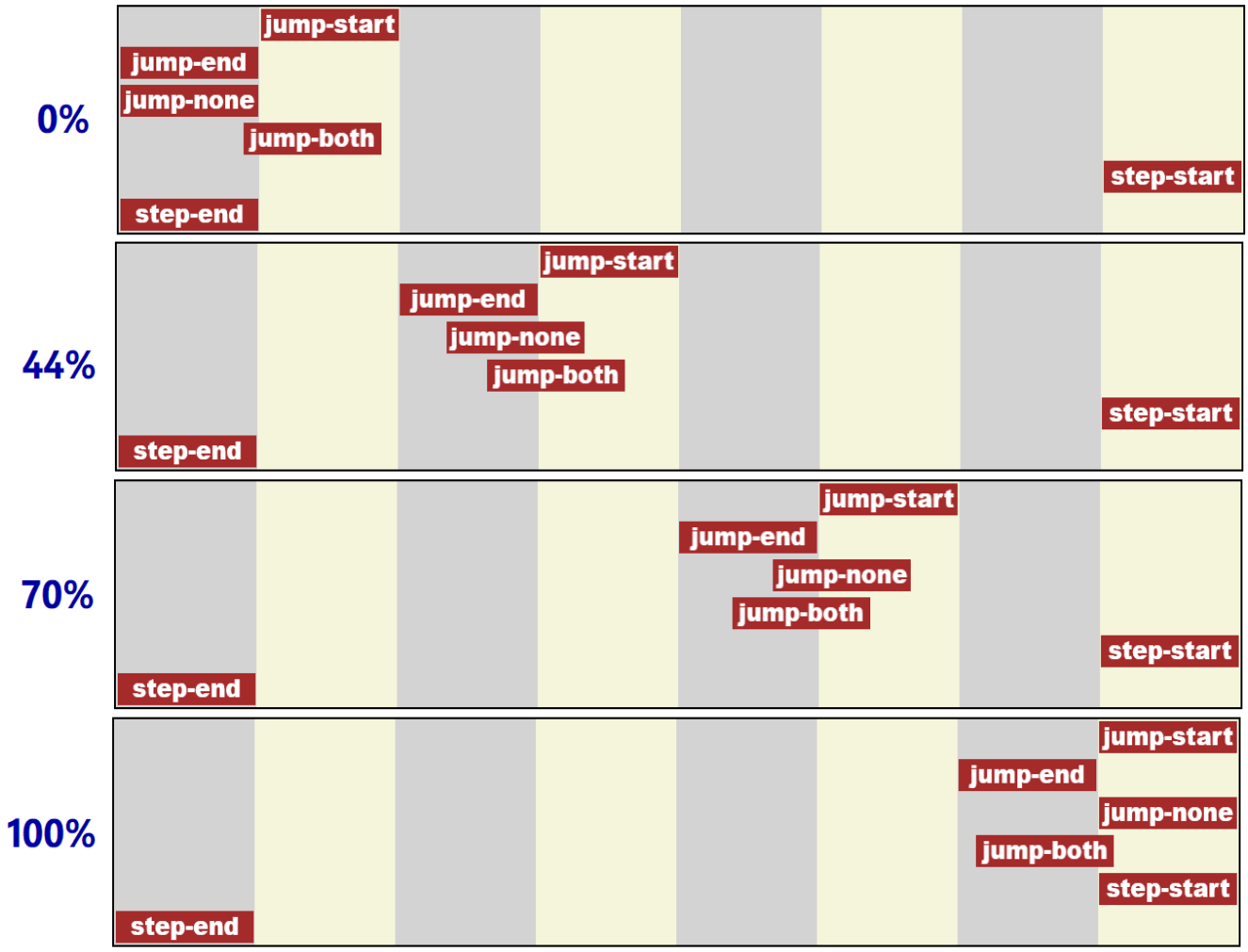
ملف html

```
<div class="parent">
   <div class="jstart">jump-start</div>
   <div class="jend">jump-end</div>
   <div class="jnone">jump-none</div>
   <div class="jboth">jump-both</div>
   <div class="sstart">step-start</div>
   <div class="send">step-end</div>
</div>
```

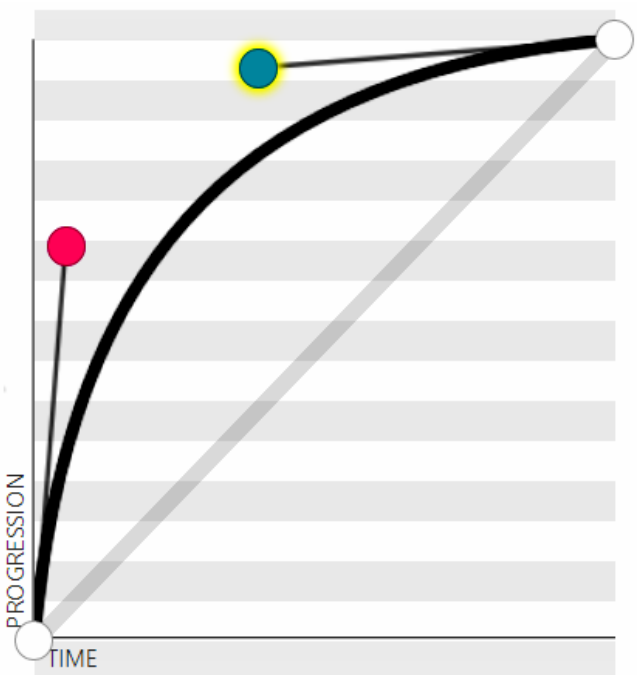
ملف css

```
.parent{
       width: 610px;
       height: 111px;
       padding: 1px;
       border: 1px solid;
       background: repeating-linear-gradient(to right,
                   lightgray 0% 12.5%, beige 12.5% 25%);
        }
.jstart, .jend, .jnone, .jboth,  .sstart, .send{
       font: 800 12.5px arial;
       color: white;
       text-align: center;
       width: 75px;
       height: 16px;
       background:brown;
       margin-bottom: 3px;
  }
.parent:hover > div{
       transition-duration: 7s;
       transform: translate(535px);
       }
.jstart{
      transition-timing-function: steps(7, jump-start);
       }
.jend{
     transition-timing-function: steps(7,jump-end);
```

```
    }
.jnone{
      transition-timing-function: steps(7,jump-none);
     }
.jboth{
      transition-timing-function: steps(7,jump-both);
     }
.sstart{
      transition-timing-function: step-start;
      }
.send{
    transition-timing-function: step-end;
    }
```



لاحظ أن القيم التي تبدأ بقفزة قد بدأت قبل بداية فترة الانتقال (0%)، وأن القيم التي تنتهي بقفزة لم تنته إلا بعد انتهاء فترة الانتقال (100%).



– ***cubic-bezier()*** : دالة تقوم بتحديد تسارع وتباطؤ عملية الانتقال بناء على منحنى السرعة الذي له نقطة بداية ونقطة نهاية وكل نقطة لها مقبض يحدد مقدار إنحناء منحنى السرعة حيث إن كل مقبض له إحداثيين الأول أفقي على المحور x-axis وهو يمثل نسبة الزمن اللازم لتنفيذ عملية الانتقال والآخر رأسي على المحور y-axis وهو يمثل نسبة التقدم في عملية الانتقال.

وعلى ذلك فالدالة تقبل 4 معاملات، اثنان يمثلان نقطة بداية عملية الانتقال وقيمتهما 0، واثنان يمثلان نقطة نهاية عملية الانتقال وقيمتهما هي 1 وهم كالتالي:

الأول: x-time : وهو بداية زمن عملية الانتقال وقيمتة المبدأية هي 0 وأي زيادة في قيمة هذا المعامل تعني زيادة في زمن بداية الانتقال مما يعني تباطؤ عملية الانتقال في بدايتها.

الثاني: x-transition : وهو مقدار التقدم في بداية عملية الانتقال وقيمته المبدأية هي 0 وأي زيادة في قيمة هذا المعامل تعني تسارع عملية الانتقال بدايتها.

الثالث: y-time : وهو زمن نهاية عملية الانتقال وقيمته المبدأية هي 1 وأي تناقص في قيمة هذا المعامل تعني تناقص زمن الانتقال مما يعني تسارع عملية الانتقال في نهايتها.

الرابع: y-transition : وهو مقدار التقدم في نها ية عملية الانتقال وقيمته المبدأية هي 1 وأي تناقص في قيمة هذا المعامل تعني تباطؤ عملية الانتقال في نهايتها.

ملاحظة: أسماء هذه المعاملات غير متفق عليها وإنما اخترتها لتوضيح ما تمثله فقط.

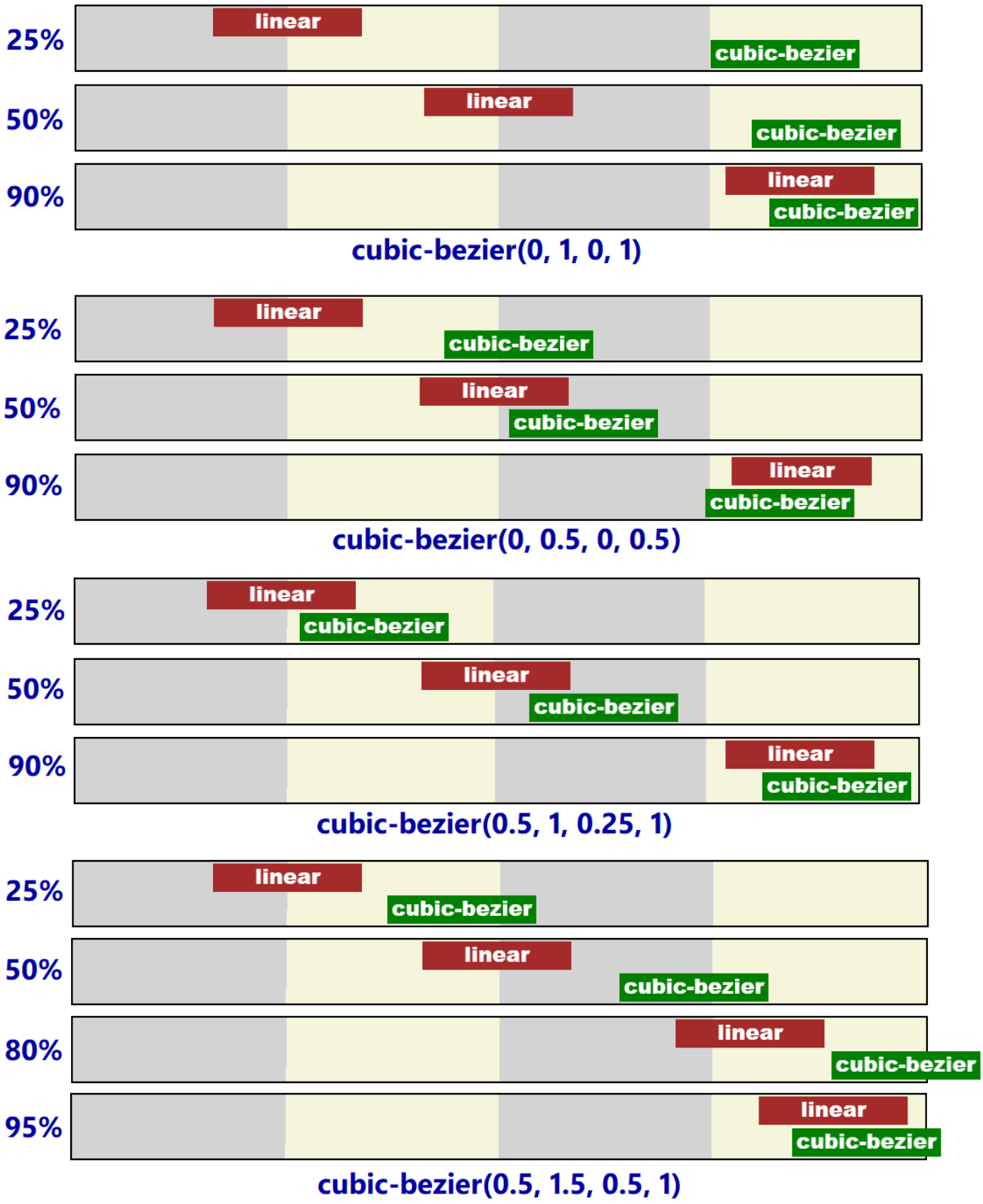
استخدم الكود التالي وقم بتجربة قيم الدالة cubic-bezier كما بالشكل ولاحظ تأثير تغيير القيم على تسارع العنصر في بداية ونهاية عملية الانتقال كالتالي:

ملف html

```
<div class="parent">
     <div class="linear">linear</div>
     <div class="cubicbz">cubic-bezier</div>
</div>
```

ملف css

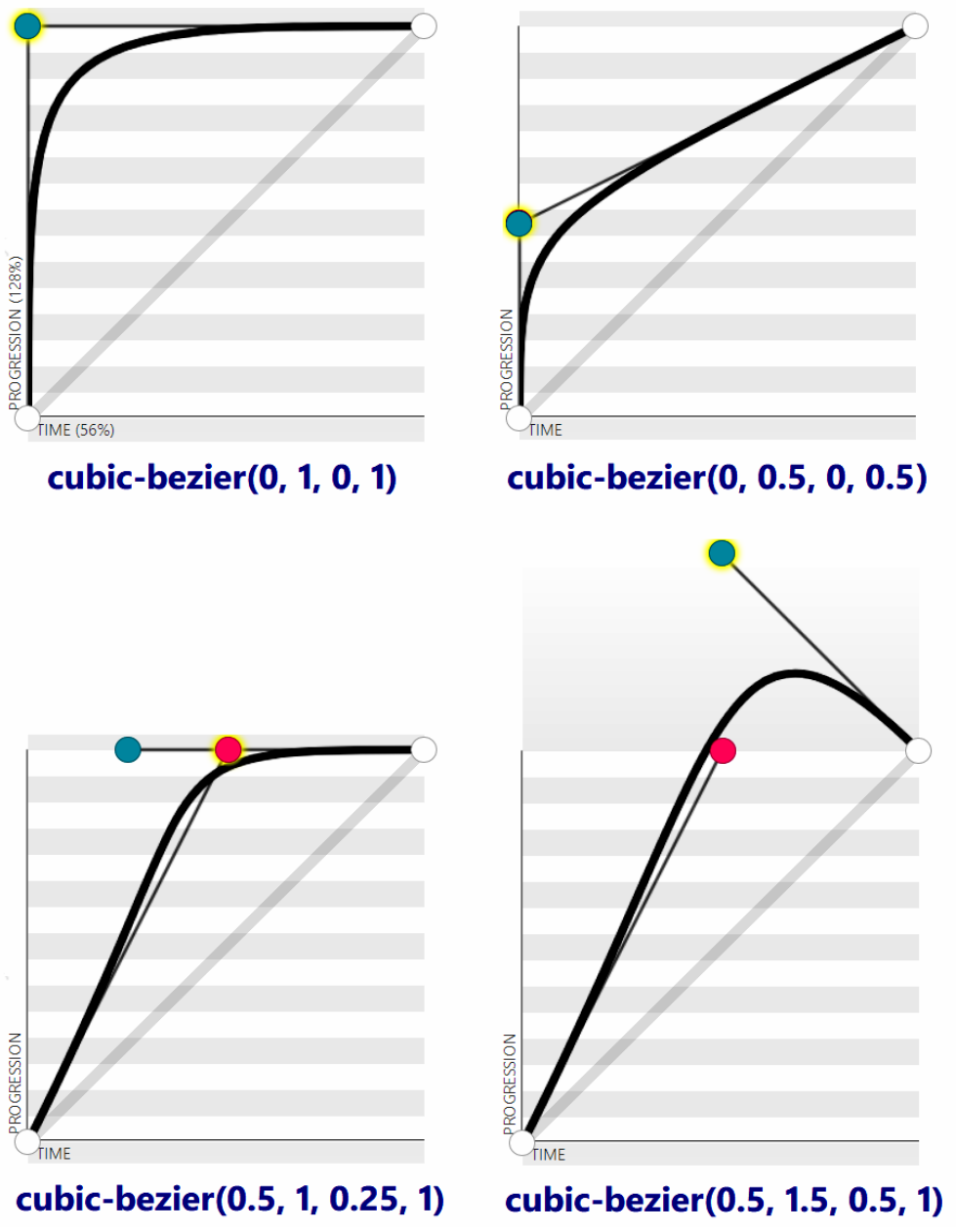
```
.parent{
       width: 485px;
       height: 34px;
       padding: 1px;
       border: 1px solid;
       background: repeating-linear-gradient(to right,
                   lightgray 0% 25%, beige 25% 50%);
        }
.linear, .cubicbz{
      font: 800 12px arial;
      color: white;
      text-align: center;
      width: 85px;
      height: 16px;
      margin-bottom: 2px;
      }
.parent:hover > div{
      transition-duration: 12s;
      transform: translate(400px);
      }
.linear{
      transition-timing-function: ease;
      background: brown;
      }
.cubicbz{
      transition-timing-function: cubic-bezier(0,1,0,1);
      background: green;
        }
```



لاحظ أنه تم إضافة عنصر له قيمة linear للخاصية transition-timing-function لمقارنة تسارعه وتباطؤه مع تسارع وتباطؤ العنصر ذي القيمة cubic-

bezier لنفس الخاصية مع اختلاف القيم في كل مرة.

الشكل التالي يوضح منحنى التسارع والتباطؤ للقيم بالشكل السابق للعنصر ذي قيمة cubic-bezier للخاصية transition-timing-function:



شاهد أكواد خاصية transition-timing-function على الروابط التالية:

https://codepen.io/wailmonir/pen/dyOEmpx
https://codepen.io/wailmonir/pen/wvobmgJ
https://codepen.io/wailmonir/pen/KKNLomZ

وكما سبق أن ذكرنا فالخاصية transition هي اختصار الخصائص الأربع السابقة، لذلك فقيمتها يمكن أن تجمع قيم الأربع خصائص أو تكتفي بقيمة واحدة على الأقل لأي خاصية منهم، وتكون قيم الخصائص الباقية هي القيم الافتراضية ولكن من المنطقي ضرورة تحديد فترة زمنية للانتقال transition-duration حتى لا يكون الانتقال فجائياً:

ملف html

```
<div>Transition</div>
```

ملف css

```
div{
    font: 800 10px arial;
    text-align: center;
    width: 75px;
    height: 75px;
    background: olive;
    border: 1px solid;
    margin: 40px;
    padding-top: 40px;
    border-radius: 50%;
    box-sizing: border-box;
    }
div:hover{
    background: darkviolet;
    color: white;
    transform: skew(45deg);
    border: 5px solid red;
    }
```

لاحظ الشكل الأصلي على اليسار، وعلى اليمين الشكل بعد إتمام عملية الانتقال.

والقيم الافتراضية للخاصية transition كالتالي:

```
transition: all 0s 0s ease;
```

أضف الكود التالي لتغيير قيم خصائص الانتقال وباقي القيم تبقى افتراضية كالتالي:

```
transition: background, color;
```

لاحظ أنه بدون تحديد زمن للانتقال لن تلاحظ أي تغيير لأن الانتقال سيكون مفاجئاً، لذلك أضف الكود التالي للعنصر div قبل أن تحدد خصائص الانتقال كالتالي:

```
transition: 3s;
```

قم الآن بتحديد قيم خصائص الانتقال وافصل بينها بفاصلة ويجب أن يكون لكل خاصية زمن الانتقال الخاص بها وإلا اتخذت الزمن الافتراضي 0s كالتالي:

```
transition: 3s background, 4s color;
```

قم بتحديد زمن تأخير الانتقال لكل خاصية على أن يكون بعد زمن الانتقال، لأن الزمن الأول سيكون دائما زمن الانتقال وفي حالة عدم الرغبة في تحديد زمن للانتقال يتم تحديده يدوياً 0s كالتالي:

```
transition: 3s 2s background, 0s 4s color;
```

لاحظ الآن أن لون background يتأخر 2s ثم يستغرق 3s في عملية الانتقال، وأن اللون color يتأخر 4s ويكون الانتقال مفاجئاً لأن زمن الانتقال هو 0s وباقي خصائص العنصر يكون انتقالها مفاجئاً لأن لم يتم تحديدها في الخاصية transition.

أضف الكود التالي لتغيير قيمة خاصية تزامن الانتقال transition-timing-function كالتالي:

```
transition: 3s 2s background ease-in, 2s 4s color linear;
```

لاحظ اختلاف تسارع وتباطؤ عملية انتقال لون النص عن لون الخلفية، كما أن باقي خصائص العنصر لن تلاحظ عليها أي تغيير لأن زمن انتقالها 0s (الانتقال مفاجئ).

لا يشترط ترتيب قيم الخصائص داخل قيمة الخاصية transition إلا في حالة زمن

الانتقال وزمن التأخير، إذ يجب أن نحدد زمن الانتقال قبل زمن التأخير كما ذكرنا من قبل.

backface-visibility

خاصية تحدد إمكانية ظهور الجانب الخلفي للعنصر من عدمه عند عمل دوران له بزاوية مقدارها 180deg والخاصية غير وراثية وتقبل نوعين من القيم كالتالي:

- ***visible*** : وهي القيمة الافتراضية وتسمح بظهور الوجه الخلفي للعنصر ويكون ظهوره معكوساً للجانب الأمامي للعنصر.
- ***hidden*** : قيمة تعني إخفاء الوجه الخلفي للعنصر.

ملف html

```
<div class="parent">
<div class="back">Backface</div>
<div class="front">Frontface</div>
</div>
```

ملف css

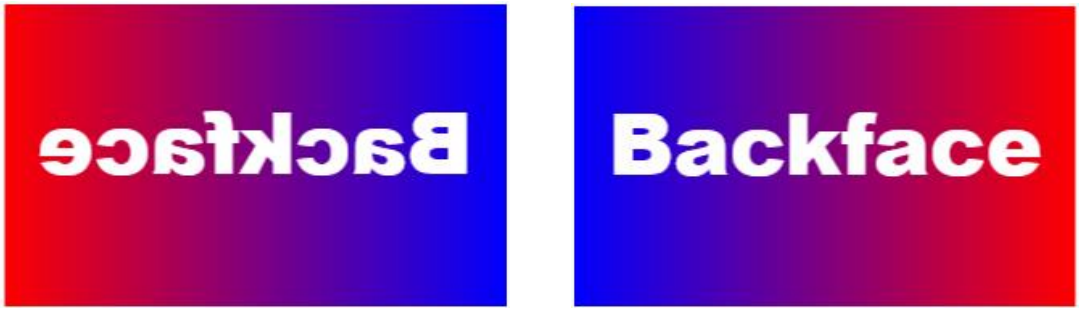
```
.parent{
        width: 250px;
        height: 150px;
        margin: 80px;
        perspective: 500px;
        transform-style: preserve-3d;
        transition: all 3s;
        }
.front, .back{
     position: absolute;
     font: 800 42px arial;
     text-align: center;
     width: 250px;
     height: 150px;
     padding-top: 45px;
     box-sizing: border-box;
     }
.front {
     background: linear-gradient(to right,green,orange);
     backface-visibility: hidden;
```

```
    }
.back{
    background: linear-gradient(to right,blue,red);
    color: white;
    transform: rotateY(180deg);
    }
.parent:hover{
    transform: rotateY(180deg);
    }
```

الكود السابق، أنشأ عنصر div أب بدون تنسيقات تسمح بظهوره، وبداخله عنصري div أبناء الأول front والثاني back، ولكل منهما تنسيقاته كما بالشكل، ونتيجة لاستخدام القيمة absolute للخاصية position للعنصرين، أصبح العنصرين فوق بعضهما، المطلوب عند دوران العنصر الأب بزاوية 180deg لا يتم عرض الوجه الخلفي للعنصر الأعلى (front) ولكن المطلوب أن يختفي الوجه الخلفي للعنصر front وتظهر واجهة العنصر الخلفي (back) الذي قمنا بعمل دوران 180deg له حتى يعود لوضعه الطبيعي عند دوران العنصر الأب 180deg.

الكود سينتج الشكل على اليسار وعند دوران العنصر سينتج الشكل على اليمين.

للتوضيح قم بتعليق العنصر front مؤقتاً ليظهر العنصر back بالشكل التالي:



وسيكون هذا هو الترتيب الفعلي للعنصرين مع إزاحة العنصر front يميناً وإلى أسفل لتخيل وضع العنصرين بالنسبة لبعضهما كما بالشكل التالي:

المطلوب الآن عند دوران العنصر الأب بزاوية deg180، فسيتم عمل دوران لعنصري الأبناء، وحتى لا يظهر الجانب الخلفي للعنصر front يجب إسناد القمة hidden للعنصر front، حتى لا يحجب العنصر الخلفي back.

لاحظ ضرورة استخدام الخاصية transform-style وإسناد القيمة preserve-3d لعرض العنصرين في الوضع ثلاثي الأبعاد حتى يظهر العنصر الخلفي عند دوران العنصرين، لأن عدم استخدام هذه الخاصية وهذه القيمة سيعرض الجانب الخلفي للعنصر الأمامي front.

ويمكن تطبيق الكود ومشاهدة النتيجة النهائية للعمل على هذا الرابط.

https://codepen.io/wailmonir/pen/xxRBYKz

animation

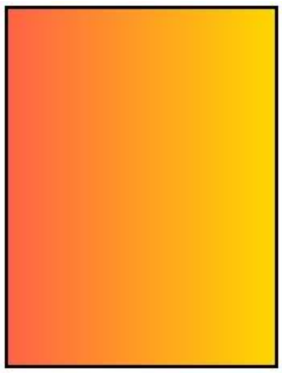
يطلق عليها animation أو التحريك برغم من أنها تقوم بتغيير قيم خصائص العنصر بصورة متتابعة وليس الحركة فقط وإنما خصائص المظهر والحجم والميل والدوران وغيرها وهذه الخاصية هي اختصار لعدد من الخصائص الفرعية التي تتحكم في هذا التغيير:

ملف html

```
<div></div>
```

ملف css

```
div{
    width: 75px;
    height: 100px;
    background: linear-gradient(to right, tomato, gold);
    margin: 40px;
    border: 1px solid;
   }
```



animation-name

خاصية تحدد اسم مخصص للحركة animation ويتم هذا التحريك حسب قاعدة تقوم بإجراء تعديلات مخصصة متتابعة على خصائص العنصر وتسمى هذه القاعدة @keyframes وتتكون من عدد من مراحل التغيير في خصائص العنصر، على أن تتضمن كل مرحلة الخصائص المطلوب تغييرها وقيم كل خاصية في هذه المرحلة، وبعد الانتهاء من استكمال هذه القاعدة يسند اسمها كقيمة للخاصية animation-name، ويتم بناء هذه القاعدة كالتالي:

1– كتابة الكلمة المحفوظة keyframes وقبلها الرمز @.

2– اختيار اسم للقاعدة، وبعده يتم فتح قوسين {} ليتم كتابة مراحل التحريك داخلهما كما بالشكل التالي:

```
@keyframes animation-name { ..... }
```

3– مراحل التحريك هي الفترات الزمنية التي ستتغير عندها قيم خصائص العنصر وتبدأ كل مرحلة بتحديد فترة التحريك ثم يتم فتح قوسين {} لتحتوي على الخصائص المطلوب تغييرها وقيمها في هذه المرحلة ويتم تحديد مراحل التحريك بطريقتين:

– تقسيم مراحل التحريك إلى مرحلتين فقط، فتكون الأولى هي مرحلة البداية وتسمي from وتحتوي خصائص العنصر وقيمها في بداية التحريك وتكون الثانية مرحلة النهاية وتسمي to وتحتوي نفس الخصائص والقيم المطلوب الوصول إليها في نهاية التغيير كما بالشكل التالي:

```
from{width: 100px;}
to   {width: 300px;}
```

ويمكن الاستغناء عن مرحلة البدء from والاكتفاء بمرحلة النهاية to واعتبار أن قيم خصائص العنصر داخل العنصر هي نفسها قيم مرحلة البدء from.

– تقسيم مراحل التحريك إلى مرحلتين أو أكثر، واسم كل مرحلة يكون بداية الفترة الزمنية لكل مرحلة على هيئة نسبة مئوية من فترة التغيير وكل مرحلة تحتوي خصائص العنصر المطلوب تغييرها وقيم كل مرحلة كما بالشكل التالي:

```
0% {width: 100px;}
50% {width: 300px;}
100% {width: 500px;}
```

ويمكن الاستغناء عن مرحلة البدء 0% والاكتفاء بباقي المراحل واعتبار أن قيم خصائص العنصر داخل العنصر هي نفسها قيم مرحلة البدء 0%.

وعلى ذلك يكون الشكل النهائي لقاعدة @keyframes كالتالي:

```
@keyframes animation-name1{
        from{
             width: 100px;
             height: 100px;
            }
        to{
           width: 200px;
           height: 75px;
           }
        }
```

أو تكون على الشكل التالي:

```
@keyframes animation-name2{
        0%{
           color: salmon;
           background: blue;
          }
        50%{
           color: white;
           background: brown;
           }
       100%{
           color: black;
           background: green;
           }
        }
```

ويكون اسم القاعدة هو قيمة الخاصية animation-name، ويمكن إنشاء عدة قواعد ولكل منها مراحلها المختلفة وخصائصها المختلفة وقيمها المختلفة.

ويمكن أن يسند للعنصر أكثر من تحريك animation وكل تحريك له قاعدته الخاصة والتي يصبح اسمها هو قيمة الخاصية animation-name، ويتم الفصل بين اسم التحريك والتحريك الذي يليه بفاصله كالتالي:

أضف الكودين الخاصين بكلا قاعدتي التحريك @keyframes إلى ملف التنسيقات css الذي يحتوي العنصر div، ثم أضف الكود التالي للعنصر div:

```
animation-name: animation-name1, animation-name2;
```

لاحظ عدم حدوث أي تغيير في خصائص العنصر لعدم تحديد فترة زمنية لعملية التحريك حيث إن قيمتها الافتراضية هي 0s وبالتالي لن تتم عملية التحريك.

animation-duration

خاصية تحدد الفترة الزمنية لعملية التحريك وتكون عبارة عن رقم يتبعه رمز الفترة الزمنية إما s (second ثانية) أو ms (millisecond ملي ثانية) وفي حالة تطبيق أكثر من تحريك للعنصر يكون لكل تحريك فترته الزمنية ويتم الفصل بين كل فترة زمنية والتي تليها بفاصلة، ويمكن الاكتفاء بفترة واحدة لكل الخصائص إذا تساوت الفترات في القيمة والخاصية غير وراثية ولها قيمة افتراضية 0s.

أضف الكود التالي للكود السابق للعنصر div:

```
animation-duration:  4s, 2000ms;
```

ولاحظ أن التحريك الأول يستغرق مدة 4s والتحريك الثاني يستغرق مدة 2s.

animation-delay

خاصية تحدد فترة التأخير قبل بدء عملية التحريك وتكون عبارة عن رقم يتبعه رمز الفترة الزمنية إما s (second ثانية) أو ms (millisecond ملي ثانية) وفي حالة تطبيق أكثر من تحريك للعنصر يكون لكل تحريك فترة تأخير البدء الخاصة به ويتم الفصل بين كل فترة تأخير والتي تليها بفاصلة، ويمكن الاكتفاء بفترة تأخير واحدة لكل الخصائص إذا تساوت فترات التأخير في القيمة والخاصية غير وراثية ولها قيمة افتراضية 0s.

أضف الكود التالي للكود السابق للعنصر div:

```
animation-delay:  3s, 1000ms;
```

لاحظ تأخر بداية التحريك الأول 3s وتأخر بداية التحريك الثاني 1s.

animation-timing-function

خاصية تزامن التحريك من حيث التسارع والتباطؤ طوال فترة التحريك، والخاصية غير وراثية ولها قيمة افتراضية ease والخاصية تقبل عدة أنواع من القيم، وهي نفس أنواع قيم الخاصية transition-timing-function التي سبق دراستها فلا حاجة لإعادة شرحها ويمكن الرجوع إليها في شرح الخاصية transition.

أضف الكود التالي للكود السابق للعنصر div:

```
animation-timing-function: ease-out, ease-in;
```

لاحظ اختلاف التسارع والتباطؤ لكل تحريك عن الآخر وداخل التحريك نفسه.

animation-iteration-count

خاصية تحدد عدد مرات تكرار التحريك، وهي خاصية غير وراثية ولها قيمة افتراضية 1 وتقبل نوعين من القيم كالتالي:

- *number* : رقم يدل على عدد مرات التكرار وقد يكون الرقم جزء من الواحد.
- *infinite* : وتعني أن التحريك يتكرر بلا نهاية.

أضف الكود التالي للكود السابق للعنصر div:

```
animation-iteration-count: 3, infinite;
```

لاحظ أن التحريك الأول تكرر 3 مرات ثم توقف والتحريك الثاني يتكرر لما لا نهاية.

animation-play-state

خاصية تحدد إمكانية توقف التحريك أو استمراره وهي خاصية غير وراثية وتقبل نوعين من القيم كالتالي:

- *running* : القيمة الافتراضية وتعني أن التحريك يعمل بشكل طبيعي حسب الإعدادت الخاصة به.
- *paused* : وتعني تعطيل وتوقف تحريك العنصر.

أضف الكود التالي للكود السابق للعنصر div ولاحظ توقف تحريك العنصر.

```
animation-play-state: paused;
```

animation-direction

خاصية تحدد اتجاه عملية تحريك العنصر، والخاصية غير وراثية.

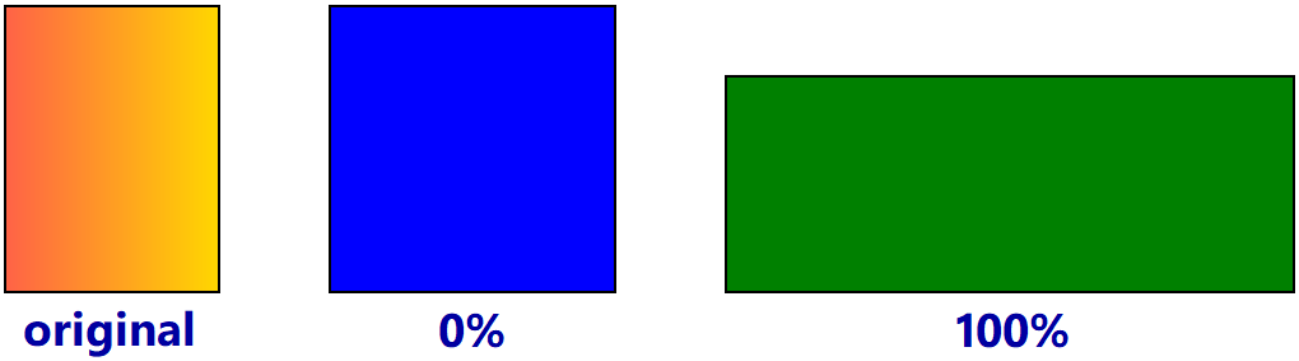
ملف html

```
<div></div>
```

ملف css

```
div{
    width: 75px;
    height: 100px;
    background: linear-gradient(to right, tomato, gold);
    margin: 40px;
    border: 1px solid;
    animation-name:  animation-name;
    animation-duration:  3s;
    animation-iteration-count:  3;
    animation-direction:  normal;
   }
@keyframes animation-name{
       from{
            width: 100px;
            height: 100px;
            background: blue;
           }
       to{
          width: 200px;
          height: 75px;
          background: green;
          }
       }
```

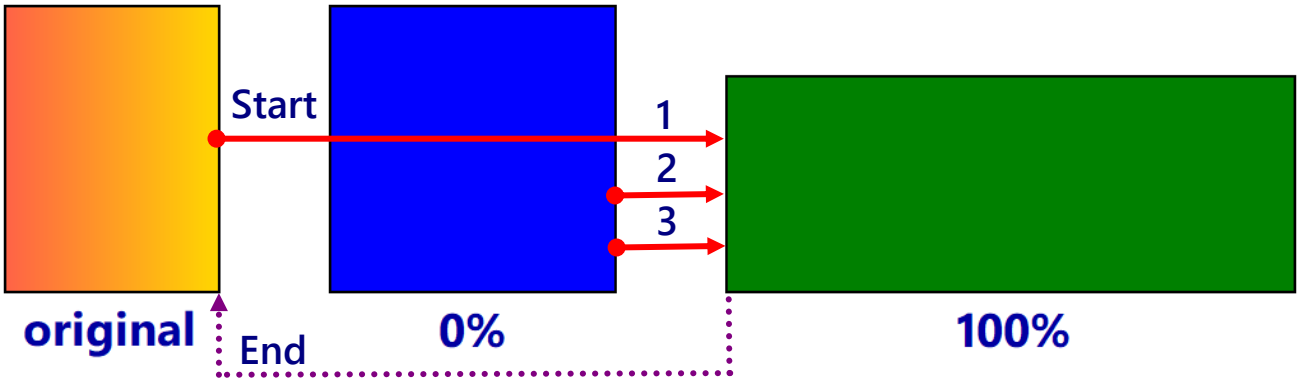
الشكل التالي يوضح وضع العنصر في جميع أحواله، أولاً وضعه الأصلي original قبل بداية عملية التحريك، وثانياً وضعه في الإطار الأول في بداية عملية التحريك (المرحلة from أو 0%) وثالثاً وضع العنصر في الإطار الأخير في نهاية عملية التحريك (المرحلة to أو 100%) كالتالي:



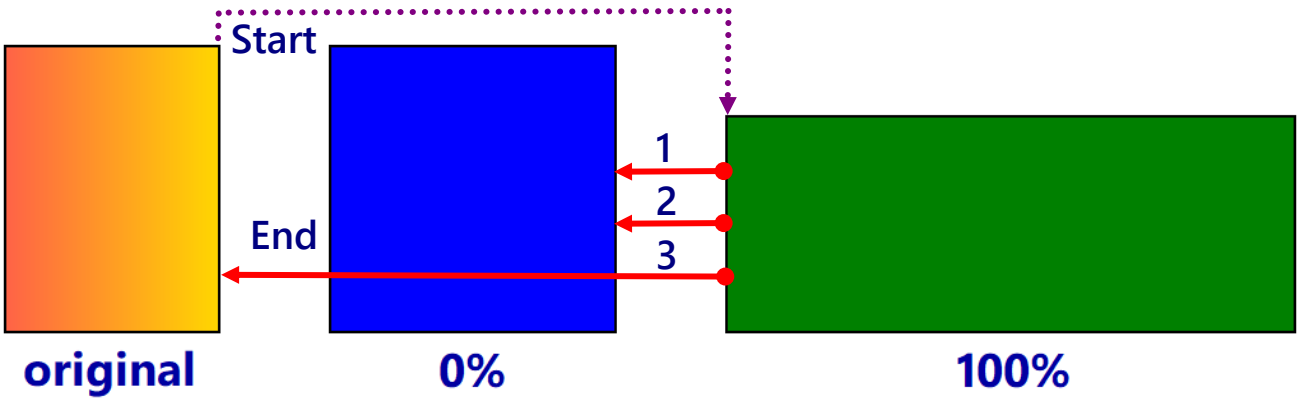
والخاصية animation-direction تقبل عدة أنواع من القيم كالتالي:

– *normal* : اتجاه تحريك العنصر هو نفسه اتجاه ترتيب المراحل الزمنية لعملية التحريك حيث يبدأ من مرحلة البدء 0% أو from ثم يتجه إلى مرحلة نهاية التحريك 100% أو to ثم يعود فجائياً لوضع العنصر الأصلي قبل بدء عملية التحريك.

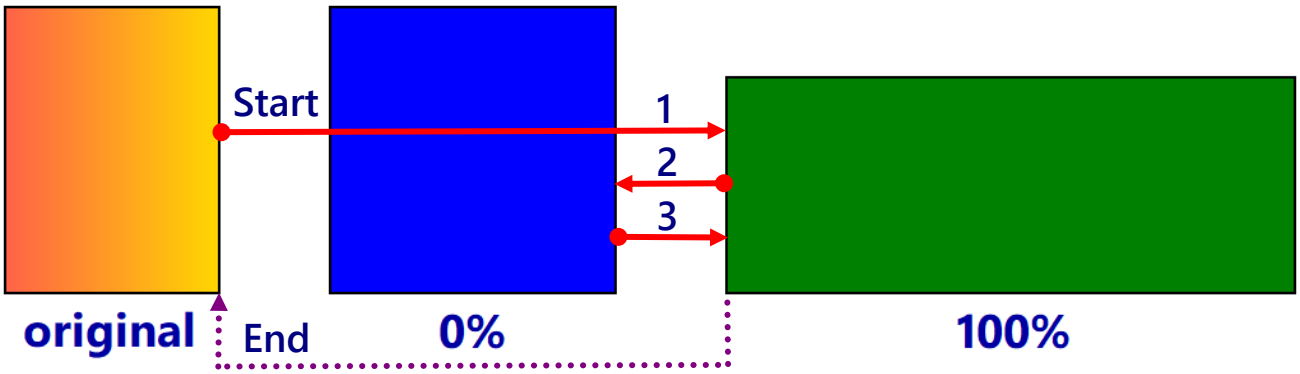
لاحظ الشكل التالي مع العلم أن عدد مرات تكرار تحريك العنصر هي 3 مرات كالتالي:



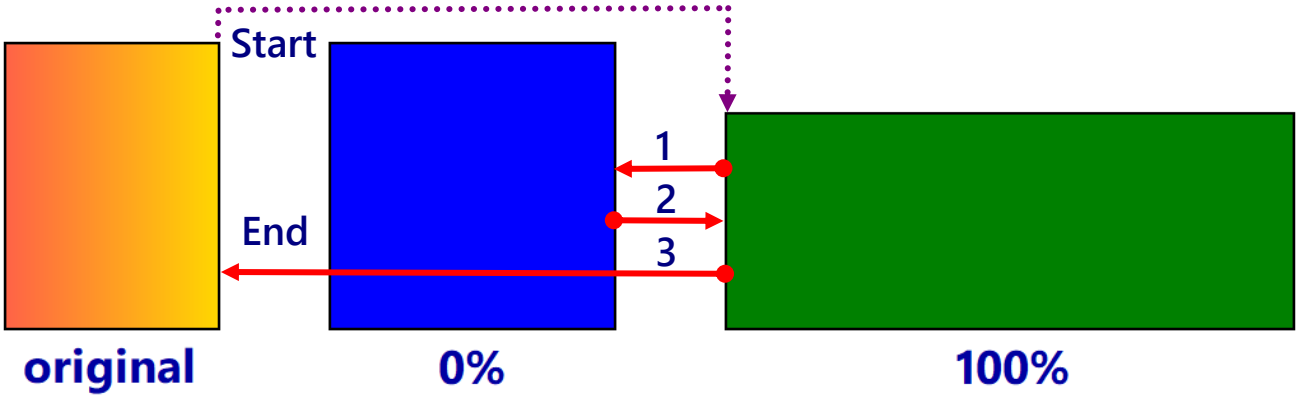
– *reverse* : اتجاه التحريك يكون عكس مراحل التحريك، حيث ينتقل العنصر إلى الإطار الأخير فجائياً ثم يبدأ التحريك من المرحلة الأخيرة (100% أو from) وينتهي بمرحلة البدء (0% أو to) وبعد انتهاء التكرارات يعود فجائياً إلى الوضع الأصلي.



– ***alternate*** : اتجاه التحريك هو نفس اتجاه مراحل التحريك ولكن بعد وصول التحريك لمرحلة النهاية 100% لا يعود العنصر فجائياً لوضعه الأصلي، ولكن يستمر التحريك في الاتجاه العكسي أي عكس مراحل التحريك ويستغرق نفس الفترة الزمنية للتحريك، ويتم خصم الرجوع العكسي من عدد مرات التكرار، أي أن عدد التكرارات يجب أن يكون أكبر من 1 لتطبيق هذه القيمة، أما إذا تكرر التحريك مرة واحدة فقط فسيكتفي التحريك بالمرحلة الأولى فقط وهي الوصول لمرحلة النهاية 100% ثم يعود بشكل مفاجئ لوضعه الأصلي ولن يكمل الرجوع العكسي (نفس سلوك الخاصية normal).



– ***alternate-reverse*** : وهي قيمة تجمع القيمتين reverse و alternate حيث يبدأ التحريك عكسياً من مرحلة 100% وينتهي بمرحلة البدء 0% ثم يتجه عكس هذا الاتجاه (الاتجاه الأصلي للتحريك) من مرحلة البدء إلى مرحلة النهاية، ثم يعود فجائياً إلى وضعه الأصلي ويجب أن يتوافر شرط زيادة عدد مرات التكرار عن 1 وإلا فإنه سيسلك نفس سلوك القيمة reverse.



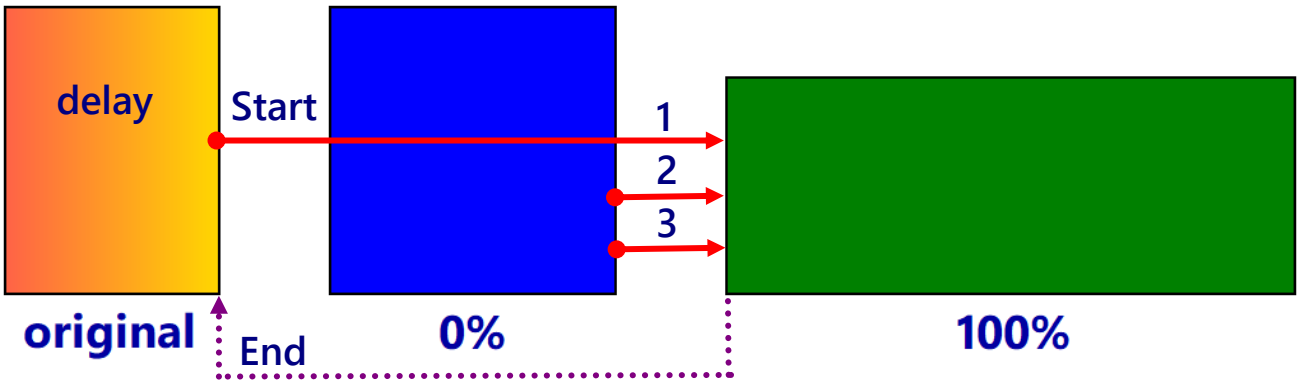
animation-fill-mode

خاصية تحدد وضع العنصر بعد انتهاء التحريك، ويعتمد ذلك على قيمة خاصيتي animation-direction و animation-iteration-count والخاصية غير وراثية وتقبل عدة أنواع من القيم كالتالي:

– ***none*** : مهما كان اتجاه التحريك أو مهما كان عدد مرات تكرار التحريك فإن العنصر سينتظر فترة التأخير على وضعه الأصلي قبل الدخول إلى الإطار الأول في مرحلة البدء (from أو 100%) وبعد انتهاء عملية التحريك فإن ال عنصر يتوقف دائماً عند وضعه الأصلي قبل الدخول إلى الإطار الأول لمرحلة البدء.

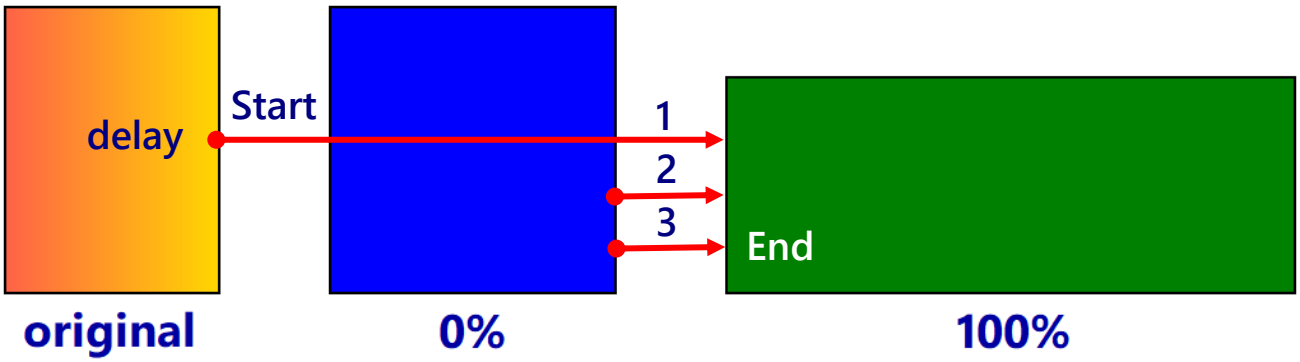
أضف الكود التالي للكود السابق للعنصر div:

```
    animation-delay: 2s;
    animation-fill-mode:  none;
```

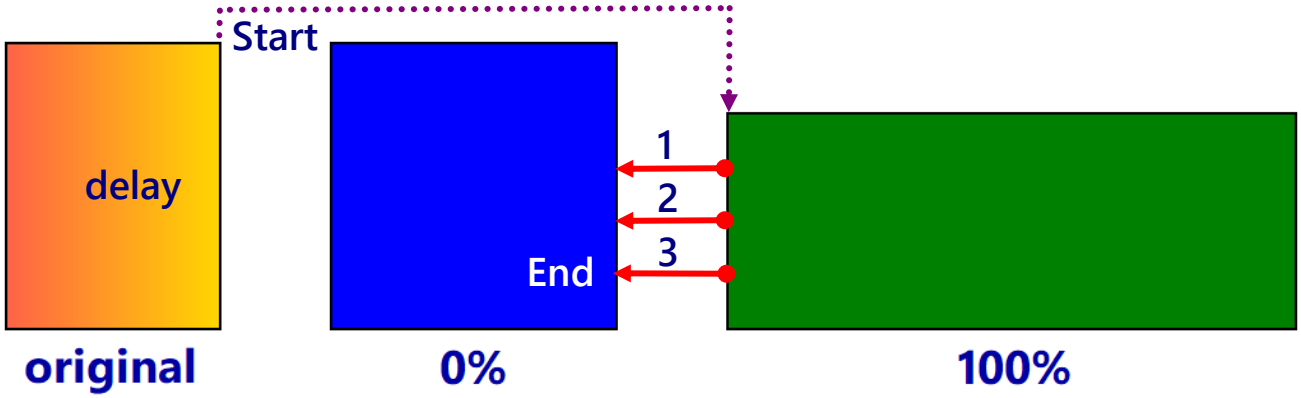


– ***forwards*** : العنصر ينتظر فترة التأخير على وضعه الأصلي قبل الدخول إلى إلى عملية التحريك أياً كان اتجاه بدء عملية التحريك (normal أو reverse)، كما أن العنصر يتوقف دائماً عند وضعه في آخر إطار ينتهي عنده التحريك ويتوقف ذلك على اتجاه التحريك وعدد مرات تكرار التحريك كالتالي:

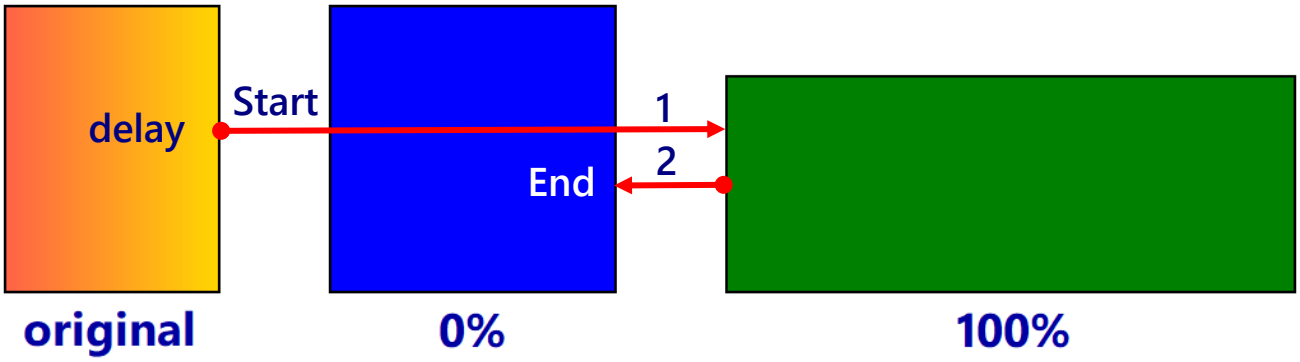
1– normal: إذا كان اتجاه التحريك طبيعياً وأياً كان عدد مرات التكرار فسيتوقف العنصر عند وضعه في الإطار الأخير لمرحلة نهاية التحريك (to أو 100%).



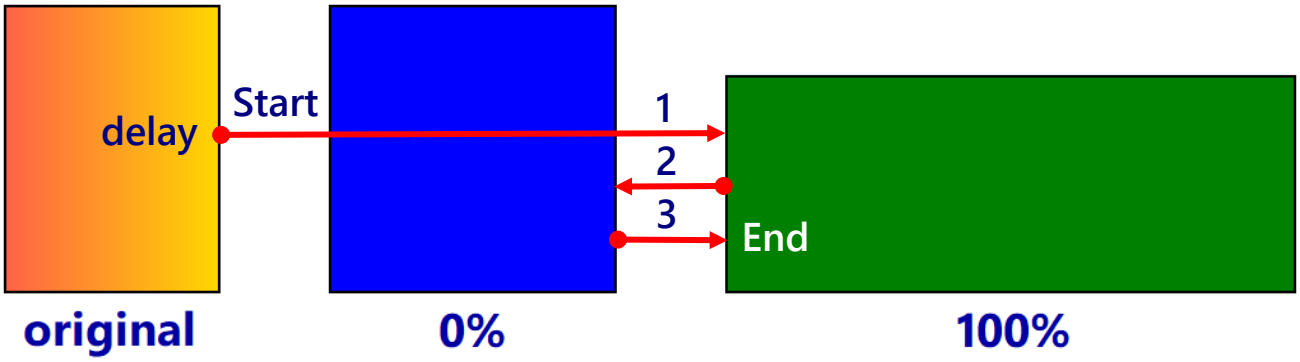
2- reverse : إذا كان اتجاه التحريك عكسياً فسيتوقف العنصر عند وضعه في الإطار الأول لمرحلة بدء التحريك أياً كان عدد مرات التكرار.



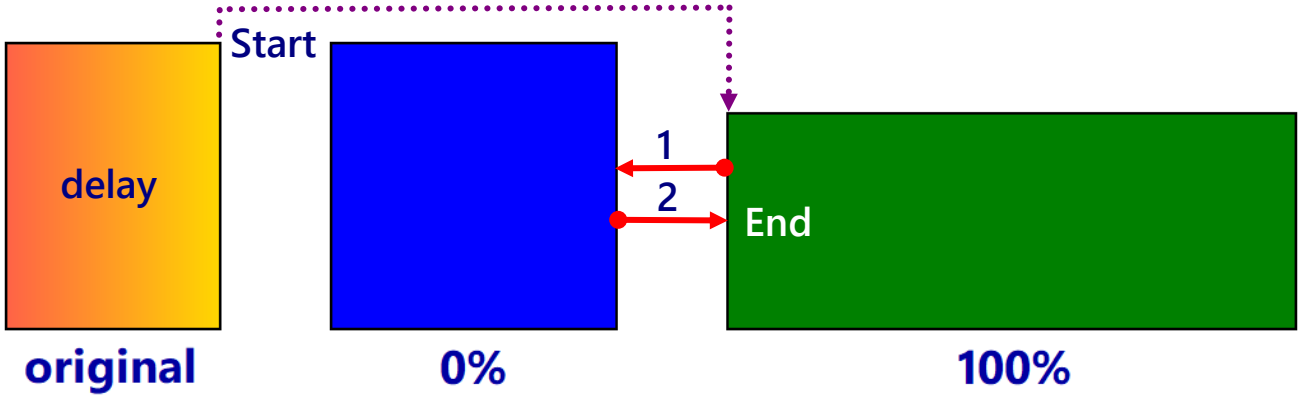
3- alternate : إذا كان اتجاه التحريك تبادلياً، فسيتوقف العنصر عند وضعه في الإطار الأول لمرحلة بدء التحريك إذا كان عدد مرات التكرار زوجياً.



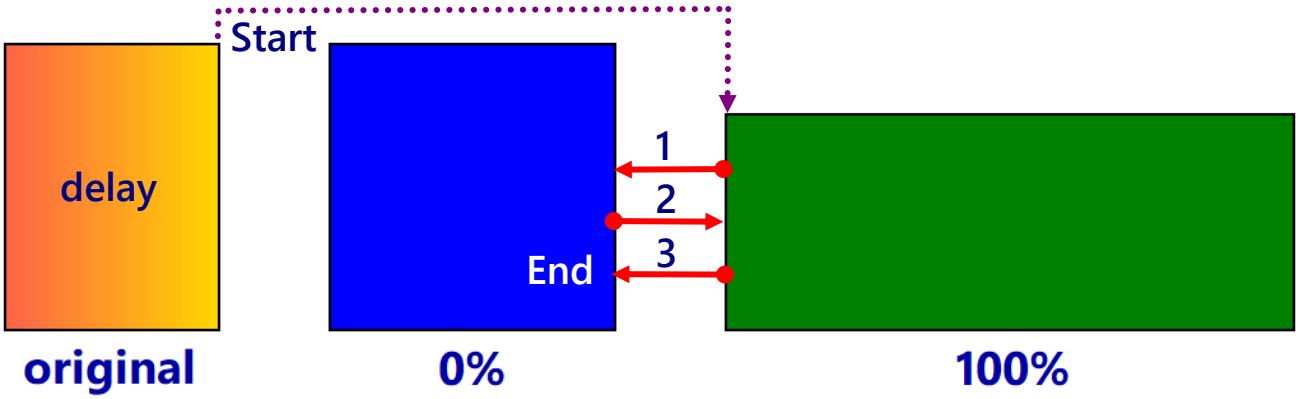
وسيتوقف عند وضعه في الإطار الأخير لمرحلة النهاية إذا كان عدد مرات التكرار فردياً.



4- alternate-reverse : إذا كان اتجاه التحريك تبادلياً عكسياً، فسيتوقف العنصر عند وضعه في الإطار الأخير لمرحلة النهاية إذا كان عدد مرات التكرار زوجياً.

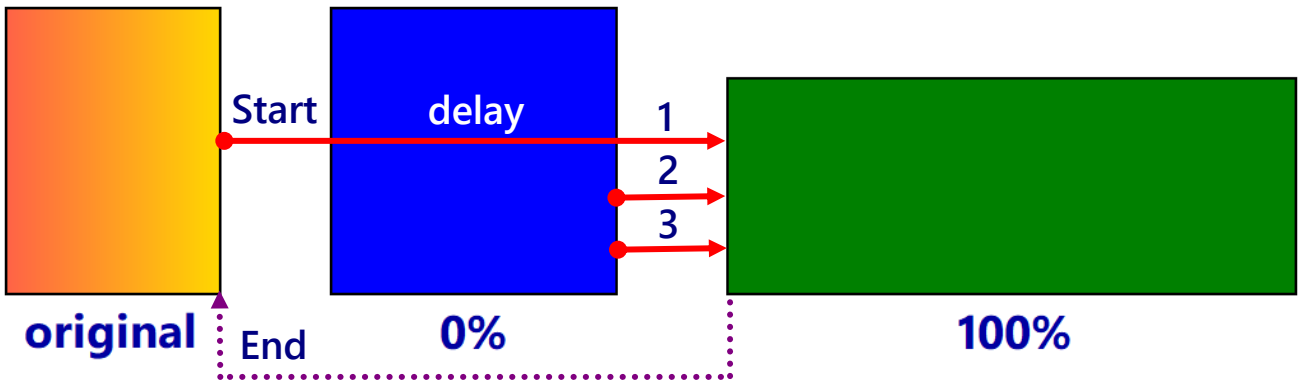


وسيتوقف عند وضعه في الإطار الأول لمرحلة البدء إذا كان عدد مرات التكرار فردياً.

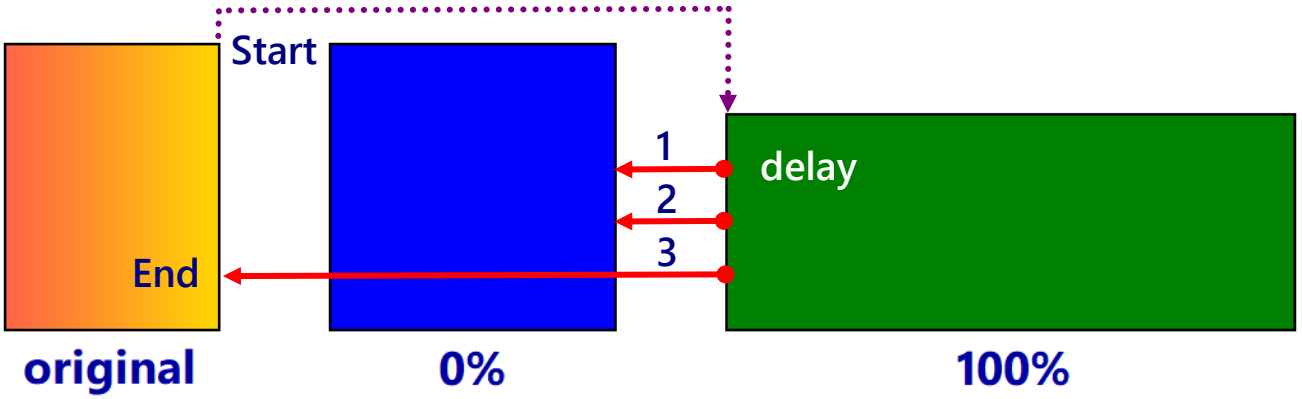


- *backwards* : يتوقف العنصر عند وضعه الأصلي قبل بدء عملية التحريك أياً كان اتجاه التحريك وأياً كان عدد مرات التحريك، ولكن الفرق كالتالي:

أن العنصر عند بداية التحريك يقفز إلى الإطار الأول في مرحلة بدء التحريك ثم ينتظر فترة التأخير قبل بدء الحركة إذا كان اتجاه التحريك normal أو alternate.

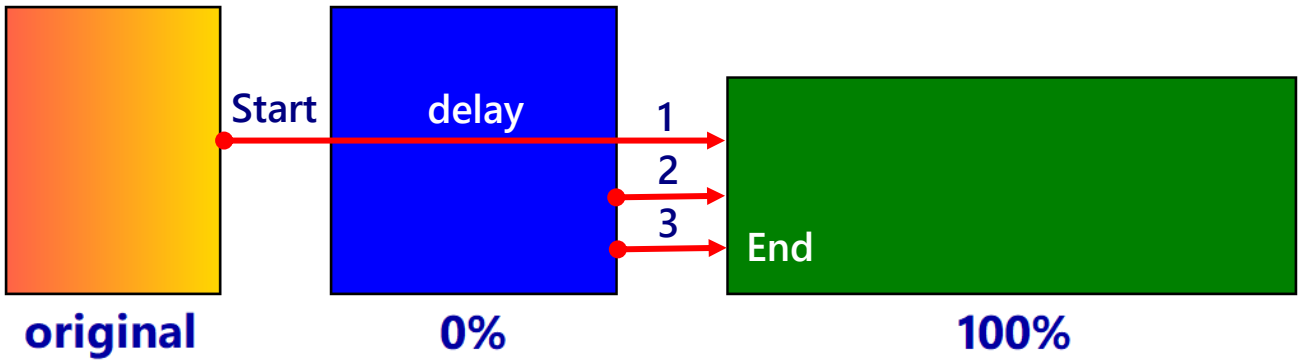


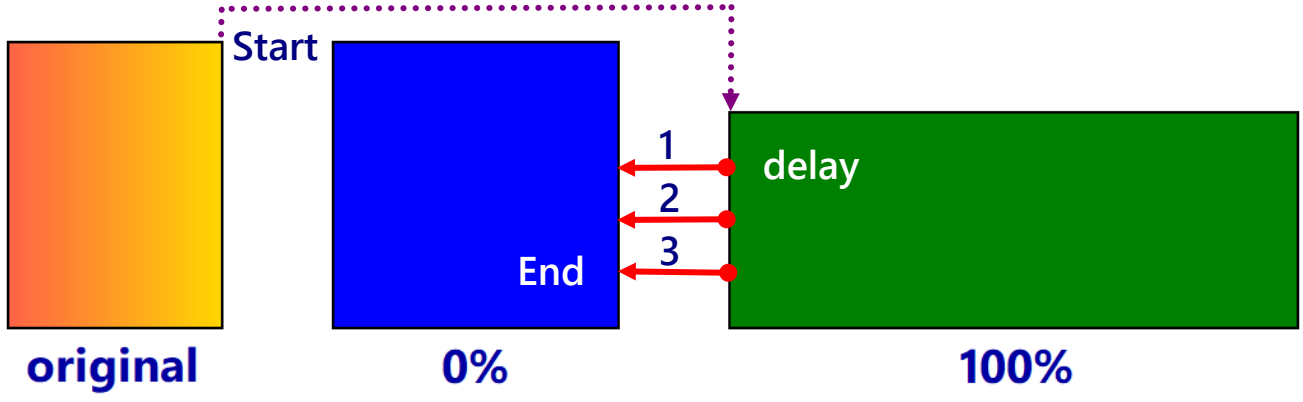
أما إذا كان اتجاه التحريك reverse أو alternate-reverse فإن العنصر يقفز إلى الإطار الأخير في مرحلة نهاية التحريك ثم ينتظر فترة التأخير قبل بدء الحركة.



وفي كلا الحالتين وبعد انتهاء عملية التحريك يعود العنصر إلى وضعه الأصلي.

– ***both*** : هذه القيمة تشبه القيمة backwards في أنها عند بداية الحركة يقفز العنصر إلى أول إطار في عملية التحريك حسب اتجاه تحريك العنصر السابق شرحة في القيمة backwards. وتشبه القيمة forwards في أنها بعد انتهاء الحركة لا يعود العنصر لوضعه الأصلي قبل عملية التحريك ولكن يظل محتفظاً بوضعه الذي كان عليه في آخر إطار للحركة حسب اتجاه التحريك السابق شرحه في القيمة forwards.





وكما سبق أن ذكرنا، فالخاصية animation هي اختصار كل الخصائص السابقة وقيمتها يمكن أن تجمع كل أو بعض هذه الخصائص، والخصائص التي لا يتم إسناد قيمة لها تكون ممثلة بقيمتها الافتراضية كالتالي:

```
animation: name 0s 0s ease 1 running normal none;
```

وتوضيح قيم الخاصية كالآتي:

name قيمة الخاصية animation-name.

0s قيمة الخاصية animation-duration ويجب تحديد قيمة أكبر من 0s حتى يبدأ التحريك ويسلك مراحله المتعددة ويجب أن تتواجد في القيمة في حالة وجود قيمة الخاصية animation-delay.

0s قيمة الخاصية animation-delay ويجب أن يسبقها في الترتيب قيمة الخاصية animation-duration وفي حالة وجود قيمة واحدة فسيتم احتسابها قيمة للخاصية animation-duration التي لا يمكن للتحريك أن يبدأ بدون تخصيص قيمتها.

ease قيمة الخاصية animation-timing-function.

1 قيمة الخاصية animation-iteration-count.

running قيمة الخاصية animation-play-state.

normal قيمة الخاصية animation-direction.

none قيمة الخاصية animation-fill-mode.

لا يشترط وجود خاصية معينة لبدأ التحريك باستثناء الخاصية animation-duration ولا يشترط الترتيب بين قيم الخصائص باستثناء قيمة الخاصية animation-duration التي يجب أن تسبق قيمتها قيمة الخاصية animation-delay، وعلى ذلك فسلوك الكود التالي هو نفسه سلوك الكود الذي يليه:

```
animation: name 3s 2s 2 ease running reverse both;
```

```
animation: both reverse running name 3s 2s 2 ease;
```