Transform

تعريف: "Transform" هو مصطلح عام يُستخدم لوصف أي عملية تُغير من مظهر أو وضعية عنصر معين. يمكن أن يعني ذلك نقل العنصر، تكبيرة، تدويره، أو تغيير شكله.

Translate

تعريف: تُعني نقل العنصر من مكانه الأصلي إلى مكان آخر. يتم تحديد النقطة الجديدة بناءً على إحداثيات X و Y.

transform: translate(X , Y);

مثال: إذا كان لديك مربع (Square) في صفحة ويب وأردت نقله إلى اليمين بمقدار 50 بيكسل وإلى الأسفل بمقدار 100 بيكسل، ستستخدم:

transform: translate(50px, 100px);

TranslateX

تعريف: هذه الخاصية تُستخدم لتحريك (نقل) العنصر أفقيًا على المحور X (يمينًا أو يسارًا). عندما تقوم بتحديد قيمة لـ translateX، فإن العنصر سيتحرك إما باتجاه اليمين (إذا كانت القيمة موجبة) أو باتجاه اليسار (إذا كانت القيمة سالبة).

مثال:

```
transform: translateX(50px);
```

في هذا المثال، يتم نقل العنصر 50 بيكسل إلى اليمين، أما إذا أردت تحريكه إلى اليسار، يمكنك استخدام:

```
transform: translateX(-50px);
```

TranslateY

تعريف: هذه الخاصية تُستخدم لتحريك (نقل) العنصر عموديًا على المحور Y (للأعلى أو للأسفل). عندما تقوم بتحديد قيمة لـ translateY، فإن العنصر سيتحرك إما باتجاه الأسفل (إذا كانت القيمة موجبة) أو باتجاه الأعلى (إذا كانت القيمة سالبة).

مثال:

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

في هذا المثال، يتم نقل العنصر 100 بيكسل إلى الأسفل، أما إذا أردت تحريكه للأعلى، يمكنك استخدام:

transform: translateY(-100px);

Scale

خاصية scale في CSS تُستخدم لتكبير أو تصغير حجم عنصر معين. يمكن أن تُطبق على محور واحد فقط (أفقي أو عمودي) أو على المحورين معًا. يتم تغيير حجم العنصر بناءً على نسب محددة، دون تغيير موقعه.

القيم التي تُعطى لـ scale هي عادة أرقام موجبة تمثل النسبة التي سيتم تكبير أو تصغير العنصر بها.

القيمة 1: تعني الحجم الأصلي للعنصر (بدون تغيير).

أكبر من 1: تعني تكبير العنصر. على سبيل المثال scale(2) تقوم بتكبير حجم العنصر بمقدار الضعف.

أقل من 1 ولكن أكبر من 0: تعني تصغير العنصر. على سبيل المثال، scale(0.5) يقلل حجم العنصر إلى نصف حجمه الأصلي.

أنواع الـ Scale:

scale (محورين معًا)

يتم تكبير أو تصغير العنصر على المحورين X و Y معًا.

مثال:

```
transform: scale(2);
```

هذا الكود سيقوم بتكبير العنصر على المحورين الأفقي والعمودي بنفس النسبة (الضعف).

scaleX (المحور الأفقي)

يتم تكبير أو تصغير العنصر على المحور X فقط.

مثال:

```
transform: scaleX(1.5);
```

هذا الكود سيقوم بتكبير العنصر أفقيًا بنسبة 1.5، بينما يظل حجمه العمودي كما هو.

scaleY (المحور العمودي)

يتم تكبير أو تصغير العنصر على المحور Y فقط.

مثال:

```
transform: scaleY(0.5);
```

هذا الكود سيقلل حجم العنصر عمودياً إلى نصفه، بينما يظل حجمه الأفقي كما هو.

Rotate

خاصية rotate في CSS تُستخدم لتدوير عنصر حول نقطة الأصل (Origin) الخاصة به، والتي تكون عادةً في مركز العنصر. هذه الخاصية تتيح لك تدوير العناصر حول المحور الأفقي والمحور العمودي أو حول محورين معًا.

rotate()

تُستخدم لتدوير العنصر حول المحور Z (وهو المحور الذي يمتد من الشاشة نحوك أو بعيدًا عنك).

ما هي الوحدات التي تستخدم مع ال rotate؟

deg

ال 360deg تعني دورة كاملة.

turn

ال 1turn تعني الدورة الكاملة ، و 0.5turn تعني نصف دورة.

مثال:

```
transform: rotate(90deg);
```

هذا الكود سيقوم بتدوير العنصر بزاوية 90 درجة (ربع دورة) حول مركزه.

transform: rotate(0.5turn);

هذا الكود سيقوم بتدوير العنصر نصف دورة، وهو ما يعادل 180deg.

rotateX()

تُستخدم لتدوير العنصر حول المحور X (المحور الأفقي)..

أمثلة:

transform: rotateX(45deg);

هذا الكود سيقوم بتدوير العنصر بزاوية 45 درجة حول المحور الأفقي، مما يؤدي إلى ميل العنصر عموديًا.

transform: rotateX(1turn);

هذا الكود سيقوم بتدوير العنصر دورة كاملة حول المحور X (دورة واحدة).

rotateY

تُستخدم لتدوير العنصر حول المحور Y (المحور العمودي).

أمثلة:

transform: rotateY(90deg);

هذا الكود سيقوم بتدوير العنصر بزاوية 90 درجة حول المحور العمودي، مما يجعل أحد جانبي العنصر يظهر بينما يختفي الآخر تدريجيًا.

transform: rotateY(0.5turn);

هذا الكود سيقوم بتدوير العنصر نصف دورة حول المحور Y، مما يؤدي إلى إظهار الوجه الخلفي للعنصر.

rotateZ()

يعمل تمامًا مثل rotate، حيث يدور العنصر حول المحور Z. يتم تطبيقه بشكل رئيسي في التأثيرات ثلاثية الأبعاد.

أمثلة:

transform: rotateZ(60deg);

هذا الكود يدور العنصر بزاوية 60 درجة حول المحور Z.

Skew

خاصية skew في CSS تُستخدم لإمالة عنصر ما على محورين (المحور الأفقي X والمحور العمودي Y). حيث تُؤدي إلى تحريف الشكل الأصلي للعنصر بحيث يبدو مائلاً بزاوية معينة.

skewX()

تُستخدم لإمالة العنصر على المحور الأفقي X.

عندما تطبق زاوية باستخدام skewX، فإن الجزء العلوي أو السفلي من العنصر يتم إمالته إلى اليمين أو اليسار حسب قيمة الزاوية.

أمثلة:

```
transform: skewX(30deg);
```

skewY()

تُستخدم لإمالة العنصر على المحور العمودي Y.

عندما تطبق زاوية باستخدام skewY، فإن الجوانب اليمنى أو اليسرى من العنصر يتم إمالتها لأعلى أو لأسفل حسب قيمة الزاوية.

أمثلة:

transform: skewY(45deg);

skew()

تُستخدم لإمالة العنصر على المحورين X و Y معًا.

يمكنك تحديد زاويتين: واحدة للمحور X وواحدة للمحور Y.

أمثلة:

transform: skew(30deg, 20deg);

عند استخدام خاصية skew في CSS، إذا قمت بتمرير زاوية واحدة فقط مثل:

transform: skew(20deg,);

فهذا يعني أنك قمت بتحديد زاوية 20deg للإمالة على المحور الأفقي X فقط.

Transform-origin

خاصية transform-origin في CSS تُحدد نقطة الأصل (Origin) التي يتم عندها تطبيق تأثيرات التحويلات (Transformations) مثل rotate, scale, skew, وtranslate.

بعبارة أخرى، هي النقطة التي حولها يُدور أو يُكبر أو يُموج أو يُنقل العنصر.

بشكل افتراضي (Default)، نقطة الأصل (Origin) لـ transform تكون في مركز العنصر (Center). أي أنه إذا قمت بتدوير أو تكبير العنصر، فسوف يتم ذلك حول مركزه (Center).

مثال1:

```
transform-origin: center;
```

center

نقطة الأصل (Origin) في منتصف العنصر (افتراضي).

ال top, bottom, left, right تحديد نقطة الأصل على الحواف العلوية أو السفلية أو اليسرى أو اليمنى من العنصر.

مثال2:

```
transform-origin: top;
```

مثال3:

تدوير حول الزاوية العلوية اليسرى:

```
transform-origin: top left;

transform: rotate(45deg);
```

في هذا المثال، العنصر سيتم تدويره بزاوية 45 درجة حول الزاوية العلوية اليسرى.

مثال4:

تكبير حول الزاوية السفلية اليمنى:

```
transform-origin: bottom right;

transform: scale(1.5);
```

هنا، سيتم تكبير العنصر بنسبة 1.5 مرة حول الزاوية السفلية اليمنى.

Combining Multiple Operations

في CSS، يمكنك دمج عمليات تحويل متعددة (Transform) على عنصر واحد لجعل التأثيرات أكثر تفاعلية. بدلاً من تطبيق عملية تحويل واحدة فقط مثل rotate أو scale، يمكنك تطبيق عدة عمليات تحويل في سطر واحد باستخدام خاصية transform. يتم تطبيق هذه العمليات بالتسلسل الذي تكتبه.

مثال:

```
transform: rotate(45deg) scale(1.5) translateX(50px);
```

في المثال يتم:

تدوير العنصر بزاوية 45 درجة.

تكبير العنصر بمقدار 1.5 مرة.

نقل العنصر 50px إلى اليمين.

Transition

الـ transition في CSS هي خاصية تُستخدم لإبطاء التأثيرات عند تغيير قيم خصائص CSS على عنصر معين. بدلاً من أن يحدث التغيير بشكل مفاجئ وفوري، تقوم خاصية transition بإبطاء عملية التغيير على مدار فترة زمنية محددة، مما يعطي تأثيرًا مرئيًا.

تفاصيل مكونات الـ Transition:

الـ transition يتكون من عدة أجزاء يمكن تحديدها:

1- الخاصية (Property): تُحدد الخاصية التي ترغب في تطبيق الـ transition عليها مثل width، transform، إلخ.

2- المدة (Duration): تحدد المدة التي يستغرقها الانتقال من الحالة القديمة إلى الحالة الجديدة. تُكتب المدة بوحدات مثل ms (ميلي ثانية) أو s (ثانية).

أمثلة على استخدام transition:

transition: transform 250ms

في هذا المثال، يتم تطبيق الـ transition على خاصية transform فقط، والمدة الزمنية للانتقال هي 250ms (ربع ثانية).

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 250ms;
}
.box:hover {
  transform: scale(1.5);
}
```

الشرح: عند تمرير الماوس فوق العنصر (الذي يحتوي على box class)، سيتم تكبير العنصر باستخدام خاصية transform: scale(1.5). هذا التكبير لن يحدث فورًا، بل سيحدث بشكل سلس على مدار 250 ملي ثانية.

```
transition: all 250ms
```

في هذا المثال، يتم تطبيق الـ transition على جميع خصائص العنصر التي يمكن أن تتغير.

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: all 250ms;
}

.box:hover {
  width: 150px;
  height: 150px;
  background-color: red;
}
```

الشرح: عند تمرير الماوس فوق العنصر، سيتم تكبير حجمه وتغيير لونه. كل هذه التغييرات ستحدث بشكل سلس خلال 250 ملي ثانية.

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 250ms, width 1s;
}
.box:hover {
  width: 200px;
  transform: rotate(45deg);
}
```

الشرح: عند تمرير الماوس فوق العنصر:

سيتم تدوير العنصر بزاوية 45 درجة باستخدام transform: rotate(45deg). هذه العملية ستستغرق 250 ملي ثانية.

سيتم تغيير عرض (Width) العنصر من 100px إلى 200px. هذه العملية ستستغرق ثانية واحدة كاملة.

Timing Functions

تُستخدم ال Timing Functions للتحكم في كيفية تسريع أو تباطؤ تأثيرات التحولات (transitions) على مدار فترة زمنية معينة. هذه الدوال تحدد كيفية توزيع الوقت خلال التحول أو الحركة، مما يؤثر بشكل مباشر على كيفية إدراك التأثيرات البصرية.

أشهر Timing Functions:

- linear
- ease
- ease-in
- ease-out
- ease-in-out
- steps()
- Custom cubic-bezier

linear

التأثير: يحدث التغيير بوتيرة ثابتة، مما يعني أن السرعة تظل ثابتة من البداية إلى النهاية.

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s linear;
}
.box:hover {
  transform: translateX(200px);
}
```

- **الشرح:** عند تمرير الماوس فوق العنصر، سيتحرك بشكل ثابت (بدون تسارع أو تباطؤ) نحو اليمين لمسافة 200px.

ease-out

شرح:

التأثير: يبدأ التحول بسرعة ثم يبطئ تدريجياً نحو النهاية.

الاستخدام: مناسب للمواقف التي تريد فيها أن يبدو العنصر كما لو أنه يتباطأ قبل التوقف.

مثال:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s ease-out;
}
.box:hover {
  transform: translateX(200px);
}
```

الشرح: عند تمرير الماوس فوق العنصر، سيتحرك بسرعة في البداية، ثم يتباطأ تدريجياً قبل أن يصل إلى الموضع النهائي.

ease-in

التأثير: يبدأ التحول ببطء ثم تزداد السرعة تدريجياً نحو النهاية.

الاستخدام: مناسب للمواقف التي تريد فيها أن يظهر العنصر كما لو أنه يتسارع من حالة السكون.

مثال:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s ease-in;
}
.box:hover {
  transform: translateX(200px);
}
```

الشرح: عند تمرير الماوس فوق العنصر، سيتحرك ببطء في البداية، ثم تزداد سرعته تدريجياً حتى يصل إلى الموضع النهائي.

ease-in-out

التأثير: يبدأ التحول ببطء، يسرع في المنتصف، ثم يبطئ في النهاية.

مثال:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s ease-in-out;
}

.box:hover {
  transform: translateX(200px);
}
```

الشرح: عند تمرير الماوس فوق العنصر، سيبدأ التحول ببطء، ثم يسرع في الوسط، ويبطئ مرة أخرى قبل أن يصل إلى الموضع النهائي.

ease

هذه هي القيمة الافتراضية لـ timing function في CSS.

الاستخدام: شائع جداً للاستخدامات العامة لأنه يعطي تأثيرًا سلسًا وجذابًا.

مثال:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s ease;
}
.box:hover {
  transform: translateX(200px);
}
```

steps()

التأثير: يقسم التحول إلى عدد معين من الخطوات بدون أي تغيير تدريجي بين الخطوات.

الشرح: بدلاً من انتقال سلس، يقفز العنصر من حالة إلى أخرى بناءً على عدد الخطوات المحددة.

steps(number_of_steps, direction)

ال number_of_steps: عدد الخطوات التي سيتم تقسيم التحول إليها.

ال direction (اختياري): يمكن أن يكون start أو end.

مثال:

```
.box {
  transition: transform 2s steps(5, end);
}
.box:hover {
  transform: translateX(200px);
}
```

النتيجة: عند تمرير الماوس فوق العنصر، سيتحرك في خمس خطوات منفصلة دون أي تغيير سلس بين هذه الخطوات.

Custom cubic-bezier

التأثير: تسمح لك هذه الطريقة بتحديد منحنى مخصص (Custom curve) للتحكم في تسارع وتباطؤ التحول بدقة.

الاستخدام: يستخدم عندما تحتاج إلى التحكم الكامل في توقيت التحولات.

cubic-bezier(x1, y1, x2, y2)

حيث تمثل القيم الأربع النقاط المحورية للمنحنى.

مثال:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s cubic-bezier(0.68, -0.55, 0.27, 1.55);
}
.box:hover {
  transform: translateX(200px);
```

}

الشرح: هذا المثال يطبق منحنى مخصص للتحول، مما يتيح تأثيرًا غير تقليدي حيث قد يسرع العنصر بشكل كبير ثم يتباطأ بسرعة قبل أن يصل إلى موضعه النهائي.

Transition-delay

في CSS، خاصية transition-delay تُستخدم لتحديد المدة الزمنية التي يجب أن تمر قبل أن يبدأ التحول (transition) في الحدوث بعد حدوث تفاعل معين (مثل تحريك الفأرة فوق العنصر).

يمكن أن تكون قيمة (Value) ال transition-delay إما موجبة أو سالبة:

قيمة موجبة: تعني أن التحول سيبدأ بعد مرور هذه المدة الزمنية.

قيمة سالبة: تعني أن التحول سيبدأ قبل حدوث التفاعل.

القيمة الافتراضية هي 0s، أي أن التحول يبدأ فورًا دون أي تأخير.

مثال:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: transform 2s;
  transition-delay: 1s;
```

```
}

.box:hover {
  transform: scale(1.5);
}
```

الشرح: في هذا المثال، عند تمرير الماوس فوق العنصر الذي يحتوي على "class "box، سيحدث تأخير لمدة 1 ثانية قبل أن يبدأ التحول (الذي يستغرق 2 ثانية) لتكبير العنصر إلى 1.5 ضعف حجمه الأصلي.

ملاحظة هامة:

التأخير لا يؤثر على مدة التحول نفسه، فهو فقط يؤخر بداية التحول.

ما هي ال Keyframes؟

في CSS، الـ keyframes هي طريقة لإنشاء الرسوم المتحركة (Animation) التي تتغير على مراحل معينة. كل مرحلة تُسمى "نقطة أساسية" (Keyframe)، حيث تقوم بتحديد الوضع الذي يجب أن تكون عليه العناصر في تلك النقطة المحددة من الرسوم المتحركة (Animation). تُستخدم هذه المراحل لضبط خصائص العنصر مثل الموضع، الحجم، اللون، والشفافية وغيرها على مدار فترة زمنية.

كيف تعمل الـ Keyframes؟

في CSS، عندما نستخدم @keyframes لإنشاء الرسوم المتحركة (Animation)، فإننا نحدد مراحل هذه الرسوم من خلال نقاط أساسية، وهي from و to.

ال **from** تُمثل نقطة البداية للرسوم المتحركة (Animation). أي أن الخصائص التي تكتب داخل from تطبق على العنصر في بداية الرسوم المتحركة (Animation).

ال **to** تمثل نقطة النهاية للرسوم المتحركة (Animation). الخصائص التي تكتب داخل to تطبق على العنصر عندما تصل الرسوم المتحركة (Animation) إلى نهايتها.

مثال:

لنفترض أننا نريد تحريك مربع بحيث يتحرك من اليسار إلى اليمين، ونغير لونه من الأحمر إلى الأخضر خلال مدة الرسوم المتحركة (Animation).

```
@keyframes moveAndChangeColor {
  from {
    transform: translateX(0); /* المربع يبدأ من موقعه الأصلي */
    background-color: red; /* المربع يبدأ باللون الأحمر */
  }
  to {
    transform: translateX(300px); /* المربع يتحرك 300 بكسل إلى اليمين */
    background-color: green; /* المربع يصبح أخضر في النهاية */
  }
}
```

تطبيق الرسوم المتحركة على عنصر:

الآن، نقوم بتطبيق هذه الرسوم المتحركة على عنصر (مثل مربع) باستخدام الخاصية animation:

```
.square {
  width: 100px;
  height: 100px;
  background-color: red;
  animation: moveAndChangeColor 4s ease-in-out;}
```

خصائص الرسوم المتحركة (Animation)

animation-timing-function

هذه الخاصية تحدد سرعة الرسوم المتحركة (Animation) خلال مدتها. بمعنى آخر، هي التي تتحكم في تسريع أو إبطاء الرسوم المتحركة (Animation) في مراحل مختلفة.

القيم التي تأخذها:

ال linear: تجعل الرسوم المتحركة (Animation) تتحرك بسرعة ثابتة من البداية إلى النهاية.

ال ease: هذا هو الإعداد الافتراضي، حيث تبدأ الرسوم المتحركة (Animation) ببطء، ثم تسرع، ثم تبطئ مرة أخرى في النهاية.

ال ease-in: تبدأ الرسوم المتحركة (Animation) ببطء وتسرع مع مرور الوقت.

ال ease-out: تبدأ الرسوم المتحركة (Animation) بسرعة وتتباطأ في النهاية.

ال ease-in-out: تبدأ ببطء، تسرع في الوسط، ثم تتباطأ في النهاية.

مثال:

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

animation-delay

تحدد هذه الخاصية متى ستبدأ الرسوم المتحركة (Animation) بعد تطبيقها على العنصر.

بمعنى آخر، يمكن تأخير بدء الرسوم المتحركة (Animation) لمدة زمنية معينة.

القيم التي تأخذها:

وقت بالثواني أو الميلي ثانية: على سبيل المثال، 2s تعني أن الرسوم المتحركة (Animation) ستبدأ بعد 2 ثانية.

القيم السالبة: إذا حددت قيمة سالبة مثل 1s- فسيبدأ العنصر الرسوم المتحركة (Animation) كما لو أن ثانية قد مرت بالفعل.

مثال:

```
animation-delay: 3s;
```

animation-iteration-count

تحدد هذه الخاصية عدد مرات تكرار الرسوم المتحركة (Animation).

القيم التي تأخذها:

عدد صحيح: مثل 1، 2، إلخ، والذي يمثل عدد المرات التي ستتكرر فيها الرسوم المتحركة (Animation).

ال infinite: تعني أن الرسوم المتحركة (Animation) ستستمر بالتكرار إلى ما لا نهاية.

مثال:

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

animation-fill-mode

هذه الخاصية تحدد كيف ستتصرف الرسوم المتحركة (Animation) بعد أن تنتهي، سواء ستتوقف عند الحالة النهائية أو تعود إلى حالتها الأصلية.

القيم التي تأخذها:

ال none: هذا هو الوضع الافتراضي، حيث يعود العنصر إلى حالته الأصلية بعد انتهاء الرسوم المتحركة (Animation).

ال forwards: عند انتهاء الرسوم المتحركة (Animation)، يبقى العنصر في الحالة النهائية التي وصل إليها.

ال backwards: عند تحديد تأخير للرسوم المتحركة (animation-delay)، يتم تطبيق حالة البداية مباشرة.

ال both: يجمع بين تأثيري forwards و backwards.

مثال:

```
animation-fill-mode: forwards;
```

animation-direction

تحدد هذه الخاصية ما إذا كانت الرسوم المتحركة ستعمل في اتجاهها الطبيعي أو ستعمل بالعكس.

القيم التي تأخذها:

ال normal: (الوضع الافتراضي) الرسوم المتحركة تعمل من البداية إلى النهاية.

ال reverse: تجعل الرسوم المتحركة تعمل بالعكس، من النهاية إلى البداية.

ال alternate: تتناوب الرسوم المتحركة بين الاتجاه الطبيعي والعكسي. بمعنى، في كل تكرار، تبدأ في الاتجاه المعاكس.

ال alternate-reverse: تتناوب الرسوم المتحركة بين الاتجاه العكسي والطبيعي، ولكن تبدأ بالعكس.

مثال:

```
animation-direction: alternate;
```

كيفية تجميع الخصائص في سطر واحد باستخدام animation:

يمكنك دمج جميع هذه الخصائص في خاصية واحدة تسمى animation. ترتيب القيم في هذه الخاصية هو:

```
animation: [name] [duration] [timing-function] [delay] [iteration-count]
[direction] [fill-mode];
```

مثال:

```
.square {
  animation: moveAndChangeColor 4s ease-in-out 1s infinite alternate
forwards;
}
```

شرح المثال:

ال moveAndChangeColor: اسم الرسوم المتحركة.

ال 4s: مدة الرسوم المتحركة 4 ثوانٍ.

ال ease-in-out: الرسوم تبدأ ببطء، تسرع في الوسط، وتتباطأ في النهاية.

ال 1s: تأخير الرسوم المتحركة لمدة ثانية واحدة قبل أن تبدأ.

ال infinite: الرسوم المتحركة تستمر إلى ما لا نهاية.

ال alternate: الرسوم تتناوب بين الاتجاه الطبيعي والعكسي في كل تكرار.

ال forwards: تبقى الرسوم المتحركة في حالتها النهائية بعد الانتهاء.